Optimisation strategies for the design of experiments in materials discovery.

PhD dissertation

Simona Capponi

Supervisors:
Dr John Fearnley
Dr Vladimir Gusev
Prof Andy Cooper

Advisors:
Prof Rahul Savani
Dr Rasmus Ibsen-Jensen

Department of Computer Science
University of Liverpool
August 26, 2023

Contents

1 Introduction **1**
- 1.1 Bayesian optimisation for materials design . . . 2
  - 1.1.1 Bayesian inference . . . 3
  - 1.1.2 The strategy behind Bayesian optimisation . . . 4
  - 1.1.3 Gaussian processes as surrogate models . . . 5
- 1.2 Our contributions . . . 6
- 1.3 Structure of the thesis . . . 10

2 Background **12**
- 2.1 Bayesian Optimization . . . 12
- 2.2 Gaussian processes . . . 14
  - 2.2.1 Regression with Gaussian processes . . . 16
- 2.3 Examples of kernel functions . . . 19
  - 2.3.1 Model selection . . . 20
  - 2.3.2 Acquisition functions . . . 21
- 2.4 Batched Bayesian Optimization . . . 23
- 2.5 Bayesian optimisation in discrete spaces . . . 25

3 Methods for Integrating Predicted Data into Bayesian Optimization **27**
- 3.1 Introduction . . . 27
- 3.2 Problem statement . . . 28
- 3.3 Related work . . . 29
- 3.4 Our Methods . . . 33
  - 3.4.1 The exclusion radius method . . . 33
  - 3.4.2 Discrepancy prediction method . . . 33
- 3.5 Experimental Setup . . . 38
  - 3.5.1 The Predictors . . . 38
  - 3.5.2 Experimental parameters . . . 42
- 3.6 Results . . . 45

3.7 Conclusion 53

4 Enhanced Methods for Integrating Predicted Data into Bayesian Optimization 54
4.1 Introduction 54
4.2 The early switch exclusion radius method 56
4.2.1 Early switch method experimental settings 56
4.2.2 Results of the early switch method 56
4.2.3 Early switch: conclusions 66
4.3 The adaptive radius exclusion method 67
4.3.1 Results of the adaptive radius exclusion radius method 68
4.3.2 Adaptive radius exclusion method: conclusions 76
4.4 Predicting the discrepancy with GBR 82
4.4.1 The gradient boosting regressor model 82
4.4.2 Experimental results 84
4.4.3 Discrepancy prediction method with GBR: conclusions 89
4.5 Discrepancy prediction method with augmented real data 90
4.5.1 Experimental results 90
4.5.2 Discrepancy prediction method with augmented real data: Conclusions 98
4.6 Summary and Final Conclusions 99

5 Batched Bayesian Optimization for Molecular Structures 102
5.1 Introduction 102
5.2 Previous work 105
5.2.1 Thompson sampling BBO: TS-BBO 105
5.2.2 BBO based on Determinantal Point Processes (DPP-BBO) . . 106
5.2.3 LAW2ORDER 107
5.3 Molecular descriptors and similarity kernels 108
5.3.1 Coulomb descriptors and RBF kernels 109
5.3.2 SOAP descriptors and REMatch kernels 110
5.4 Experiments 113
5.4.1 Experimental setup 114
5.5 Results 116
5.6 Conclusions 125

6 Conclusions 126
6.1 Overview 126
6.2 Summary of experimental results 127
6.2.1 Methods for the exploitation of predicted data 127

6.2.2 LAW2ORDER BBO method for molecular properties optimisation . . . 131
6.3 Future work . . . 134
6.3.1 Exclusion radius and discrepancy prediction method for materials design . . . 134
6.3.2 Improving the LAW2ORDER method . . . 134
6.3.3 Towards application to biomolecules . . . 135
6.3.4 Software considerations . . . 135

A Results Tables for Methods to integrate Predicted Data into Bayesian optimisation 137
A.1 The Early Switch Method . . . 138
A.1.1 Simple regrets at low error regime, for the early switch method 138
A.1.2 Simple regrets at medium error regime, for the early switch method . . . 139
A.1.3 Simple regrets at high error regime, for the early switch method 140
A.1.4 Predicted regrets at low error regime, for the early switch method 141
A.1.5 Predicted regrets at medium error regime, for the early switch method . . . 142
A.1.6 Predicted regrets at high error regime, for the early switch method . . . 143
A.2 The Adaptive Radius Exclusion Method . . . 144
A.2.1 Simple regrets at low error regime, for adaptive radius exclusion method . . . 144
A.2.2 Simple regrets at medium error regime, for adaptive radius exclusion method . . . 145
A.2.3 Simple regrets at high error regime, for adaptive radius exclusion method . . . 146
A.2.4 Predicted regrets at low error regime, for adaptive radius exclusion method . . . 147
A.2.5 Predicted regrets at medium error regime, for adaptive radius exclusion method . . . 148
A.2.6 Predicted regrets at high error regime, for adaptive radius exclusion method . . . 149

B Regrets for the discrepancy prediction method with augmented real data 150
B.1 Average regrets for discrepancy prediction method with augmented real data . . . 151
B.1.1 Simple average regrets using a GP to predict the discrepancy . 152

B.1.2 Simple average regrets using a GBR to predict the discrepancy 153
B.1.3 Predicted average regrets using a GP to predict the discrepancy 154
B.1.4 Predicted average regrets using a GBR to predict the discrepancy 156
B.2 Final regrets for discrepancy prediction method with augmented real data 158
B.2.1 Simple final regrets using a GP to predict the discrepancy . . 159
B.2.2 Simple final regrets using a GBR to predict the discrepancy . 160
B.2.3 Predicted final regrets using a GP to predict the discrepancy . 161
B.2.4 Predicted final regrets using a GBR to predict the discrepancy 162

Bibliography **172**

List of Figures

3.1 High-level workflow for multi-fidelity BO method. Steps which differ from standard BO are in orange. The level of accuracy to use at the current BO step t is selected by the acquisition function. The best point $\mathbf{x}_{\text{opt}}$ is evaluated as the optimum point of the predictive mean at the highest level of accuracy. $\boldsymbol{\mu}_1$ 32

3.2 Left: workflow diagram of the exclusion radius method. Steps which differ from standard BO are in orange. Right: scheme of the exclusion radius method for a 1-dimensional problem. Real points are indicated with $\times$, and predicted points with a circle. Each time a new real point is acquired, a region centered in that point is computed according to point 7 of Algorithm 2. The predicted points which fall any of these regions are shown by grey circles: those points are removed from the data set of the surrogate model, $\mathcal{D}$, and will no longer be considered for the hyper-parameters optimization since the next BO step. The green circles indicate the predicted points which are outside of the aforementioned regions, and that will remain in $\mathcal{D}$. 36

3.3 Top left: workflow diagram of the discrepancy prediction method. Steps which differ from standard BO are in orange. The graph on the right shows a 1D example of the discrepancy prediction method: in the top graph the actual objective function f and the predictor p are plotted respectively with a continuous and a dash black line, while the red points indicate the set of initial predicted points. The real points are indicated with a black $\times$. Each time a real point $(\mathbf{x}_t, y_t)$ is acquired, the predicted output at $\mathbf{x}_t$ is computed, and the point $(\mathbf{x}_t, f(\mathbf{x}_t) - p(\mathbf{x}_t))$ is added to the data of the surrogate model for the discrepancy, GP_δ, as shown at point 8 of Algorithm 3. These additional predicted points, plotted in the top graph with green $\times$, are used to train GP_δ. The bottom graph shows the actual discrepancy over the whole search space and (red continuous line), while the grey dashed line and the blue dashed line plot the discrepancy predicted by GP_δ at $t = 0$ and $t = 5$ respectively, showing significant improvement as more data becomes available. 37

3.4 Top: the error parameters used to generate predictors for each function. N has been divided by Δf to partially normalize across the benchmarks. Bottom: example of predictors at the three error levels for the Michalewicz function. 41

3.5 Experimental results for all methods on all benchmarks. The curves for standard BO, for the discrepancy prediction method, and for multi-fidelity Bayesian optimisation experiments start at observation 5, as they are all initialized with 5 real points. The curves for the exclusion radius method start from 0 as no real points are used for the initialization. 47

3.6 Here the average regrets shown in Figure 3.5 are plotted on a linear scale. 48

3.7 The number of real observations needed to get within 5% of the optimal value. 49

3.8 Total computational time in seconds for each method. 49

4.1 Final **predicted** regrets for different values of radius parameter r and early switch iteration t_{sw} for all the benchmark functions. Blue and red squares indicate experimental settings at which the early switch method performs better and worse than the exclusion radius method respectively. 58

4.2 Final **simple** regrets for different values of radius parameter r and early switch iteration t_{sw} for all the benchmark functions. Blue and red squares indicate experimental settings at which the early switch method performs better and worse than the exclusion radius method respectively. 59

4.3 Predictive accuracy A_p for the early switch method, computed according to Eq. (4.3), in the **low** predictors' error regime. The values of A_p have been observed at BO step numbers 11, 21, 41, 51, 61 and 71. Each curve refers to experiment with a different setting for the parameter t_{sw}. The graphs show that, for each value of t_{sw}, A_p is minimum when the BO step number is equal to $t_{sw}+1$, i.e. just after all the predicted points are deleted. 61

4.4 Predictive accuracy A_p for the early switch method, computed according to Eq. (4.3), in the **medium** predictors' error regime. The values of A_p have been observed at BO step numbers 11, 21, 41, 51, 61 and 71. Each curve refers to experiment with a different setting for the parameter t_{sw}. The graphs show that, for each value of t_{sw}, A_p is minimum when the BO step number is equal to $t_{sw}+1$, i.e. just after all the predicted points are deleted. 62

4.5 Predictive accuracy A_p for the early switch method, computed according to Eq. (4.3), in the **low** predictors' error regime. The values of A_p have been observed at BO step numbers 11, 21, 41, 51, 61 and 71. Each curve refers to experiment with a different setting for the parameter t_{sw}. The graphs show that, for each value of t_{sw}, A_p is minimum when the BO step number is equal to $t_{sw}+1$, i.e. just after all the predicted points are deleted. 63

4.6 Average **predicted** regrets achieved by computing the best point at each iteration t among all the points returned by the acquisition function up to the t^{th} step. The results are plotted for the Ackley, Griewank and Michalewicz functions. 70

4.7 Average **predicted** regrets achieved by computing the best point at each iteration t among all the points returned by the acquisition function up to the t^{th} step. The results are plotted for the Rastrigin and Styblinski-Tang. 71

4.8 Average **simple** regrets achieved by computing the best point at each iteration t among all the points returned by the acquisition function up to the t^{th} step. The results are plotted for the Ackley, Griewank and Michalewicz functions. 72

4.9 Average **simple** regrets achieved by computing the best point at each iteration t among all the points returned by the acquisition function up to the t^{th} step. The results are plotted for the Rastrigin and Styblinski-Tang. 73
4.10 Predictive accuracy A_p for the adaptive radius exclusion method, computed according to Eq. (4.3), all predictors' error regimes. The values of A_p have been observed at BO step numbers 11, 21, 41, 51, 61 and 71. Each curve refers to experiment with a different setting for the parameter t_{sw}. The graphs show that, for each value of t_{sw}, A_p is minimum when the BO step number is equal to $t_{sw}+1$, i.e. just after all the predicted points are deleted. 77
4.11 Predictive accuracy of the GP surrogate model for the adaptive radius exclusion method is displayed in the 2-dimensional graphs. The 3-dimensional graphs below show the posterior variance of the surrogate model, calculated at the point $\mathbf{x}^*$ returned by the acquisition function at BO step numbers $t^* = \{15,\ 25,\ 35,\ 45,\ 55,\ 65,\ 75,\ 80\}$, as indicated at Section 4.3. Both top and bottom graphs plot data obtained for the Ackley and Griewank functions respectively. 78
4.12 Predictive accuracy of the GP surrogate model for the adaptive radius exclusion method is displayed in the 2-dimensional graphs. The 3-dimensional graphs below show the posterior variance of the surrogate model, calculated at the point $\mathbf{x}^*$, returned by the acquisition function at BO step numbers $t^* = \{15,\ 25,\ 35,\ 45,\ 55,\ 65,\ 75,\ 80\}$, as indicated at Section 4.3. Both top and bottom graphs plot data obtained for the Michalewicz and Rastrigin functions respectively. 79
4.13 Predictive accuracy of the GP surrogate model for the adaptive radius exclusion method is displayed in the 2-dimensional graphs. The 3-dimensional graphs below show the posterior variance of the surrogate model, calculated at the point $\mathbf{x}^*$, returned by the acquisition function at BO step numbers $t^* = \{15,\ 25,\ 35,\ 45,\ 55,\ 65,\ 75,\ 80\}$, as indicated at Section 4.3. Both top and bottom graphs plot data obtained for the Styblinski-Tang function. 80
4.14 Final **predicted** average regrets (top) and **simple** average regrets (bottom) for each level of predictors' error. Dashed lines represent the final average predicted regret reached with exclusion radius method. Each colour refers to a distinct benchmark function. 81
4.15 Comparison between the average **predicted** regrets obtained using a GP or a GBR surrogate model for the discrepancy. The results are plotted for all three predictors' error regimes for all the benchmark functions and compared to standard BO. 86

4.16 Comparison between the average **simple** regrets obtained using a GP or a GBR surrogate model for the discrepancy. The results are plotted for all three predictors' error regimes for all the benchmark functions and compared to standard BO. 87
4.17 Comparison between final average **predicted** regrets using a GP or a GBR surrogate model for the discrepancy. The results are plotted for all three predictors' error regimes for all the benchmark functions and compared to standard BO. 88
4.18 Comparison between final average **simple** regrets using a GP or a GBR surrogate model for the discrepancy. The results are plotted for all three predictors' error regimes for all the benchmark functions and compared to standard BO. 88
4.19 Average **simple** regrets for all the benchmark functions, using a **GP** surrogate model and a sampling threshold **T=0.3**. Each curve corresponds to a different value of the ratio λ/l. The black dotted curve refers to the discrepancy prediction method without any additional real data. 94
4.20 Average **simple** regrets for all the benchmark functions, using a **GBR** surrogate model and a sampling threshold **T=0.3**. Each curve corresponds to a different value of the ratio λ/l. The black dotted curve refers to the discrepancy prediction method without any additional real data. 95

5.1 Illustration of how a SOAP kernel descriptor is built: on the left side an element of the K^{SOAP} matrix is built from the local environments of the atoms i and j, H_i and H_j respectively. On the right side it is shown the whole SOAP kernel as a matrix where each row corresponds to an atom in the molecule, and it corresponds to $\mathbf{p}$, introduced in Eq. (5.16). The vector $\mathbf{p}$ is given by the concatenation of the pairwise power spectra defined in Eq. (5.14). 112
5.2 REMAtch kernel between two molecular structures, $\mathbf{A}$ and $\mathbf{B}$, which is related to the covariance matrix $\mathbf{C}(\mathbf{A}, \mathbf{B})$ via Eq. (5.18a). Each element of $\mathbf{C}(\mathbf{A}, \mathbf{B})$ is given by the SOAP similarity kernel between a local environment of $\mathbf{A}$ and a local environment of $\mathbf{B}$. 113
5.3 Comparison of the average regrets curves obtained with Coulomb descriptors and RFB kernel (left) and SOAP descriptors and REMatch kernels. Experimental results achieved with LAW2ORDER method at different value of the parameter $\mathbf{b}$ are compered with the results given by Thompson method (black line) 117

5.4 Average final regrets for all the optimisation tasks, for Coulomb descriptors, in linear scale. The curves plotted in the left column and in the right refer to results obtained with EI-LAW2ORDER and Thomp-LAW2ORDER BBO methods respectively. 122

5.5 Logarithm of average final regrets for all the optimisation tasks, for Coulomb descriptors. The curves plotted in the left column and in the right refer to results obtained with EI-LAW2ORDER and Thomp-LAW2ORDER BBO methods respectively. All regrets have been shifted by +1, so the values 0 on the y axis correspond to a null regret. 123

5.6 Computational time for the EI-LAW2ORDER, the Thomp-LAW2ORDER and the Thompson methods, using Coulomb descriptors. Time is expressed in hours. 124

5.7 Computational time for the LAW2ORDER and the Thompson methods, using Coulomb descriptors (left graph) and SOAP descriptors (right graph). Time is expressed in hours. 124

6.1 133

B.1 Average **simple** regrets for all the benchmark functions, using a **GP** surrogate model and a sampling threshold **T=0.5**. Each curve corresponds to a different value of the ratio λ/l. The black dotted curve refers to the discrepancy prediction method without any additional real data. 152

B.2 Average **simple** regrets for all the benchmark functions, using a **GBR** surrogate model and a sampling threshold **T=0.5**. Each curve corresponds to a different value of the ratio λ/l. The black dotted curve refers to the discrepancy prediction method without any additional real data. 153

B.3 Average **predicted** regrets for all the benchmark functions, using a **GP** surrogate model and a sampling threshold **T=0.3**. Each curve corresponds to a different value of the ratio λ/l. The black dotted curve refers to the discrepancy prediction method without any additional real data. 154

B.4 Average **predicted** regrets for all the benchmark functions, using a **GP** surrogate model and a sampling threshold **T=0.5**. Each curve corresponds to a different value of the ratio λ/l. The black dotted curve refers to the discrepancy prediction method without any additional real data. 155

B.5 Average **predicted** regrets for all the benchmark functions, using a **GBR** surrogate model and a sampling threshold **T=0.3**. Each curve corresponds to a different value of the ratio λ/l. The black dotted curve refers to the discrepancy prediction method without any additional real data. 156
B.6 Average **predicted** regrets for all the benchmark functions, using a **GBR** surrogate model and a sampling threshold **T=0.5**. Each curve corresponds to a different value of the ratio λ/l. The black dotted curve refers to the discrepancy prediction method without any additional real data. 157
B.7 Comparison between averaged final **simple** regrets $\hat{R}_s$ achieved with $T = 0.3$ or $T = 0.5$, using a **GP** surrogate model for the discrepancy. The values of $\hat{R}_p$ are plotted versus the value of λ/l. The black dashed line shows the averaged final **predicted** regret observed for the discrepancy prediction method. 159
B.8 Comparison between averaged final **simple** regrets $\hat{R}_s$ achieved with $T = 0.3$ or $T = 0.5$, using a **GBR** surrogate model for the discrepancy. The values of $\hat{R}_p$ are plotted versus the value of λ/l. The black dashed line shows the averaged final **predicted** regret observed for the discrepancy prediction method. 160
B.9 Comparison between averaged final **predicted** regrets $\hat{R}_p$ achieved with $T = 0.3$ or $T = 0.5$, using a **GP** surrogate model for the discrepancy. The values of $\hat{R}_p$ are plotted versus the value of λ/l. The black dashed line shows the averaged final **predicted** regret observed for the discrepancy prediction method. 161
B.10 Comparison between averaged final **predicted** regrets $\hat{R}_p$ achieved with $T = 0.3$ or $T = 0.5$, using a **GBR** surrogate model for the discrepancy. The values of $\hat{R}_p$ are plotted versus the value of λ/l. The black dashed line shows the averaged final **predicted** regret observed for the discrepancy prediction method. 162

List of Tables

3.1 The benchmark functions that we use 39

3.2 Experimental setup for each method. # Real and # Pred. denote the number of real and predicted points used to initialize the method. $\hat{A}/\Delta f$ denotes the predictor accuracies that were tested. r/l gives the values of the radius parameter for the exclusion radius method, while cost denotes the costs that were tested for the multi-fidelity method. . 44

3.3 Absolute values of the set up parameters for the exclusion radius method and for each benchmark functions. 44

3.4 Number of steps needed to get within 5% of the optimal point for low error predictors. 50

3.5 Number of steps needed to get within 5% of the optimal point for medium error predictors. 51

3.6 Number of steps needed to get within 5% of the optimal point for high error predictors. 52

4.1 Comparison between the best average **simple** regrets obtained with the exclusion radius method and with the early switch method for each of the benchmark functions, at low, medium and high levels of error. The values in bold indicate the best method for a given function. The columns r/l report the the value of the parameter r/l at which the best results have been achieved for each of the two methods. 64

4.2 Comparison between the best average **predicted** regrets obtained with the exclusion radius method and with the early switch method for each of the benchmark functions, at low, medium and high levels of error. The values in bold indicate the best method for a given function. The columns r/l report the the value of the parameter r/l at which the best results have been achieved for each of the two methods. 65

4.3 Comparison between the best average **simple** regrets obtained with the exclusion radius method and with the adaptive radius exclusion method for each of the benchmark functions, at low, medium and high levels of error. The values in bold indicate the best method for a given function. The value of the parameter r/l at which the best results have been achieved for the original exclusion method have been reported in the column with the same name. The adaptive radius exclusion method has always been performed setting $r/l = 0.05$. 74

4.4 Comparison between the best average **predicted** regrets obtained with the exclusion radius method and with the adaptive radius exclusion method for each of the benchmark functions, at low, medium and high levels of error. The values in bold indicate the best method for a given function. The value of the parameter r/l at which the best results have been achieved for the original exclusion method have been reported in the column with the same name. The adaptive radius exclusion method has always been performed setting $r/l = 0.05$. . . . 75

4.5 Comparison between the final **simple** average regrets obtained with the original discrepancy prediction method, the discrepancy prediction method with gradient boosting regressor, and the version of the same method with augmented real data. The regrets for each of these methods are indicated respectively with R_{discr}, R_{GBR} and $R^{\text{best}}_{\text{agument}}$. For this last one, only the best results are displayed, and the corresponding choice of the surrogate model for the discrepancy is reported in the column Method, where the values GPM or GBR indicate a Gaussian process or a gradient boosting regressor respectively. The values of the parameters T and λ/l giving $R^{\text{best}}_{\text{agument}}$ are also shown in the columns T and λ/l. 96

4.6 Comparison between the final **predicted** average regrets obtained with the original discrepancy prediction method, the discrepancy prediction method with gradient boosting regressor, and the version of the same method with augmented real data. The regrets for each of these methods are indicated respectively with R_{discr}, R_{GBR} and $R^{\text{best}}_{\text{agument}}$. For this last one, only the best results are displayed, and the corresponding choice of the surrogate model for the discrepancy is reported in the column Method, where the values GPM or GBR indicate a Gaussian process or a gradient boosting regressor respectively. The values of the parameters T and λ/l giving $R^{\text{best}}_{\text{agument}}$ are also shown in the columns T and λ/l. 97

5.1 A summary of our data sets. 114

5.2 Comparison final simple regrets obtained with the LAW2ORDER and Thompson BBO methods, using Coulomb descriptors and RBF kernel. The best result for each data set is reported in bold 119

5.3 Comparison final simple regrets obtained with the LAW2ORDER and Thompson BBO methods, using SOAP descriptors and REMatch kernel. The best result for each data set is reported in bold 120

A.1 Final **simple** average regrets in the **low** error regime. Results for the exclusion radius method radius are compared to the best results for the early switch method, obtained for tbest t_{best}. Values in bold indicate the lowest regret at each value of r/l. The absolute best result for each benchmark is underlined. 138

A.2 Final **simple** average regrets in the **medium** error regime. Results for the exclusion radius method radius are compared to the best results for the early switch method, obtained for tbest t_{best}. Values in bold indicate the lowest regret at each value of r/l. The absolute best result for each benchmark is underlined. 139

A.3 Final **simple** average regrets in the **high** error regime. Results for the exclusion radius method radius are compared to the best results for the early switch method, obtained for tbest t_{best}. Values in bold indicate the lowest regret at each value of r/l. The absolute best result for each benchmark is underlined. 140

A.4 Final **predicted** average regrets in the **low** error regime. Results for the exclusion radius method radius are compared to the best results for the early switch method, obtained for tbest t_{best}. Values in bold indicate the lowest regret at each value of r/l. The absolute best result for each benchmark is underlined. 141

A.5 Final **predicted** average regrets in the **medium** error regime. Results for the exclusion radius method radius are compared to the best results for the early switch method, obtained for tbest t_{best}. Values in bold indicate the lowest regret at each value of r/l. The absolute best result for each benchmark is underlined. 142

A.6 Final **predicted** average regrets in the **high** error regime. Results for the exclusion radius method radius are compared to the best results for the early switch method, obtained for tbest t_{best}. Values in bold indicate the lowest regret at each value of r/l. The absolute best result for each benchmark is underlined. 143

A.7 Final average **simple** regrets comparison between exclusion radius method and the adaptive radius exclusion method in **low** noise regime. For the adaptive radius exclusion method only the best results are reported, together with the value of t^* at which they have been achieved. The best result for each benchmark function is in bold. 144
A.8 Final average **simple** regrets comparison between exclusion radius method and the adaptive radius exclusion method in **medium** noise regime. For the adaptive radius exclusion method only the best results are reported, together with the value of t^* at which they have been achieved. The best result for each benchmark function is in bold. . . 145
A.9 Final average **simple** regrets comparison between exclusion radius method and the adaptive radius exclusion method in **high** noise regime. For the adaptive radius exclusion method only the best results are reported, together with the value of t^* at which they have been achieved. The best result for each benchmark function is in bold. 146
A.10 Final average **predicted** regrets comparison between exclusion radius method and the adaptive radius exclusion method in **high** noise regime. For the adaptive radius exclusion method only the best results are reported, together with the value of t^* at which they have been achieved. The best result for each benchmark function is in bold. 147
A.11 Final average **predicted** regrets comparison between exclusion radius method and the adaptive radius exclusion method in **medium** noise regime. For the adaptive radius exclusion method only the best results are reported, together with the value of t^* at which they have been achieved. The best result for each benchmark function is in bold. . . 148
A.12 Final average **predicted** regrets comparison between exclusion radius method and the adaptive radius exclusion method in **high** noise regime. For the adaptive radius exclusion method only the best results are reported, together with the value of t^* at which they have been achieved. The best result for each benchmark function is in bold. 149

Chapter 1

Introduction

In the last two decades there has been a considerable effort to accelerate scientific discovery by the use of artificial intelligence and machine learning techniques. Artificial intelligence (AI) is the field of developing computers and robots that are capable of behaving in ways that mimic and human capabilities, like the ability to coordinate motor actions in the real world, or to derive information from data, all without human intervention.

Machine learning (ML) is an area of artificial intelligence that focuses on the development of algorithms and models that enable computers to recognize patterns in data, and make predictions or decisions based on these patterns, without being explicitly programmed.

In the context of applied science AI is increasingly employed in several ways [48], for example to develop robotic platforms capable of performing experiments autonomously, while both AI and ML are widely used to make decisions on the experiments to perform, based on the analysis of input data. Machine learning and artificial intelligence algorithms can analyze and process large amounts of data, and consequently identify patterns not recognizable by humans [60]. Also, they can provide a prediction for physical systems before these are actually observed in the real world, augmenting the information available to researchers.

Furthermore, using ML models for decision making allows us to take advantage of parallel computational resources to handle a large number of decisions simultaneously.

Integrating robotic platforms and artificial intelligence in the process of scientific discovery has also the advantage to let human experts to focus on the most complex and strategic tasks, once repetitive tasks are assigned to machines.

Presently, this approach to design experiments has already been implemented successfully in the field of applied sciences like Biology [74, 89, 116], drug discovery [25, 109], Engineering [112], Physics [28, 54], Chemistry and Materials Design [8, 22].

In Chemistry and Materials Science the application of AI algorithms has become extremely popular, and it is often combined with the automatisation of the experimental workflow via robotic platforms. This is due mainly to the growing implementation of high throughput experiments, a methodology consisting in the synthesis and characterization of a large number of compounds, which can vary from thousands up to millions. This approach allows researchers to rapidly identify compounds with the desired properties. High throughput Chemistry is performed in an automated fashion in order to reduce the required experimental time, thanks also to the possibility of executing multiple tasks in parallel. However, as the chemical space is large and high dimensional, high throughput alone might not be enough to accomplish the optimisation of chemical properties in the desired time window.

To overcome this problem a *closed loop* approach has been recently introduced, where robotic automatisation is combined with the use of decision-making machine learning algorithms, designed to find the optimal solution exploring just a portion of the search space [16, 18, 27]. A closed loop is in fact an active learning workflow, where a search algorithm provides recommendations of the experiments to perform, and such an algorithm is in turn informed with the data coming from real world experiments.

As both the set up of the robotic platforms and the execution of ML and deep learning algorithms are becoming more affordable, this experimental paradigm is becoming more common. An important factor is the increasing availability of computational resources as GPUs, which have strongly accelerated the training and testing of ML models. For example, the use of first principle simulations for materials discovery has been largely replaced by ML and AI methods with remarkably lower computational cost [11, 12, 99].

Generally, a closed loop experiment requires three components: a robotic platform, software to analyse the experimental results, and an algorithm to recommend new experiments.

The work described in this thesis is focused on this last element. Specifically, we develop new Bayesian optimisation methods designed to be applied to materials discovery, we test them on synthetic and real world tasks, and we compare them against already established techniques

1.1 Bayesian optimisation for materials design

Bayesian optimisation is a global optimisation method designed to search the optimum of black-box functions that are expensive to evaluate. Black-box functions are functions for which relationship between a set of inputs and the corresponding set of outputs is not known, ans as a consequence they cannot be expressed in an

analytical form. This makes their optimisation challenging, as methods based on gradient descent are not applicable.

To address this issue gradient-free optimisation methods have been largely investigated in literature, among these random search, simulated annealing [58], particle swarm optimization [56], Nelder–Mead simplex method [78, 96] and genetic algorithms [45, 97]. The application of these frameworks however might become unfeasible if the evaluation cost of the objective is elevated, in which case it is necessary to reduce the number of evaluations required to reach convergence[1].

Bayesian optimisation achieves that by focusing the search in regions of the input space where the probability of finding the optimum is high. The location of these points is computed based on the information from previous evaluations. In practice however, exploration outside these regions is allowed in order to gain more information about the objective function, so to improve the decision of the regions to sample and avoid the optimisation process to get stuck.

This makes BO particularly suitable for design of experiments, which very often implies an optimisation task where the objective function is represented by the relationship between a set of controllable variables and outcome of the experiment. Typically a such relationship is indeed a black-box function, and its evaluation requires us to perform real world experiments, which can be time consuming and expensive in terms of material cost. Another advantage of BO is that it can handle noisy functions [75, 82], which is very important when the data is obtained via real world experiments, which are intrinsically noisy.

1.1.1 Bayesian inference

Bayesian inference is a statistical model where an initial hypotheses is updated as more information becomes available, using Bayes' theorem.

Let us assume that we formulate a certain hypotheses, A, and we want to compute the probability that A actually occurs. Before any evidence is taken into account A has an initial probability $P(A)$, called *prior probability*, which is usually defined based on some assumption about A. Let us assume that, after defining $P(A)$, new related data is obtained: the observation of this data is indicated as the event B. Bayes' theorem states that the conditional probability $P(A|B)$ that A occurs, given that B has occurred, is given by:

[1] As the actual global optimum might never be found in finite time, a criterion to establish if the optimisation process has converged is set a priori. Most commonly the optimisation is considered to have converged if it has reached a point where further iterations do not improve the objective function's value by more than a given threshold.

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)} \tag{1.1}$$

where $P(A|B)$ is called the *posterior probability*, $P(A)$ is the prior probability over A, while $P(A|B)$, which is called the *likelihood*, is the probability of observing B given that the hypotheses A is true. Thus the likelihood indicates the compatibility of B, i.e. the observed data, with the hypotheses A. $P(B)$ is the probability to observe B regardless of any hypotheses.

1.1.2 The strategy behind Bayesian optimisation

The surrogate model. The main strategy of Bayesian optimisation to deal with a black-box objective function is to model it with a probabilistic model, called *surrogate model*, as it replaces the actual objective. Typically the surrogate model is a parametric function, like the Gaussian processes introduced in Section 1.1.3, describing the probability over the infinitely many objective functions that are compatible with the currently observed data. A specific surrogate model is identified by a set of parameters $\boldsymbol{\theta}$: a given set of parameters represent a hypotheses about the objective function.

Applying Bayesian inference over the parameters $\boldsymbol{\theta}$ we can find the most probable function to have generated the data observed. For this purpose we initially need to compute the most likely surrogate model given the set of all the data acquired so far, which from now on we indicate as $\mathcal{D}$. This can be done exploiting Bayes' theorem: in the context of Bayesian optimisation each set of parameters $\boldsymbol{\theta}$ of the surrogate model represents a given hypotheses on the actual objective, while the data set $\mathcal{D}$ represents the evidence. Thus Eq. (1.1) becomes:

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})} \tag{1.2}$$

The most probable values for the parameters $\boldsymbol{\theta}$ is thus calculated by maximising $p(\boldsymbol{\theta}|\mathcal{D})$ in Eq. (1.2). Once the surrogate model has been inferred, it is possible to estimate the most probable value of the objective, as explained in Chapter 2.

As new evaluations of the objective become available, Bayesian inference is repeated, and the the surrogate model is updated.

The acquisition function. At each stage of the optimisation process one or more new input points are selected, based on a policy that chooses the most useful points with respect to the optimisation task at hand, and the objective function is evaluated at these points.

In BO this policy is implemented by maximising a utility function, called the *acquisition function*, which uses the surrogate model to evaluate the next evaluation points. This adaptive sampling strategy is meant to allow us to find the optimum point by exploring only a small portion of the search space, reducing the number of function evaluations. However acquisition functions are designed in such a way that a portion of points is also sampled from unexplored regions of the search space. In other words a balance between exploration and exploitation is enforced, in order to avoid that poor estimations suggested by the surrogate model results in the excluding the actual optimal region.

Application of BO to design of experiments. Applying BO to design of experiments requires the following steps:

1. Defining the objective to optimise. This is a physical quantity or a property to minimise or maximise, for example maximising the yield of a chemical reaction.

2. Defining the input variables, which are the controllable variables of the experiment to be tuned to optimise the objective. Typically these are environmental variables like temperature, concentration and time, or they can be the composition and the structure of the molecules involved, as shown in Chapter 5.

3. Building a surrogate model of the experimental outcome as a function of the controllable variables.

4. Using the acquisition function to suggest the new conditions in which to perform the next experiment.

5. Performing the experiment and measuring the outcome.

6. Updating the surrogate model.

1.1.3 Gaussian processes as surrogate models

The most common surrogate model used in Bayesian optimisation is the *Gaussian process* (GP), which is a stochastic process representing the probability distribution over all the possible objective functions.

GPs can be seen as an extension of multivariate normal distributions to the space of functions. However, while a normal distribution is a stochastic process defined over individual data points, Gaussian processes model a probability distribution of functions defined over these points. Thus Gaussian processes are a family of infinite-dimensional stochastic processes, and they are fundamentally different concepts from

normal distributions. For this reason we will indicate them differently: $\mathcal{N}(\cdot)$ is a normal distribution, while $\mathcal{GP}(\cdot)$ is a Gaussian process.

As a consequence of the distribution being Gaussian, GPs are fully characterised by a mean and a covariance function, which we indicate respectively with $\mu(\mathbf{x})$ and $k(\mathbf{x}, \mathbf{x}')$. The mean represents also the most probable function, the covariance represents the related uncertainty:

$$f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{1.3}$$

Given a finite set of data points, there are infinite possible functions that could have generated them, but via Bayesian inference it is possible to determine the most likely one, as well as the related uncertainty. The possibility of determining the uncertainty over the modelled objective function is one of the most important advantages of Gaussian processes.

GPs allow us to incorporate prior knowledge, which can be done in several ways. A possible strategy is choosing the analytical form for the covariance function, so as to encode general assumptions: for example the belief that that f is periodic can be expressed by a periodic covariance function, while selecting a radial basis function for the covariance implies that f is smooth.

Alternatively, prior knowledge can be encoded by defining prior distributions over the objective that reflect prior beliefs about the analytical form of objective. In practice this is achieved by defining an initial GP having a mean with a given analytical form.

Finally, prior knowledge can be incorporated by augmenting the training data, for example by adding extra data coming from similar optimisation tasks that have been previously solved, or synthetic data points that provide an approximation of the actual objective function. This last method has been used in the work described in Chapters 3 and 4.

1.2 Our contributions

In this work we develop Bayesian optimisation methods developed to be applied to the field of materials design, aiming to further accelerate the optimisation process. Two possible approaches to achieve this goal are introducing prior knowledge into Bayesian optimisation or parallelizing the evaluation of the objective function. The introduction of prior knowledge into BO is intended to improve the prediction accuracy of the aforementioned surrogate model.

The motivation behind the second approach is the assumption that evaluating the objective is expensive, in terms of time or material cost, which is certainly true for experimental evaluations. Fortunately it is often the case that experiments can

be performed in parallel, and thus it is possible to execute multiple recommended experiments simultaneously. In this scenario it is convenient to use *batch* BO, or BBO, a variant of the BO, where multiple points are selected at each optimisation step.

We explore both approaches: the study reported in Chapters 3 and 4 is focused on the integration of inexpensive predicted data into the optimisation process. The approaches described in these chapters are based on sequential Bayesian optimisation over continuous search spaces. On the other hand, the work described in Chapter 5 is focused on the development of a BBO technique to address materials design tasks defined in a discrete space.

Integrating Predicted Data into Bayesian optimisation. The core idea of this work, which has been published at the BNAIC 2021 conference [1], is to initialize the optimisation process with predicted data, generated by an approximate model of the actual objective function to optimize. This model, which we call a *predictor*, is assumed is assumed to be much less expensive to evaluate than the actual objective, so that an arbitrary number of predicted points can be exploited to warm start the optimisation, virtually at no extra cost. Unfortunately, in real the world inexpensive predictors may suffer from low accuracy, and to take this into account we assumed that the predictors have a non negligible predictive error. In order to be as general as possible, we did not formulate any assumption about the predictors' errors, nor about the consistency of such errors across the search space. For this reason it is possible that even a predictor with small average error[2] provides very inaccurate predictions in some regions of the space.

More specifically, two methods are proposed, dealing differently with the inaccuracy of the predictor. The first one, called *exclusion radius method*, deletes some of the predicted points at each step of the optimisation process. Indeed, while the predicted points allow us to define an initial surrogate model that reflects the prior information available, their presence in the data set could be detrimental later on in the optimisation due to their inaccuracy. We thus built this method so that as more real data is acquired, BO relies more and more on the evaluations of the actual objective and less and less on the predicted data. However, since the sampling of the search space is not uniform, we might still have large regions where the objective f is completely unexplored. In these regions the surrogate model has poor information about f, and the related uncertainty is high, so approximate estimations coming from the predictor are still valuable. For this reason we set a maximum distance from the real points behind which predicted points are kept. Operationally, we elim-

[2]As described in Chapter 3, we estimated the average error of each predictor by computing the absolute value of the difference between predicted and real data over a grid of 30000 points.

inate predicted points only if they fall within a spherical region having a predefined radius r, which we call the *exclusion radius*. The value of r is a hyper-parameter of the exclusion radius method, which needs to be tuned to achieve good performance of the method. As will be shown in Chapter 3, the optimal choice for r depends on the accuracy of the predictor: if predictors with low level of accuracy are used, the method performs better with higher values of r, compared to the case where more accurate predictors are used. This is because increasing r leads to a faster elimination of the predicted points, which is beneficial if we generated predicted points with low accuracy and thus low informativeness.

On the other hand, as the initial set of predicted points is spread over all the search space, the value of the exclusion radius must be set taking into account also the extension of such space. For example we set r to be a fixed fraction of the length of one dimension of the search space, which in our case was a 2-dimensional square.

The second method we propose, called *discrepancy prediction method* , estimates the difference between the predictor and the actual objective, which we call *discrepancy*, and corrects the predicted points accordingly. To predict the discrepancy we use a second surrogate model. One advantage of the discrepancy prediction method with respect to the exclusion radius method is the absence of extra hyper-parameters to tune, but, as we will show, the exclusion radius method performs better overall.

Both these approaches have the advantage of being fast and conceptually simple, and they generally outperform the related strategies published in literature, like multi-fidelity BO, while using significantly less computational time

In Chapter 4 we propose a few variants of the exclusion radius method, in order to explore optimal rates at which to discard predicted data, as soon as their presence is more detrimental than helpful for the optimisation process. Additionally, we investigate strategies to optimise the correction of the predictor's error, with the purpose of improving the discrepancy prediction method.

For the exclusion radius method we define two heuristic approaches to delete all the predicted points before the BO process ends. The first one is the *early switch* method, a naive strategy where all the predicted data is erased at a given iteration number, which is set a priori, and which is a parameter of the method. The second one is called *adaptive radius exclusion method*, and it works as exclusion radius method, except that the radius r is not kept constant, but it is increased by a constant amount at each iteration. As the early switch method, adaptive radius exclusion aims to discard all the predicted points at a given step of the optimisation process, and the increase rate of the radius r is adjusted accordingly. However, this second method aims to do progressive elimination We also propose two variants for the discrepancy method: one consists in predicting the discrepancy using a gradient boosting regressor model instead of a GP, while in the other one we introduce a stochastic criterion to establish at each iteration whether the next point to evaluate

should be given by the acquisition function or a random point should be selected, within a neighbourhood of the current best point. The idea behind this strategy is to improve the predictive accuracy of the discrepancy in a region of the search space where the probability to find the optimum is higher than elsewhere. As the extra evaluations are counted against the total budget of observations, this last method effectively increases the degree of exploitation in the sampling of the points to evaluate compared to the original discrepancy prediction method.

Batched Bayesian Optimization for Molecular Structures. The second approach we investigated in order to accelerate the Bayesian optimisation process for materials design is the application of batch BO, or BBO, and it is fully reported in Chapter 5.

As discussed in Chapter 2, the advantage of BBO over sequential BO is the possibility to acquire a larger number of observations potentially at no extra computational time, wherever parallel computational resources are available. For this reason several studies have applied BBO to the design of experiments. However, while several frameworks have been proposed to solve problems defined in a continuous search space, the application of BBO to discrete spaces has so far received little attention. To fill this gap we explore the applicability of LAW2ORDER [79], a state of the art BBO method for discrete spaces, and we revise it to be applicable to materials properties optimisation problems. For this purpose we integrate it with similarity kernels appropriate for chemical applications. We also asses the performance of our adjusted version of the method by benchmarking on different molecular data set, and comparing the outcome with the results obtained with Thompson sampling based BBO, a method which has already been successfully applied to the fields of Chemistry and Materials Science [43, 95, 107].

The LAW2ORDER BBO method belongs to the family of BBO approaches relying on determinantal point processes to maximise the diversity among all the points which will be evaluated in parallel. Diversity is an important requirement for BBO to be really advantageous over BO: indeed if the batch points are too similar, acquiring multiple points at the same step will result in no relevant gain in information. The Thompson sampling based BBO for example, is a poorly exploratory method, and this potentially diminishes the advantage of parallelism. On the contrary, previous works using determinantal point processes (DPPs), briefly described in Chapter 5, enforce only diversity.

Differently from these DPPs based approaches, LAW2ORDER reintroduces also the acquisition function in the sampling policy, as it will be described in Section 5.2.3. The purpose is to take into account also the informativeness in the points sampling. LAW2ORDER is originally formulated for permutation problems, and the rationale

for applying it to materials design is that the search space of permutation problems not only is discrete, but it is also large, analogously to many materials design tasks.

We also explored the impact of the chosen *molecular descriptors* on the performance of this method. Molecular descriptors are mathematical representations of molecules' properties, while *similarity kernels*, closely related to the variance of GPs, quantify the similarity between input variables, as explained in Chapter 2. The choice of kernel to encode similarity between molecules is in part conditioned by the choice of molecular descriptors, as will be shown in more details in Chapter 5.

For our work we chose two types of descriptors, the Coulomb descriptors and the SOAP descriptors, which are described in Chapter 5, each one of them used in combination with a different similarity kernel.

Comparison between LAW2ORDER method and Thompson sampling method shows that the first one performs better than the second for Coulomb descriptors only, while for the SOAP descriptors the performances of the two methods are comparable. With both methods, Coulomb descriptors give better results than SOAP descriptors.

1.3 Structure of the thesis

This thesis is structured in six chapters, as described below.

Chapter 2: Background. This chapter provides the theoretical background related to the work described in the following chapters. The topics covered are Bayesian optimisation and Gaussian processes, as well as BBO. Furthermore there will be a short section about the previous work published in the literature to extend BO to discrete spaces. This topic is preliminary to the content of Chapter 5, although the most closely related techniques in discrete spaces are described in that chapter.

Chapter 3: Methods for Integrating Predicted Data into Bayesian optimisation. Chapter 3 is a slightly extended version of the paper presented at the BNAIC conference, and it is focused on the results and the implementation details of the exclusion radius method and the discrepancy prediction method. As the proposed methods are benchmarked against a multi-fidelity approach, a general introduction to the framework of multi-fidelity optimisation is also given.

Chapter 4: Enhanced Methods for Integrating Predicted Data into Bayesian Optimization. Several variants of the exclusion radius method and discrepancy prediction method have been investigated, and the related results are reported in this chapter, and compared to the original methods. In some cases this comparison is complemented with additional experiments intended to assess the causes of the

observed differences in the performance of each method. Although all the methods reported in this chapter do not represnt an improvement with respect to their original versions, described in the previous chapter, we illustrate them here for completeness.

Two modified versions are proposed for the exclusion radius and for the discrepancy prediction methods respectively. For the latter we also explored the integration of gradient boosting descent (GBR) into BO, as predictive model for the estimation predictors' inaccuracy from the available data. A brief introduction to ensemble methods and GBR is thus provided.

Chapter 5: Batched Bayesian Optimization for Molecular Structures. This chapter is dedicated to the description of our work on batch BO for materials design in discrete spaces. A summary of previous work is reported at the beginning of this chapter, including a description of the Thompson sampling method, against which LAW2ORDER is compared. For a better understanding of LAW2ORDER, which is based on stochastic models called *determinantal point processes* (DPPs), a brief overview of DPPs will precede the description of the experimental results. Furthermore, the concept of *chemical descriptors* will be introduced, and the descriptors used for our work, namely Coulomb matrices and SOAP descriptors, will be illustrated.

Chapter 6: Conclusions. This chapter reports a summary of all the main findings of all the chapters and provides the overall conclusion of all the work described in this thesis.

Chapter 2

Background

This chapter is focused on the description of Bayesian optimisation (BO), which is at the base of all the work described in this thesis. We start with a description of the basic principles of BO, which includes a brief illustration of its two main components, surrogate models and acquisition functions, and Gaussian process surrogate models will be introduced: the content related to Gaussian process is adapted from the Book [115], and it follows the same notations.

Significant parts of the presentation of the background material are taken from the book

Following this general part, we will introduce *batched* Bayesian optimisation (BBO) and Bayesian optimisation in discrete spaces, as these topics are preliminary to the work reported in Chapter 5.

2.1 Bayesian Optimization

Bayesian optimisation (BO) is a derivative free method developed to solve global optimisation problems where the objective function f satisfies the following properties:

- f is a black-box function, which usually cannot be expressed by an analytical form.
- f may be expensive to evaluate.
- f may not be differentiable.
- f may have multiple local optima.

All of the optimisation tasks considered in this work are formulated as minimisation problems[1] : given the objective function $f : \mathbf{x} \in \mathcal{X} \rightarrow y$, the problem to solve is

[1]Maximisation problems are converted into minimisation ones by transforming $f \rightarrow -f$.

formulated as:

$$\mathbf{x}_{\text{opt}} = \arg\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}), \tag{2.1}$$

where $\mathcal{X}$ is the whole search space. BO is an iterative optimisation method which deals with expensive black-box functions with two fundamental strategies:

1. The use of a surrogate model, $\mathcal{M}$, to approximate the unknown objective, at significantly lower evaluation cost. $\mathcal{M}$ provides an estimation of the real value of f by combining the data already observed with pior beliefs about f. This estimation is updated during the optimisation process, every time new data becomes available.

2. The use of a utility function, called *acquisition function*, which evaluates the best point $\mathbf{x}^*$ where to query f at the next step, for the purpose of finding the optimum value.

Formally, step 2 is executed by maximising the acquisition function, which from now on will be denoted by a:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathcal{X}} a(\mathbf{x}). \tag{2.2}$$

In this way the original optimisation problem over the objective f is solved by solving a maximisation problem over the function a, which is much cheaper to evaluate. The purpose of introducing the acquisition function is to apply a strategy for the selection of the points at which f will be evaluated, so as to reduce the number of steps required to approach the optimum point. Such a strategy uses the information provided by the surrogate model. For example, in most cases $\mathcal{M}$ is a Gaussian process (GP), which is a stochastic process defined in the function space, describing the probability distribution over the objective, and which is fully characterised by a mean and a covariance. In this case the acquisition function combines the mean and covariance into a criterion that directs the search for $\mathbf{x}^*$. GPs will be presented in Section 2.2. However other choices for the surrogate model are also possible, like random forests or neural networks. We used a GP, as our goal aimed to develop methods suitable for materials design, where the objective is expensive and the observations are the outcome of real world experiments, which are intrinsically noisy.

The Bayesian optimisation process. In its basic form BO operates in three phases, as shown in Algorithm 1.

Firstly the surrogate model $\mathcal{M}$ is used to learn the latent objective function f from the data set of previous observations. This data set is called *training data set*, as it is used to train the model, and here it is denoted by: $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1,\dots,n}$.

The second phase consists in selecting the next query point by maximizing the acquisition function. At the third phase the objective function is evaluated at the new point $\mathbf{x}^*$ returned by the acquisition function according to Eq. (2.2), the data set $\mathcal{D}$ is augmented with the point $(\mathbf{x}^*, f(\mathbf{x}^*))$, and the surrogate model $\mathcal{M}$ is updated using the updated data set. Specific examples of acquisition functions will be discussed in Section 2.3.2. All three steps are repeated until a given stopping condition is reached, typically after a given number of steps have been completed.

Algorithm 1: Bayesian Optimization

Input: Objective function f, stopping condition
Output: Optimum point $\mathbf{x}_{opt}$

Initialize the model data set $\mathcal{D}_0$
for $n \leftarrow 0, 1, ...$ **do**
  Update surrogate model $\mathcal{M}$ the current data set using $\mathcal{D}_n$
  Select new ${\mathbf{x}^*}_{n+1}$ by optimizing the acquisition function a:

$$ {\mathbf{x}^*}_{n+1} = \arg\max_{\mathbf{x} \in \mathcal{X}} a(\mathbf{x}|\mathcal{D}_n) $$

  Compute $y_{n+1} = f(\mathbf{x}_{n+1})$
  Augment the data set $\mathcal{D}_{n+1} = \{\mathcal{D}_n, (\mathbf{x}_{n+1}, y_{n+1})\}$
  if *stopping criterion is reached* **then**
    break
  end
end

2.2 Gaussian processes

Gaussian processes are a class of non-parametric regression models.

Formally, a GP is defined as a collection of random variables, any finite combination of which is a joint Gaussian distribution. In other words, a GP can be considered to be a multivariate normal distribution over functions, which is thus completely specified by its mean $\mu(\mathbf{x})$ and its covariance function (or kernel function), $k(\mathbf{x}, \mathbf{x}')$:

$$ f(\mathbf{x}) \sim \mathcal{GP}\left(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\right) \tag{2.3} $$

where

$$\mu(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \tag{2.4}$$

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') &= \operatorname{cov}(f(\mathbf{x}), f(\mathbf{x}')) \\ &= \mathbb{E}\left[(f(\mathbf{x}) - \mu(\mathbf{x}))(f(\mathbf{x}') - \mu(\mathbf{x}'))\right] \end{aligned} \tag{2.5}$$

Eq. (2.5) shows that the mean of the GP, also called *predictive mean*, is the expected value for the objective function and, being the probability distribution normal, also the most probable value. The covariance function $k(\mathbf{x}, \mathbf{x}')$ represents instead the uncertainty over the estimation $\mathbb{E}[f(\mathbf{x})]$. If no data has been observed yet, $\mu(\cdot)$ represents the *prior beliefs* over f, i.e. it reflects the initial assumptions on the objective. Usually however, no prior assumptions are given, and this is expressed by fixing the initial value of the predictive mean to a constant, typically $\boldsymbol{\mu_0} = \mathbf{0}$, where $\boldsymbol{\mu_0}$ is called the *prior mean*.

Gaussian Prior. A *Gaussian prior* is a GP which describes the probability of a given observation y to occur, independently from previously acquired data. This probability is given by:

$$p(y) = \mathcal{N}\left(y|\mu(\mathbf{x}), k\left(\mathbf{x}, \mathbf{x}'\right)\right) \tag{2.6}$$

where $p(y)$ is the Gaussian prior. From Eq. (2.6) it is possible to see that, once the prior mean is set, the covariance function fully determines the Gaussian prior: the prior beliefs about our objective are fully encoded by $k(\mathbf{x}, \mathbf{x}')$ alone.

Posterior distribution. The predictions given by the prior can be improved by conditioning $p(y)$ over the training data set $\mathcal{D}$: this corresponds to restricting the prior distribution to contain only those functions which agree with the observed data points. The resulting probability distribution over functions is called the *posterior* distribution, and it is given by $p(y^\star|\mathbf{x}^\star, \mathcal{D}, \theta)$, where $\mathbf{x}^\star$ is a test input, while $y^\star = f(\mathbf{x}^\star) + \epsilon$ is a noisy evaluation of the objective function. The noise ϵ is assumed to be independent and identically distributed: $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$. The vector $\boldsymbol{\theta}$ is a set of parameters which specifies the kernel function, as described in Section 2.2.1. The posterior distribution is again a GP:

$$p(y^\star|\mathbf{x}^\star, \mathcal{D}) = \mathcal{GP}(y^\star|\mu(\mathbf{x}^\star), k_{post}(\mathbf{x}^\star, \mathbf{x}^\star)) \tag{2.7}$$

where

$$k_{post}(\mathbf{x}^\star, \mathbf{x}^\star) = \operatorname{cov}(y^\star)$$

k_{post} is the *posterior covariance*, while $\mu(\mathbf{x}^\star)$ is the *posterior mean*. Both the posterior variance and the posterior mean are related to the kernel function according to:

$$\mu(\mathbf{x}^\star) = k(\mathbf{x}^\star, X)^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y} \tag{2.8}$$

$$\sigma^2(\mathbf{x}^\star) = k(\mathbf{x}^\star, \mathbf{x}^\star) - k^\top(\mathbf{x}^\star, X)(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}k(\mathbf{x}^\star, X), \tag{2.9}$$

where X is the set of input points contained in the training data set $\mathcal{D}$, and $\mathbf{K}$ is the matrix with elements $K_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$. The term σ_n^2 accounts for possible errors in the observations y, due to presence of the noise ϵ in the measurements of the objective f.

2.2.1 Regression with Gaussian processes

The task of predicting values of the latent objective function at a set of points is in fact a regression problem. It can be formally solved in the specific case where the objective can be expressed as a linear function:

$$f(\mathbf{x}) = \mathbf{x}^\top\mathbf{w}, \tag{2.10}$$

where $\mathbf{w}$ is a vector of weights of the linear model. This class of regression tasks is called *linear Bayesian regression*. Bayesian linear regression problems can be performed by optimising $\mathbf{w}$ over the observed data, and they have an explicit solution. For problems where Eq. (2.10) does not hold, it is still possible to map the original objective function to a linear relationship via the so called *kernel trick*. Here we first introduce linear Bayesian regression briefly, and then we describe the extension of regression with Gaussian processes to the general case.

Bayesian linear regression

As mentioned before, inferring the objective f under the assumption expressed by Eq. (2.10) is equivalent to inferring the weights $\mathbf{w}$. As it will be shown below this is possible by evaluating the posterior distribution over $\mathbf{w}$ via the **Bayes' theorem**. If we denote with $\mathbf{y}$ a vector of noisy observations of f at points $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, and we have a prior distribution over the weights, $p(\mathbf{w})$, the posterior distribution conditioned over the data $(X, \mathbf{y})$ can be computed via Bayes' theorem, and it is given by:

$$p(\mathbf{w}|X, \mathbf{y}) = \frac{p(\mathbf{y}|X, \mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|X)}, \tag{2.11}$$

where $p(\mathbf{w}|X, \mathbf{y})$ and $p(\mathbf{w})$ are respectively the posterior and the prior probabilities over the weights, $p(\mathbf{y}|X, \mathbf{w})$ is the the likelihood over $\mathbf{y}$, i.e. the probability density over the observations given the parameters $\mathbf{w}$, and $p(\mathbf{y}|X)$ is the marginal likelihood, which is given by:

$$p(\mathbf{y}|X) = \int p(\mathbf{y}|X, \mathbf{w})p(\mathbf{w})d\mathbf{w}. \tag{2.12}$$

The prior over the weight is generally given by a normal probability distribution centered in 0: $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_p)$. Finally, the explicit expression for the posterior over $\mathbf{w}$ is given by [115]:

$$p(\mathbf{w}|X, \mathbf{y}) \sim \mathcal{N}\left(\bar{\mathbf{w}}, A^{-1}\right), \tag{2.13}$$

where

$$\bar{\mathbf{w}} = \sigma_n^{-2} A^{-1} X \mathbf{y} \tag{2.14}$$

$$A = \sigma_n^{-2} X X^\top + \Sigma_p^{-1} \tag{2.15}$$

Once the predictive mean over the parameters $\mathbf{w}$ is given, the mean over the objective is also known, via Eq. (2.10).

The kernel function

The assumption of a linear relationship between input variables and observables assumed in the case of linear Bayesian regression is very restrictive, and generally results in a poor approximation of the objective. Instead it is more convenient to assume the more general relationship:

$$f(\mathbf{x}) = \phi(\mathbf{x})^\top \mathbf{w}, \tag{2.16}$$

where $\phi :\to \phi(\mathbf{x})$ is feature mapping function, which can be learnt during inference. In this case the mean and kernel function of the GP surrogate model become:

$$\mathbb{E}\left[f(\mathbf{x})\right] = \phi(\mathbf{x})^\top \mathbb{E}[\mathbf{w}] \tag{2.17}$$

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') = \mathbb{E}\left[f(\mathbf{x}), f(\mathbf{x}')\right] &= \phi(\mathbf{x})^\top \mathbb{E}[\mathbf{w}\mathbf{w}^\top] \phi(\mathbf{x}') \\ &= \phi(\mathbf{x})^\top \Sigma_p \phi(\mathbf{x}) = \psi(\mathbf{x})^\top \psi(\mathbf{x}') \end{aligned} \tag{2.18}$$

Eq. (2.18) shows that the covariance depends on the input variables only via the dot product of $\phi(\mathbf{x})$, and it can be expressed as:

$$k(\mathbf{x}, \mathbf{x}') = \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle. \tag{2.19}$$

In other words the *kernel trick* has been applied as a direct consequence of the transformation (2.16). The kernel trick is a method that allows us to perform data linearisation without explicit feature mapping. This is particularly advantageous when the number of dimension is high, as feature mapping becomes computationally expensive.

As the dot product between two vectors measures their similarity [14], by virtue of Eq. (2.19) the kernel function also quantifies the similarity between $\psi(\mathbf{x})$ and $\psi(\mathbf{x}')$, i.e between the transformations of the original input points. Because of Eq. (2.16),

the same dot product is also a similarity measure between $f(\mathbf{x})$ and $f(\mathbf{x}')$. Two dissimilar objects have a small dot product, while the more similar these objects are, the higher is the dot product. If the function ϕ is normalized the maximum value of $\langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle$ is 1, corresponding to identical objects.

There are several possible choices for the specific analytic form of the kernel function, which encodes the initial beliefs about the objective function, for example its smoothness. However not all functions are valid kernels, as they must satisfy the condition of being positive semi-definite. A few examples of kernel functions are given in Section 2.3. Each type of kernel is characterised by a vector of parameters, which we will denote by $\boldsymbol{\theta}$, and which strongly impacts the behaviour of the GP model, and its predictive accuracy.

Predictions with Gaussian processes

A very important aspect of Gaussian processes, especially when applied to optimisation problems, is their capability of predicting the values of the objective function at unobserved points. This is true for all machine learning models, but the particularity of GPs is that they provide also the uncertainties over such predictions. Here we assume again that the observations are noisy: $y = f(\mathbf{x}) + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$.

Let us imagine we have a set of training inputs $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and the corresponding training outputs $\mathbf{y} = \{y_1, \ldots, y_N\}$, and we want to predict the value of the objective f at a set of test points $X_{\text{test}} = \{\mathbf{x}_{test,1}, \ldots, \mathbf{x}_{test,M}\}$. In the framework of Gaussian processes this can be done in a probabilistic way, by computing the joint prior probability distribution of the training outputs, $\mathbf{y}$, and the test outputs $\mathbf{y}_{\text{test}}$, and then conditioning this probability over the observations. Assuming a constant prior mean, $\boldsymbol{\mu}_0 = 0$, the joint prior probability distribution for noisy observations is given by[2]:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{y}_{\text{test}} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) + \sigma_n^2 I & K(X, X_{\text{test}}) \\ K(X_{\text{test}}, X) & K(X_{\text{test}}, X_{\text{test}}) \end{bmatrix}\right). \tag{2.20}$$

Here $K(\cdot,\cdot)$ is a Gram matrix, i.e a Hermitian matrix of inner products, for which each element is given by $K_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j) = \langle \psi(\mathbf{x}_i), \psi(\mathbf{x}_j) \rangle$. The probability over $\mathbf{y}_{\text{test}}$ is given by conditioning the joint prior over the training points, which results in another multivariate normal distribution:

$$\mathbf{y}_{\text{test}} | X, \mathbf{y}, X_{\text{test}} \sim \mathcal{N}\left(\bar{\mathbf{y}}_{\text{test}}, \text{cov}(\mathbf{y}_{\text{test}})\right), \tag{2.21}$$

[2]Here $\mathbf{y}$ is an array of variables and not a function, thus we have a normal distribution and the notaion $\mathcal{N}(\cdot)$ is used.

where

$$\bar{\mathbf{y}}_{\text{test}} = K(X_{\text{test}}, X)\left[K(X,X) + \sigma_n^2 I\right]^{-1}\mathbf{y} \tag{2.22}$$

$$\text{cov}(\mathbf{y}_{\text{test}}) = K(X_{\text{test}}, X_{\text{test}}) - K(X_{\text{test}}, X)\left[K(X,X) + \sigma_n^2 I\right]^{-1}K(X, X_{\text{test}}). \tag{2.23}$$

The predictions at the test points are given by $\bar{\mathbf{y}}_{\text{test}}$, which represents the most probable values of f at these points, while $\text{cov}(\mathbf{y}_{\text{test}})$ quantifies the uncertainty on these predictions. Eq. (2.22) and (2.23) show that the quality of the predictions depend uniquely on the covariance matrix $K(\cdot,\cdot)$. Thus the parameters characterising K must be optimised over the available data for the predictions to be correct. This process is described in Section 2.3.1 below.

2.3 Examples of kernel functions

Two very commonly used kernel functions are the *Radial Basis Function* (RBF) and the *Matern* kernels.

The RBF kernel. The radial basis function kernel (RBF), also called *squared exponential kernel*, is given by:

$$k_{\text{RBF}}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-\frac{1}{2\ell^2}(\mathbf{x}_i - \mathbf{x}_j)^2\right) + \sigma_n^2\delta_{ij}. \tag{2.24}$$

This kernel is meant to model noisy observations, assuming a Gaussian distribution with variance σ_n for the noise. The parameters σ_f and ℓ are respectively the signal variance, and the characteristic *length-scale*. The first one models the amplitude of the noiseless objective f, the second is a measure of spatial correlation between input variables.

The Matérn kernel. The Matérn kernel is a generalization of the RBF kernel, and it is given by:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{\Gamma(\nu)2^{\nu-1}}\left(\frac{\sqrt{2\nu}}{l}d(\mathbf{x}_i, \mathbf{x}_j)\right)^{\nu} K_\nu\left(\frac{\sqrt{2\nu}}{l}d(\mathbf{x}_i, \mathbf{x}_j)\right), \tag{2.25}$$

where Γ is the gamma function, K_ν is the modified Bessel function of the second kind, ρ and ν are positive parameters of the Matérn covariance and $d(\mathbf{x}_i, \mathbf{x}_j)$ is the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$, typically an Euclidean distance:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{d=1}^{D}\left(\mathbf{x}_{i,d} - \mathbf{x}_{j,d}\right)^2}, \tag{2.26}$$

where D is the number of dimensions of the space.

Common values for the ν parameter are 3/2 and 5/2. These examples show that typically kernel functions are parametric, and their parameters ultimately are the parameters of the GP. The GP has also an additional parameter, σ_n, which is the variance of the noise ϵ over the observations.

2.3.1 Model selection

To be able to use the surrogate model $\mathcal{M}$ as an accurate approximation of the latent objective function, we need to optimize all its parameters, based on our previous observations. Generally the hyper-parameters of $\mathcal{M}$ are organised hierarchically: at the lowest level are the parameters of the objective function f, if present. For example this is the case when BO is employed to optimise the hyper-parameters of a machine learning model, like the hyper-parameters of a neural network, which can be represented by a vector $\mathbf{w}$. The objective function in this case is the loss function of the neural network, and it is thus parameterized by $\mathbf{w}$.

At the second level there are the parameters which control the probability distributions. These are the hyper-parameters of the kernel, $\boldsymbol{\theta}$. Sometimes the surrogate model consists of a set of structures, for example a mixture of GPs or an ensemble model. In this case an additional level of parameters is present, which parametrize the whole model. These latter parameters are at the top of the hierarchy. If all these three levels of parameters are present, the inference occurs one level at the time, starting from the lowest. For each level the following steps take place:

1. Computing the posterior probability according to Bayes' rule as shown in Eq. (2.11).
2. Marginalizing the posterior probability with respect to the parameter at the current level of the hierarchy. This means integrating the posterior probability over all the possible values of this parameter, as shown in Eq. (2.12). The result of this operation is the marginal likelihood, which represents how probable the observed data is, given our surrogate model and taking into account all possible values of the parameter.

As one level of parameters is optimised, the marginal likelihood computed at step 2 represents the likelihood for the next level of parameters. Thus we compute the posterior probability as indicated at step 1, over the remaining parameters, and using the marginal likelihood calculated at the previous step as current likelihood.

Steps 1 and 2 are repeated for all the levels of parameters, from lowest to highest, until we find the overall marginal likelihood:

$$P(\mathbf{y}|X) = \sum_i p(\mathbf{y}|X, H_i)p(H_i), \tag{2.27}$$

where H_i is the set of parameters at the top of the hierarchy. The other parameters are contained in Eq. (2.27) implicitly. Afterwards, a further step is necessary, consisting in the maximisation of $P(\mathbf{y}|X)$. This allows us to find the optimal values for all parameters, although this might be unfeasible in practice, and approximate calculation methods might be required.

In majority of cases however, only the hyper-parameters of the kernel need to be tuned. The whole process of model selection is then given by the three steps below:

1. Computation of posterior:

$$p(\boldsymbol{\theta}|X, \mathbf{y}) = \frac{p(\mathbf{y}|X, \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathbf{y}|X)}. \tag{2.28}$$

2. Computation of the marginal likelihood:

$$p(\mathbf{y}|X) = \int p(\mathbf{y}|X, \boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}. \tag{2.29}$$

3. Maximization of the logarithm of the marginal likelihood[3]:

$$\boldsymbol{\theta} = \arg\max \log\, p(\mathbf{y}|X, \boldsymbol{\theta}) \tag{2.30}$$

$$\log\, p(\mathbf{y}|X, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^\top K^{-1}\mathbf{y} - \frac{1}{2}\log|K| - \frac{N}{2}\log(2\pi). \tag{2.31}$$

The maximization of $p(\mathbf{y}|X, \boldsymbol{\theta})$ in step 3 is typically performed using a gradient based optimizer. A full explanation of regression with Gaussian processes can be found in [115], Chapter 5.

2.3.2 Acquisition functions

An acquisition function gives a heuristic that specifies how desirable it is to evaluate the objective f at any given point, based on the information provided by the surrogate model. In sequential BO the next point to evaluate is returned by maximizing the acquisition function:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathcal{X}} a(\mathbf{x}). \tag{2.32}$$

[3] $\frac{N}{2}\log(2\pi)$ is a normalisation term, where N is the number of data points.

Generally, to make the optimisation process efficient, acquisition functions enforce an automatic trade off between exploitation, which focuses the search for the optimum in vicinity of the current best point, and exploration, which aims the search towards unexplored regions of the space. Commonly used acquisition functions are the *expected improvement* [51], the *upper confidence bound* [5], the *knowledge gradient* [13], and the *entropy search* [40]. All experimental results reported in this thesis have been obtained using expected improvement (EI), with the exception of the multi-fidelity experiments reported in Chapter 3, for which entropy search has been used. A brief description of these two specific acquisition functions is given below.

Expected improvement. Let us suppose that we have already acquired some data during the optimisation process, and that $f_n^{\text{best}} = f(\mathbf{x}_n^{\text{best}})$ is the best value observed so far at the step n. For minimization problems f_n^{best} is the lowest observed output: $f_n^{\text{best}} = min_{i<n} f(\mathbf{x}_i)$. It is possible to define the *improvement* I_n as:

$$I_n = \max_{\mathbf{x}\in\mathcal{X}}(f(\mathbf{x}_n^{\text{best}}) - f(\mathbf{x}), 0), \tag{2.33}$$

which is a non-negative quantity. The expected value of the improvement at a given point before it is actually acquired is thus given by:

$$\mathbb{E}[I_n(\mathbf{x})] = \mathbb{E}[\max(f(\mathbf{x}_n^{\text{best}}) - f(\mathbf{x}), 0)]. \tag{2.34}$$

Intuitively, Eq. (2.34) means that EI estimates how much closer we get to the global optimum of the objective function by evaluating f at a given input variable x. Obviously, it is desirable to select the point which maximises the expected improvement, and fortunately this acquisition function can be expressed by a closed form, as a function of the posterior mean $\mu(\mathbf{x})$ and the posterior variance $\sigma(\mathbf{x})$ of the GP [2]:

$$\text{EI}(\mathbf{x}) = \begin{cases} (\mu(\mathbf{x}) - f(\mathbf{x}^+) - \xi)\Phi(Z) + \sigma(\mathbf{x})\phi(Z) & \text{if } \sigma(\mathbf{x}) > 0 \\ 0 & \text{if } \sigma(\mathbf{x}) = 0 \end{cases}, \tag{2.35}$$

where:

$$Z = \begin{cases} \frac{\mu(\mathbf{x}) - f(\mathbf{x}^+) - \xi}{\sigma(\mathbf{x})} & \text{if } \sigma(\mathbf{x}) > 0 \\ 0 & \text{if } \sigma(\mathbf{x}) = 0 \end{cases}. \tag{2.36}$$

In the formulas above, ϕ and Φ are the probability density function and the cumulative density function of a standard normal distribution, respectively. The parameter ξ, also called jitter, tunes the balance between exploitation and exploration, and it is typically set to 0.01.

Entropy search. Entropy search is an information theory based acquisition function which aims to minimize the uncertainty about the location of the optimum point, $\mathbf{x}_{\text{opt}}$. In information theory the uncertainty over a random variable $\mathbf{x}$ with probability distribution $p(\mathbf{x})$ is quantified by the *negative differential entropy* of $p(\mathbf{x})$. The entropy H of $p(\mathbf{x})$ is defined as [42]:

$$H[p(\mathbf{x})] = -\int p(\mathbf{x})\log\, p(\mathbf{x})d\mathbf{x}. \tag{2.37}$$

When a new point $\mathbf{x}$ is acquired, the posterior distribution over the optimum point varies from $p(\mathbf{x}_{\text{opt}}|\mathcal{D}_n)$ to $p(\mathbf{x}_{\text{opt}}|\mathcal{D}_n \cup (\mathbf{x}, y)$. Ideally, at a given BO step, the best point to query next is the one which leads to the highest decrease in entropy, thus to the highest reduction in uncertainty. The expected value of such reduction can be estimated as:

$$\text{ES}_n = H\left(p(\mathbf{x}_{\text{opt}}|\mathcal{D}_n)\right) - \mathbb{E}\left[H\left(p(\mathbf{x}_{\text{opt}}|\mathcal{D}_n \cup (\mathbf{x}, y)\right)\right], \tag{2.38}$$

where ES_n is the negative differential entropy, and $\mathbb{E}\left[H\left(p(\mathbf{x}_{\text{opt}}|\mathcal{D}_n \cup (\mathbf{x}, y)\right)\right]$ is the estimated value the entropy would assume if the point $\mathbf{x}$ is acquired. The maximisation of ES_n is analytically intractable, but it can be performed via approximation, as described in the original paper by Hennig and Schuler [40].

2.4 Batched Bayesian Optimization

As mentioned before, the aim of Bayesian optimisation is to minimize the number of evaluations of the objective f. However this number can still be considerable for some optimisation problems, for example if the search space is high dimensional or large. This problem can be addressed by parallelizing the evaluations of f. In this case, the data set is augmented by multiple points at each optimisation step, instead of a single one.

The objective is then evaluated at all the new points simultaneously in parallel. The challenge of this approach is to establish a policy to select a batch of k input variables at which f will be evaluated next $\text{B}_n^k = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$, where k is the batch size, and n is the current optimisation step. For this purpose it is necessary to define an acquisition function a_{batch}, analogously to sequential BO. Each point $\mathbf{x}_{n,b} \in \text{B}_n^k$ is chosen so as to maximize a_{batch}, based on the surrogate model and the current data set, which includes the previously selected batch points at the same optimisation step n:

$$\mathbf{x}_{n,b}^* = \underset{\{\mathbf{x}_b\}_{b=1,\ldots,k} \in \mathcal{X}}{\arg\max}\; a_{\text{batch}}(\mathbf{x}). \tag{2.39}$$

Solving (2.39) presents two issues: the first one is that no observations are available at any of the points $\mathbf{x}_1, \dots, \mathbf{x}_k$. This implies that the uncertainties about the inputs $\mathbf{x}_{n,b}$, as well as about the outputs $y_{n,b} = f(\mathbf{x}_{n,b})$, must be taken into account by marginalizing over both input and output variables, using the respective predictive distributions. Formally [37]:

$$\mathbf{x}_{n,b} = \arg\max_{\mathbf{x}\in\mathcal{X}} \int a_{batch}(\mathbf{x}|\mathcal{D}_{n,j-1}) \tag{2.40}$$

$$\prod_{j=1}^{k-1} p(y_{n,j}|\mathbf{x}_{n,j}, \mathcal{D}_{n,j-1}) p(\mathbf{x}_{n,j}|\mathbf{x}_{n,j}, \mathcal{D}_{n,j-1}) d\mathbf{x}_{n,j} dy_{n,j} \tag{2.41}$$

where $\mathcal{D}_{n,j-1}$ is the available data set at step n, when the j^{th} point of the batch is selected. However this optimisation is practically intractable, and some heuristic batching policies have been proposed in literature instead. The second issue is the risk of redundancy, i.e. the sampling of batch points which give similar outputs, so that little more information is gained by acquiring an entire batch instead of a single point. On the other hand, driving the sampling by diversity only can be detrimental to the informativeness of the points about the location of the actual optimum. Thus a trade off between diversity and informativeness is necessary. Recent methods proposed in literature to satisfy this requirement include *local penalization* [37], *Parallel Predictive Entropy Search (PPES)* [94], *parallel knowledge gradient* [116], but this matter is still an active topic of research.

Local penalization. In order to avoid redundancy when sampling, González et al. [37] proposed a BBO method called *local penalization* (LP). The core idea of LP is to penalize points falling in a spherical neighbourhood of previously selected points. The radius of this sphere is determined on the basis of the estimated Lipschitz constant of the acquisition function.

Parallel Predictive Entropy Search (PPES). The PPES method, proposed by Shah and Ghahramani [94] focuses on the informativeness of the batched points. It relies on a batching policy that aims to maximize the *negative differential entropy* of the posterior distribution. From a conceptual point of view this policy corresponds to maximizing the information about the location of the global optimum.

Parallel knowledge gradient. Another BBO method based on information theory is the *parallel knowledge gradient* method [116], proposed by Frazier and coauthors, which chooses the next point by maximizing the expected difference between the current optimum value, estimated from the surrogate model, and the estimated optimum value after the new points are acquired.

2.5 Bayesian optimisation in discrete spaces

This section presents an overview of BO methods which have been proposed in literature to deal with discrete spaces. Examples of optimisation problems over discrete spaces are problems over strings [74, 102] or combinatorial problems [80], or problems defined over categorical variables [33]. Optimisation over strings is particularly relevant to materials design, as molecules are often represented as *SMILES* stings [114].

Bayesian optimisation over discrete variables is challenging for multiple reasons: in first place the similarity kernels commonly used for BO and introduced in Chapter 2 are defined for continuous variables, and they are not guaranteed to be suitable for discrete variables, as the condition of being semi-positive definite might be violated. Secondly the evaluation of the next point or batch of points at which to evaluate the objective function f is not straightforward. Indeed this step requires the maximisation of the acquisition function a, as expressed by Eq. (2.42):

$$\mathbf{x}_n = \arg\max_{\mathbf{x}\in\mathcal{X}} a(\mathbf{x}). \tag{2.42}$$

For continuous search spaces this maximization problem can be solved by gradient based methods, which in principle are not applicable to discrete spaces. A common solution to this issue is to consider the search space to be continuous, to apply a gradient based method to solve Eq. (2.42), and then to evaluate the objective function at the closest input location which is actually present in the discrete optimisation domain. However this approach raises two problems: first there is a discrepancy between the point suggested by the acquisition function and the point which is actually added to the model's data set $\mathcal{D}$. This can cause inaccuracy in the optimisation process. Furthermore, the rounding process can induce to evaluate the objective always at the same input point [33]. Several strategies have been proposed in literature to address the aforementioned problems. Among these, the most related to our research goals are the *amortized Bayesian optimisation* [102], the *BOSS* method [74], *combinatorial Bayesian optimisation with graph representations* [80]. A brief description of these methods is provided below.

Amortized Bayesian optimisation. The amortized Bayesian optimisation method [102], proposed by Swersky and co-authors, focuses on optimisation over strings, using a novel optimizer for the acquisition function. The core feature of amortized Bayesian optimisation is the implementation of the *Deep evolutionary solver algorithm* (DES) to maximize the acquisition function, which combines an evolutionary algorithm with reinforcement learning. The evolutionary algorithm mutates a population of strings according to a policy π_θ. The mutated strings that maximize the acquisition function are used to update the surrogate model. The policy π_θ is a

network policy, and its parameters $\boldsymbol{\theta}$ are optimized online via REINFORCE. The main advantage of this approach is that the the solver uses the parameters $\boldsymbol{\theta}$ learnt from previous iterations in order to progressively approach optimal locations for the acquisition function instead of solving Eq. (2.42) from scratch at each optimisation step. In addition of being suitable for discrete spaces, this BO approach is also suitable for batch optimisation.

Bayesian optimisation over String Spaces. Analogously to amortized BO, the BOSS method [74] focuses on BO over string spaces, and maximizes the acquisition function using a genetic algorithm. Additionally, a customized GP for string spaces is used, which is achieved in two steps: first the authors introduce a kernel specifically designed for string variables, then they use a vectorized form of dynamic programming, previously proposed by Beck and Cohn [10], in order to reduce the computational cost of the kernel evaluations.

Although the genetic algorithm usually requires numerous function evaluations, and thus it is expensive for optimizing a high-cost objective function, in this work it is used to optimize the acquisition function, which is designed to be much less costly. For this work in particular the authors chose expected improvement as the acquisition function.

Combinatorial Bayesian Optimization using Graph Representations. The Combinatorial Bayesian Optimization method using Graph Representations or COMBO [80] is formulated to solve optimisation problems on combinations. The main goal of this work is the design of an appropriate kernel for this class of problems. For this purpose the search space is represented as a graph, where a vertex represents a given combination, and an edge between two vertexes determines whether the combinations it connects are similar or not. Then a diffusion kernel on graphs is used [59]. In order to achieve a linear dependence of the kernel computation cost with respect to the number of input points, the original graph is decomposed in the Cartesian product of smaller sub-graphs. The acquisition function is optimized via a greedy optimisation on graphs.

Chapter 3

Methods for Integrating Predicted Data into Bayesian Optimization

3.1 Introduction

One reason why Bayesian optimisation can require numerous observations is that the surrogate model provides a poor approximation of the actual objective function. This is especially true if no assumptions are made a priori, and the training of the surrogate model relies only on the available starting data set.

It is often the case however, that prior knowledge related to the optimisation problem at hand is available. This is particularly true for the design of experiments in the field of physical sciences, where prior knowledge is typically available in the form of theoretical models, numerical simulations or previous experimental results. The resulting information can be used to produce predictions of the expected values of the objective f, and to generate a more accurate surrogate model. However these predictions can be fairly coarse, with no formal guarantee on their accuracy, and they may give completely incorrect answers for some portions of the domain.

The work reported in this chapter addresses this problem by formulating strategies to balance the informativeness of the prior knowledge, while limiting the impact of its inaccuracy.

Overview of the methods. The first method that will be introduced, called *exclusion radius method*, adds predicted data at the start of the optimisation process, and then iteratively deletes it as real data is obtained.

The second method, called *discrepancy prediction method*, also adds predicted data at the start of the process, and then as more information about f is obtained, it attempts to correct the errors in the predicted data by learning a model of the

difference between actual and predicted outputs. These methods have been tested on standard benchmarks for Bayesian optimisation, using synthetic predictors with different levels of accuracy. For this purpose predictors at a specified accuracy level were created for each given benchmark function. In Section 3.5 the predictors' accuracy is formally defined, and the method to build deterministic smooth predictors for a given function and a given accuracy level is described.

3.2 Problem statement

The work reported in this chapter aims to solve the following optimisation problem:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}), \tag{3.1}$$

where $f : \mathcal{X} \to \mathbb{R}$ is the objective function of interest.

The main working assumption is that f is expensive to evaluate, but a predictive model $p : \mathcal{X} \to \mathbb{R}$ is available, which returns an approximate value of f at each point in the domain, and which is assumed to be significantly cheaper to evaluate than the actual objective. The quality of the predictor at any given point can be quantified by the *discrepancy*, $\delta : \mathcal{X} \to \mathbb{R}$, defined as:

$$\delta(\mathbf{x}) = f(\mathbf{x}) - p(\mathbf{x}), \tag{3.2}$$

where $\delta(\mathbf{x})$ represents the error coming from using p to evaluate f instead of computing f itself. However $\delta(\mathbf{x})$ is not known a priori, and potentially the predictor provides low quality predictions. The working hypotheses for the problem we aim to solve can be summarized as follows:

1. The objective function f is much more expensive to evaluate than the predictor p, which can be evaluated essentially for free.
2. The predictor is deterministic, so querying the same point multiple times yields no extra information.
3. The predictor is a smooth function.
4. The predictor may also present systemic errors on the estimation of f.
5. The estimation error of the predictors, $\delta(\mathbf{x})$, is unknown.

The aim is to initialize the BO processes by initializing the GP surrogate with a given number of predicted points, and then discard them or reduce their impact on the predictive distribution during the optimization process.

The methods we propose do not need any formal assumption on the nature of the predictor nor on the process by which it is generated: it may be implemented using

a theoretical model, through computational simulations, or it could be a machine learning model.

Points 3 and 4 imply that, if the predictor makes a poor quality prediction for a particular point x, then it may well also make poor quality predictions for the points surrounding x. So, if one is in a region where the predictor gives poor predictions, then there may be no easy way to extract information about f, and so the main challenge is to discard these poor quality predictions while making use of good quality predictions when they arise

3.3 Related work

The sequential BO approaches that have been proposed in literature that are closest to the exclusion radius and the discrepancy prediction methods are the *warm starting* and *multi-fidelity* methods. The first one exploits the information gained from optimisation tasks which have already been solved on previous data sets, while the second relies on approximate models of the objective function to generate additional observations at a lower cost than querying the actual objective.

Warm starting

The warm starting approach assumes that the optimisation task under study has already been solved on data sets which are somehow *similar* to the current one. The resulting information can be transferred to the current task by initializing the BO process with samples from the previous tasks [88, 101].

The most straightforward way to achieve this, which is also the closest to our proposed methods, is the approach proposed by et. al. [88], who use samples of previous tasks to initialize the surrogate model at the beginning of the optimisation process. Afterwards, the difference between the present objective function and the objective of a previous task is modelled with the quantity δ_ℓ, where ℓ indicates a specific previous task. It is assumed that, for each task ℓ, δ_ℓ is drawn from an independent Gaussian process with mean function $\boldsymbol{\mu}_\ell$ and covariance kernel Σ_ℓ.

The posterior distribution over the current objective is modelled as a joint Gaussian process, which is built starting from these individual surrogate models, in the same way as it is done in the multi-fidelity paradigm, which is described in the following section. Poloczek and and co-authors also use hyper-parameters for the kernel that were optimal for the previous tasks to formulate the prior distribution over the current objective.

Multi-fidelity optimisation

Multi-fidelity optimisation methods have been developed to deal with scenarios where the optimisation process has access to lower fidelity models which approximate the actual objective function, and which can be evaluated at reduced cost [29, 47, 52, 57, 86, 87, 98].

In this setting it is assumed that these models are hierarchically ordered by their fidelity with respect to the actual objective, such that as one moves up the hierarchy the cost of evaluating the models decreases, but with the drawback of obtaining lower fidelity information about the function that is being optimized. The relationship between the actual objective function and the lower fidelity models, also called *information sources*, is expressed as [34, 41, 85, 87, 98]:

$$f(\ell, \mathbf{x}) = \rho f(\ell - 1, \mathbf{x}) + \delta_\ell(\mathbf{x}), \tag{3.3}$$

where $\ell = \{1, \ldots, n\}$ indicates the level of fidelity, ρ is a scaling constant that quantifies the correlation between the model outputs $f(\ell, \mathbf{x})$ and $f(\ell-1, \mathbf{x})$. Typically $\ell = 0$ refers to the actual objective function. The *discrepancy* $\delta_\ell(\mathbf{x})$ is the difference between the approximate model $f(\ell, \mathbf{x})$ and the actual objective $f(0, \mathbf{x})$:

$$\delta_\ell(\mathbf{x}) = f(\ell, \mathbf{x}) - f(\mathbf{x}). \tag{3.4}$$

The purpose of using lower fidelity models is to reach the optimal trade-off between cost and accuracy. For this purpose, at each step, the fidelity level ℓ of the model to sample and the next input point where to evaluate it are selected simultaneously to minimise the overall cost of the optimisation process. As a result, the original optimisation problem, which was defined over the input variables $\mathbf{x} \in \mathcal{X}$ is now defined over $\mathbf{x}$ and ℓ. To solve it, a suitable GP surrogate model and a suitable acquisition function need to be defined.

Construction of prior. In order to build a prior probability distribution for multi-fidelity optimisation problems $\delta_\ell(\mathbf{x})$ is assumed to be drawn from an independent Gaussian process:

$$\delta_\ell(\mathbf{x}) \sim GP\left(\boldsymbol{\mu}_\ell, \Sigma_\ell\right). \tag{3.5}$$

On the other hand, the objective $f(0, \mathbf{x})$ is drawn from another independent GP: $f \sim GP(\boldsymbol{\mu}_0, \Sigma_0)$. From all these independent Gaussian processes it is possible to build a new joint GP [29, 88]:

$$GP\left(\boldsymbol{\mu}(\ell, \mathbf{x}), \Sigma((\ell, \mathbf{x}), (m, \mathbf{x}'))\right), \tag{3.6}$$

where:

$$\boldsymbol{\mu}(\ell, \mathbf{x}) = \boldsymbol{\mu}(0, \mathbf{x}) \tag{3.7}$$

$$\Sigma((\ell, \mathbf{x}), (m, \mathbf{x}')) = \Sigma_0(\mathbf{x}, \mathbf{x}') + I_{\ell,m}\Sigma_0(\mathbf{x}, \mathbf{x}'). \tag{3.8}$$

The probability distribution defined by Eq. (3.6) models the information source $f(\ell, \mathbf{x})$ for all ℓ simultaneously.

Acquisition function. In order to choose the optimal fidelity level ℓ^* and the optimal point $\mathbf{x}^*$ it is necessary to use a specifically designed acquisition function which take the specific cost of each model into account [87, 117]. For example Marco et al. [70] and Swersky et al. [101] proposed a cost-sensitive version of the information gain acquisition function which has been shown to be more efficient for multi-fidelity optimisation compared to expected improvement.

Multi-fidelity BO workflow. The workflow of a multi-fidelity Bayesian optimisation, which is also represented in Figure 3.1, can be summarized as follows:

1. Initialization of GP $(\boldsymbol{\mu}_0, \Sigma_0)$ with samples from all the information sources.
2. At each optimisation step n:
   - find the values $(\ell_n, \mathbf{x}_n)$ which maximise the acquisition function.
   - evaluate the next observable at $y_n = f(\ell_n, \mathbf{x}_n)$.
3. Return the optimum point $\mathbf{x}_{\text{opt}} = \arg\min_{x' \in \mathcal{X}} \boldsymbol{\mu}(0, \mathbf{x}'))$.

Comparison with our methods.

Similar to the warm-starting method, the exclusion radius method and the discrepancy prediction methods introduce prior knowledge into BO by initializing the GP surrogate with samples similar to the real data. However they present the advantage that no results from previous optimisation tasks are required, since the initializing data is generated by an inexpensive predictor.

On the other hand the problem settings for the two methods we propose are also close to the multi-fidelity scenario, as the predictor is in fact a lower fidelity model.

However, whereas existing work in multi-fidelity optimisation balances the costs of obtaining predicted data as opposed to obtaining real data [64, 87], for the predictor p and for the objective function f the cost is not specified, as the first one is assumed to be essentially cost-free, while the second is extremely expensive, so essentially infinitely more expensive than the predictor.

The rationale behind these assumptions is that our methods are designed for applications to experimental work in natural sciences, where the cost of the objective is typically very large but not known a priori. Furthermore in this scenario it is not possible to make formal assumptions about the quality of the data produced by p. Indeed, while a scientist may have a theory that predicts the outcome of an experiment, the only way to validate the accuracy of that theory would be to run experiments. But that would be self-defeating, as it is those expensive experiments that we would like to avoid in the first place.

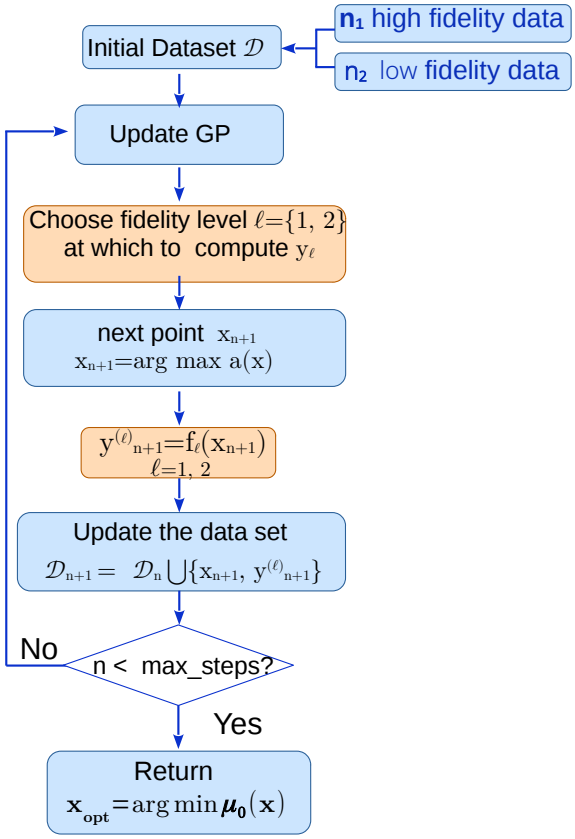

Figure 3.1: High-level workflow for multi-fidelity BO method. Steps which differ from standard BO are in orange. The level of accuracy to use at the current BO step t is selected by the acquisition function. The best point $\mathbf{x}_{\text{opt}}$ is evaluated as the optimum point of the predictive mean at the highest level of accuracy. $\boldsymbol{\mu}_1$

3.4 Our Methods

This section describes in detail the exclusion radius method and the discrepancy prediction method.

3.4.1 The exclusion radius method

The core idea of the *exclusion radius method*, described in Algorithm 2, is to sample a large number of points from the predictor and use these to initialize Bayesian optimisation. Then Bayesian optimisation is performed as normal, but in each iteration, when $f(\mathbf{x})$ is sampled at a point $x \in \mathcal{X}$, all predicted points that are within a given radius from x are removed from the model. Formally, in each iteration i, the method maintains two sets of data. The set $\mathcal{R}_i$ denotes the set of *real* data, and it contains pairs $(\mathbf{x}, f(\mathbf{x}))$. The set $\mathcal{P}_i$ denotes the set of *predicted* data, and it contains pairs $(\mathbf{x}, p(\mathbf{x}))$. At the start of the process $\mathcal{R}_0$ is empty, since no queries to f have been made, so the surrogate model is initialized with a set of initial predicted points, $\mathcal{P}_0$. $\mathcal{P}_0$ is generated selecting random input points from the domain $\mathcal{X}$ and evaluating the predictor at these points. In each step, Bayesian optimisation proposes a new point $\mathbf{x}_i \in \mathcal{X}$ to be sampled. The new data $(\mathbf{x}_i, f(\mathbf{x}_i))$ is added to $\mathcal{R}_i$, and then all predicted points that are close to $\mathbf{x}_i$ are deleted from $\mathcal{P}_i$. Specifically, all predicted points within distance r from $\mathbf{x}_i$ are deleted from $\mathcal{P}_i$. Thus, the set of points to delete at step iteration $i+1$ is given by: $\mathcal{B}_{i+1} = \{(\mathbf{x}, y) \in \mathcal{P}_i : \|\mathbf{x} - \mathbf{x}_i\|_2 \leq r\}$. The radius r is a parameter of the algorithm.

3.4.2 Discrepancy prediction method

The *discrepancy prediction method*, shown in Algorithm 3, aims to reduce the discrepancy of a point, as defined in Equation (3.2), by estimating its value, and correcting the predicted points by this estimated quantity. The estimation of $\delta(\mathbf{x})$ is maintained during the entire optimisation process.

For this purpose δ is modelled by a Gaussian process which is trained during the course of the Bayesian optimisation. Hence, this method uses two Gaussian processes: the model $\mathcal{M}$ that is used as part of Bayesian optimisation, and a model $\mathcal{M}_\delta$ that is used to predict the discrepancy.

As Bayesian optimisation proceeds, a set $\mathcal{C}_i$ containing data on the discrepancy of all points that we have sampled from f is maintained. That is, every time $f(\mathbf{x})$ is sampled, $(\mathbf{x}, f(\mathbf{x}) - p(\mathbf{x}))$ is added to $\mathcal{C}_i$. Then the surrogate model of the discrepancy, $\mathcal{M}_\delta$, is trained using $\mathcal{C}_i$. We define $\hat{\delta} : \mathcal{D} \to \mathbb{R}$ to be the expected value $\delta(\mathbf{x})$ of each point $x \in \mathcal{D}$ as predicted by $\mathcal{M}_\delta$.

Algorithm 2: The exclusion radius method

Input: The initial exclusion radius r, a predictor p, and a number of real observations N_{obs}

Initialize the starting sets of real and predicted points respectively as $\mathcal{R}_0 = \emptyset$, $\mathcal{P}_0 = \{(\mathbf{x}_j, p(\mathbf{x}_j)\}_{j=1,\cdots,N}$ and set the initial data set $\mathcal{D}_0 = \mathcal{R}_0 \cup \mathcal{P}_0$

for $i \leftarrow 0, 1, ...$ **do**

  Update the surrogate model $\mathcal{M}$ using the whole data set $\mathcal{D}_i$

  Select a new point $\mathbf{x}_{i+1}$ by optimizing the acquisition function a: $\mathbf{x}_{i+1} = \arg\max_{\mathbf{x}\in\mathcal{X}} a(\mathbf{x}, \mathcal{M})$

  Query f to obtain $y_{i+1} = f(\mathbf{x}_{i+1})$

  Create a new real data set $\mathcal{R}_{i+1} = \mathcal{R}_i \cup \{(\mathbf{x}_{i+1}, y_{i+1})\}$

  Find the predicted points in a ball around $\mathbf{x}_{i+1}$: $\mathcal{B}_{i+1} = \{(\mathbf{x}, y) \in \mathcal{P}_i : \|\mathbf{x} - \mathbf{x}_{i+1}\|_2 \leq r\}$ Exclude those points by setting $\mathcal{P}_{i+1} = \mathcal{P}_i \setminus \mathcal{B}_{i+1}$

  Set $\mathcal{D}_{i+1} = \mathcal{R}_{i+1} \cup \mathcal{P}_{i+1}$

  if *number real observations* $= N_{\text{obs}}$ **then**

    break

  end

end

Like the exclusion radius method, we split the data into real points $\mathcal{R}_i$, and predicted points $\mathcal{P}_i$, but no predicted points are deleted. Instead, in each iteration i the predicted points are updated using the currently estimation of the discrepancy $\hat{\delta}_i$. Formally this update is performed by creating a set $\mathcal{P}_i$ containing $(\mathbf{x}_p, p(\mathbf{x}_p) + \hat{\delta}(\mathbf{x}_p))$ for each predicted point $\mathbf{x}_p \in \mathcal{A}$. The values of the expected discrepancy $\hat{\delta}(\mathbf{x}_p)$ are inferred using the surrogate model $\mathcal{M}_\delta$.

The discrepancy prediction method is initialized using a mix of predicted points and real points (for the experiments a set of 45 predicted points and 5 real points are used). The small set of real points is used to train an initial discrepancy predictor, and then make an initial adjustment of the predicted data before the optimisation process begins. So formally, if k real points and l predicted points are used for initialization, then the real set is given by $\mathcal{R}_0 = \{(\mathbf{x}_i, f(\mathbf{x}_i)) \ : \ i = 1, 2, \ldots, k\}$ where each $\mathbf{x}_i$ is chosen uniformly from the domain $\mathcal{X}$, and the set of predicted points is given by $\mathcal{C}_0 = \{(\mathbf{x}_i, f(\mathbf{x}_i) - p(\mathbf{x}_i)) \ : \ i = 1, 2, \ldots, k\}$ for those same points. Then we train M_δ on $\mathcal{C}_0$, and we set $\mathcal{P}_0 = \{(\mathbf{x}_i, p(\mathbf{x}) + \hat{\delta}(\mathbf{x})) \ : \ i = 1, 2, \ldots, l\}$ as the initial set of predicted points.

Algorithm 3: The discrepancy prediction method

Input: predictor p, number of real observations N_{obs}

Initialize the $\mathcal{R}_0$, $\mathcal{P}_0$, and $\mathcal{C}_0$ as mentioned in the text, and set $\mathcal{S}_0 = \mathcal{R}_0 \cup \mathcal{P}_0$

for $i \leftarrow 0, 1, ...$ **do**

  Update the surrogate model $\mathcal{M}$ using the data in $\mathcal{S}_i$

  Update the surrogate model for the discrepancy $\mathcal{M}_\delta$ using the data in $\mathcal{C}_i$

  Select a new point $\mathbf{x}_{i+1}$ by optimizing the acquisition function a:

  $$\mathbf{x}_{i+1} = \arg\max_{\mathbf{x} \in \mathcal{D}} a(\mathbf{x}, \mathcal{M})$$

  Query f to obtain $y_{i+1} = f(\mathbf{x}_{i+1})$

  Create a new real data set $\mathcal{R}_{i+1} = \mathcal{R}_i \cup \{(\mathbf{x}_{i+1}, y_{i+1})\}$

  Set $\mathcal{C}_{i+1} = \mathcal{C}_i \cup ((\mathbf{x}_{i+1}, f(\mathbf{x}_{i+1}) - p(\mathbf{x}_{i+1})))$, and then retrain $\mathcal{M}_\delta$ on $\mathcal{C}_{i+1}$

  Compute the predicted discrepancy $\hat{\delta}(\mathbf{x}))$ using the updated model $\mathcal{M}_\delta$

  Create $\mathcal{P}_{i+1} = \{(\mathbf{x}, p(\mathbf{x}) + \hat{\delta}(\mathbf{x})) \ : \ \mathbf{x} \text{ is a point in } \mathcal{P}_0\}$

  Set $\mathcal{S}_{i+1} = \mathcal{R}_{i+1} \cup \mathcal{P}_{i+1}$

  if *number real observations* $= N_{\text{obs}}$ **then**

    break

  end

end

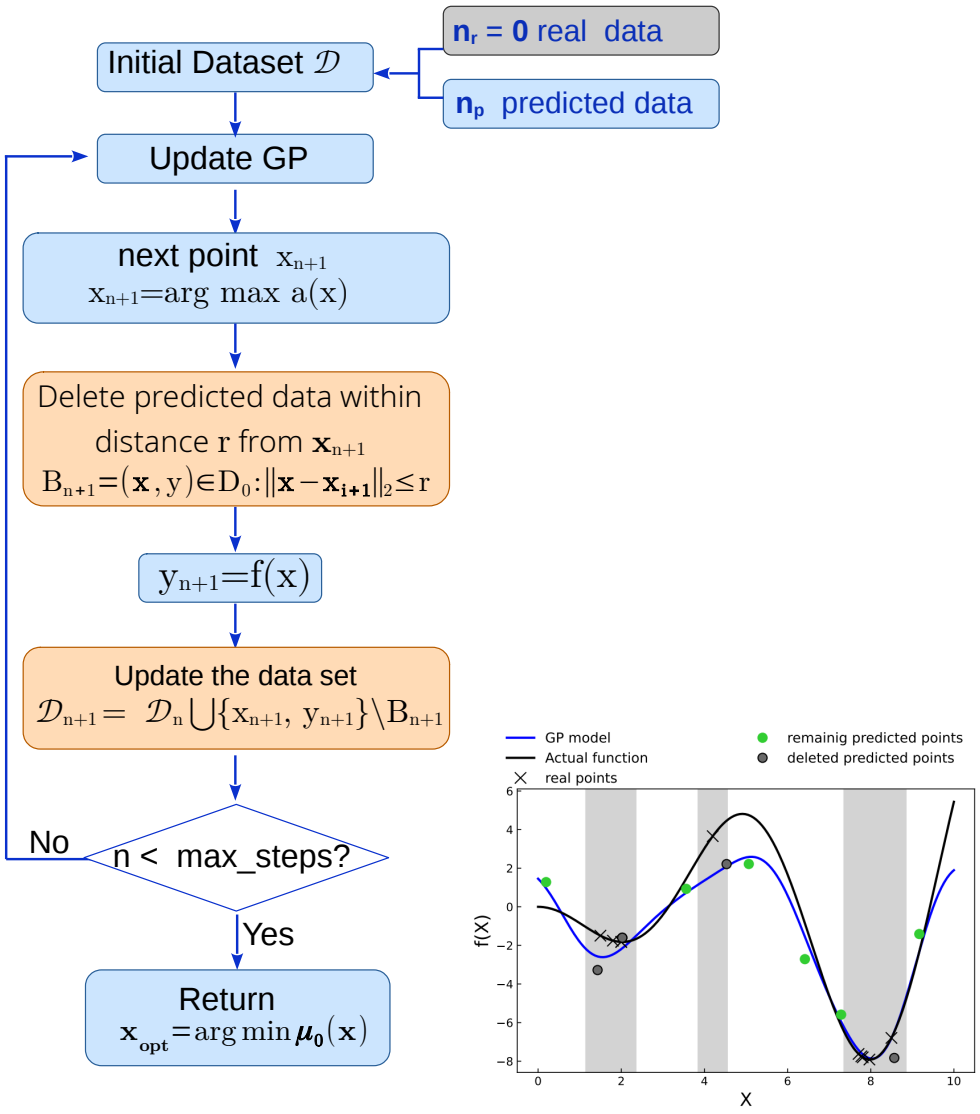

Figure 3.2: Left: workflow diagram of the exclusion radius method. Steps which differ from standard BO are in orange. Right: scheme of the exclusion radius method for a 1-dimensional problem. Real points are indicated with $\times$, and predicted points with a circle. Each time a new real point is acquired, a region centered in that point is computed according to point 7 of Algorithm 2. The predicted points which fall any of these regions are shown by grey circles: those points are removed from the data set of the surrogate model, $\mathcal{D}$, and will no longer be considered for the hyper-parameters optimization since the next BO step. The green circles indicate the predicted points which are outside of the aforementioned regions, and that will remain in $\mathcal{D}$.

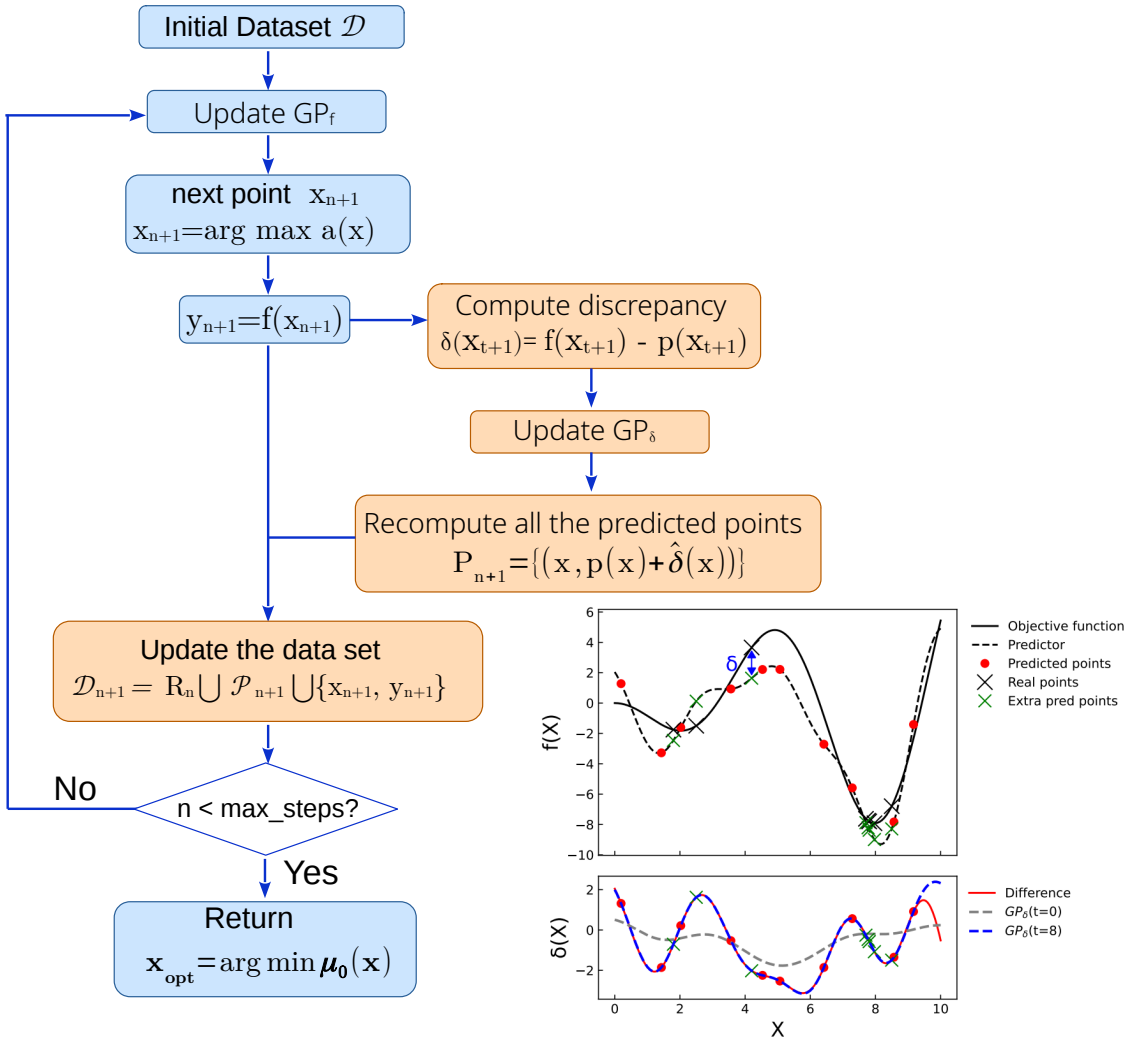

Figure 3.3: Top left: workflow diagram of the discrepancy prediction method. Steps which differ from standard BO are in orange. The graph on the right shows a 1D example of the discrepancy prediction method: in the top graph the actual objective function f and the predictor p are plotted respectively with a continuous and a dash black line, while the red points indicate the set of initial predicted points. The real points are indicated with a black $\times$. Each time a real point $(\mathbf{x}_t, y_t)$ is acquired, the predicted output at $\mathbf{x}_t$ is computed, and the point $(\mathbf{x}_t, f(\mathbf{x}_t) - p(\mathbf{x}_t))$ is added to the data of the surrogate model for the discrepancy, GP_δ, as shown at point 8 of Algorithm 3. These additional predicted points, plotted in the top graph with green $\times$, are used to train GP_δ. The bottom graph shows the actual discrepancy over the whole search space and (red continuous line), while the grey dashed line and the blue dashed line plot the discrepancy predicted by GP_δ at $t = 0$ and $t = 5$ respectively, showing significant improvement as more data becomes available.

3.5 Experimental Setup

This section is organised as follows: first a formal treatment of the predictors is introduced as they are vital to the study of our methods.

Indeed, even if the methods we propose are agnostic to the nature of the predictor and how it is generated, for our studies we needed to implement synthetic predictors, and to be able to tune their error with respect to the actual objective function, in order to assess the performance of each method against such error.

The building procedure for the predictors is an important part of our experimental set up, because, as explained in Section 3.5.1, some requirements need to be fulfilled in order to asses our methods correctly and to mimic realistic approximate models, which would be used for optimisation of experiment design: for example a low accuracy model would be continuous and smooth. It is important to notice that the procedure to build the predictor is not part of our BO methods, and the information about the predictor's accuracy is not used in the optimisation process. Thus the assumptions listed in Section 3.2 are still valid.

Following the description of the predictors, the overall experimental setup, and the setup for each of the methods will be provided.

3.5.1 The Predictors

The study presented here has been accomplished on five synthetic benchmarks functions: Ackley, Griewank, Michalewicz [50], Rastrigin [109] and Styblinski-Tang [100]. The analytical form of these functions as well as their global minima and the search domain over which they are optimized are summarized in Table 3.1.

While these benchmarks are standard, they do not come with any pre-defined predictor functions, so for the purpose of this study we build the predictors starting from the benchmark functions themselves. In doing so we also aim to tune the predictors' accuracy.

Creating the predictors. As mentioned in Section 3.2 the predictors must have the following properties:

- The predictor must be deterministic.
- The predictor must be smooth.

Furthermore, for benchmarking purposes, the discrepancy between the actual objective function and a predictor needs to be tunable, so to allow us to assess the robustness of a given optimisation method at different levels of accuracy.

Name	Formula	Minimum	Search domain
Ackley	$-20\exp\left[-0.2\sqrt{0.5\sum_{i=1}^{2}\mathbf{x}_i^2}\right] - \exp\left[0.5\sum_{i=1}^{2}\cos(2\pi\mathbf{x}_i)\right] + e + 20$	$f(0,0)=0$	$-4 \leq \mathbf{x}_i \leq 4$
Griewank	$1 + \frac{1}{4000}\sum_{i=1}^{2}\mathbf{x}_i^2 - \prod_{i=1}^{2}\cos\left(\frac{\mathbf{x}_(i}{\sqrt{i}}\right)$	$f(0,0)=0$	$-10 \leq \mathbf{x}_i \leq 10$
Michalewicz	$-\sum_{i=1}^{2}\sin(\mathbf{x}_i)\sin^{20}\left(\frac{i\,\mathbf{x}_i^2}{\pi}\right)$	$f(2.20, 1.57) = -1.801$	$0 \leq \mathbf{x}_i \leq \pi$
Rastrigin	$20 + \sum_{i=1}^{2}\left[\mathbf{x}_i^2 - 10\cos(2\pi\mathbf{x}_i)\right]$	$f(0,0)=0$	$-5.12 \leq \mathbf{x}_i \leq 5.12$
Styblinski-Tang	$\frac{1}{2}\sum_{i=1}^{2}\left(\mathbf{x}_i^4 - 16\mathbf{x}_i^2 + 5\mathbf{x}_i\right)$	$f(-2.0935, -2.0935) \simeq -78.33$	$-5 \leq \mathbf{x}_i \leq 5$

Table 3.1: The benchmark functions that we use

To achieve all these requirements we create the predictors using Gaussian processes, which are initialised with a subset of the search space $\mathcal{A} \subset \mathcal{X}$. The set $\mathcal{A}$ is chosen according to a Latin hypercube sampling [66] that is overlaid on the space $\mathcal{X}$.

Latin hypercube sampling is a method for generating near-random samples from a multivariate distribution. Compared to random sampling, it is able to reduce the number of samples necessary to approach the real distribution of the sampled function [69].

To further reduce the regularity, each point in the set $\mathcal{A}$ is perturbed using a random shift. In full detail, our technique is as follows.

1. Firstly, we extend the original search space by 10% in each dimension, obtaining a new space $\mathcal{X}' \supset \mathcal{X}$. Afterwards a subset of 400 input points $\mathcal{A} \subset \mathcal{X}'$ is generated, according to a Latin hypercube sampling [66].

2. To further increase the variability between different predictors a second set of points $\mathcal{A}'$ is generated, in which each point in $\mathcal{A}$ is translated by a random offset. This is done by constructing a point $(\mathbf{x} + \alpha,\ y + \beta)$ for each input variable $x \in \mathcal{A}$, where α and β are distributed according to $\mathcal{N}(0, 0.2)$.

3. A set of training points $\mathcal{D}_p = \{(\mathbf{x}_j, f(\mathbf{x}_j) + \alpha N)\}$ is generated, where $\mathbf{x}_j \in \mathcal{A}$,

and a value of $+1$ or -1 is randomly assigned to α, with the probability of either choice being 0.5.

4. The set of points $\{(\mathbf{x}, v(\mathbf{x}) : x \in \mathcal{A}'\}$ are fitted using a Gaussian process with squared exponential kernel, $\mathcal{GP}(\boldsymbol{\mu}(\mathbf{x}), k(, \mathbf{x}'))$. The posterior mean of the GP, $\boldsymbol{\mu}$, is our predictor function p.

The resulting predictor is a deterministic smooth function over the entire domain, which assumes values $p = \boldsymbol{\mu}_p \approx f(\mathbf{x}) \pm N$ for $x \in \mathcal{A}$, while its value in the space $\mathcal{X} \setminus \mathcal{A}$ is still affected by N.

In Step 1, we extend the domain beyond the original search space to ensure that the predictor gives reasonable answers on the boundary of the domain. Without this, the GP will have high variance on the boundary, leading to poorer quality predictions on the boundary relative to the rest of the space.

Creating predictors for benchmarking. Despite the parameter N being fixed during the construction process of the predictors we still observed a wide variety of predictors. Moreover, different benchmark functions react to increases in the error parameter to different extents. This problem can be addressed as follows. For each predictor it is possible to quantify its inaccuracy via the mean squared error between f and p, defined as:

$$\hat{E} = \frac{\sqrt{\sum_{i=1}^{M} [f(\mathbf{x}_i) - p(\mathbf{x}_i)]^2}}{M}. \tag{3.9}$$

For our setup, we sampled $\hat{E}$ according to a grid containing $M = 30000$ points. However, $\hat{E}$ is not normalized across different benchmark functions, and thus it cannot be used directly as a measure of accuracy. It is instead convenient to define the *accuracy* of a predictor as $\text{acc}(p) = \hat{E}/\Delta f$, where $\Delta f = \max_{x \in \mathcal{X}} [f(\mathbf{x})] - \min_{x \in \mathcal{X}} [f(\mathbf{x})]$.

For the experiments reported below three target error levels were fixed for each benchmark: $\text{acc}(p) = \{0.05, 0.10, 0.15\}$. From here on these three levels will be referred to as *low*, *medium*, and *high error*, respectively. Once the desired levels of accuracy have been set, for each level a subset of predictors has been selected which have that level of accuracy. This has been achieved by binary search over the values of N.

For each accuracy value 100 predictors have been generated, and then N has been adjusted upwards or downwards depending on whether the average accuracy was too high or to low relative to the target accuracy. Then, once an appropriate value of N was found, there was still a wide range of accuracies in the generated predictors. Thus all predictors that were further than 5% away from the target accuracy.

After this pruning, there were at least 20 generated predictors left in the benchmarking set. The resulting predictors and error levels are shown in Figure 3.4.

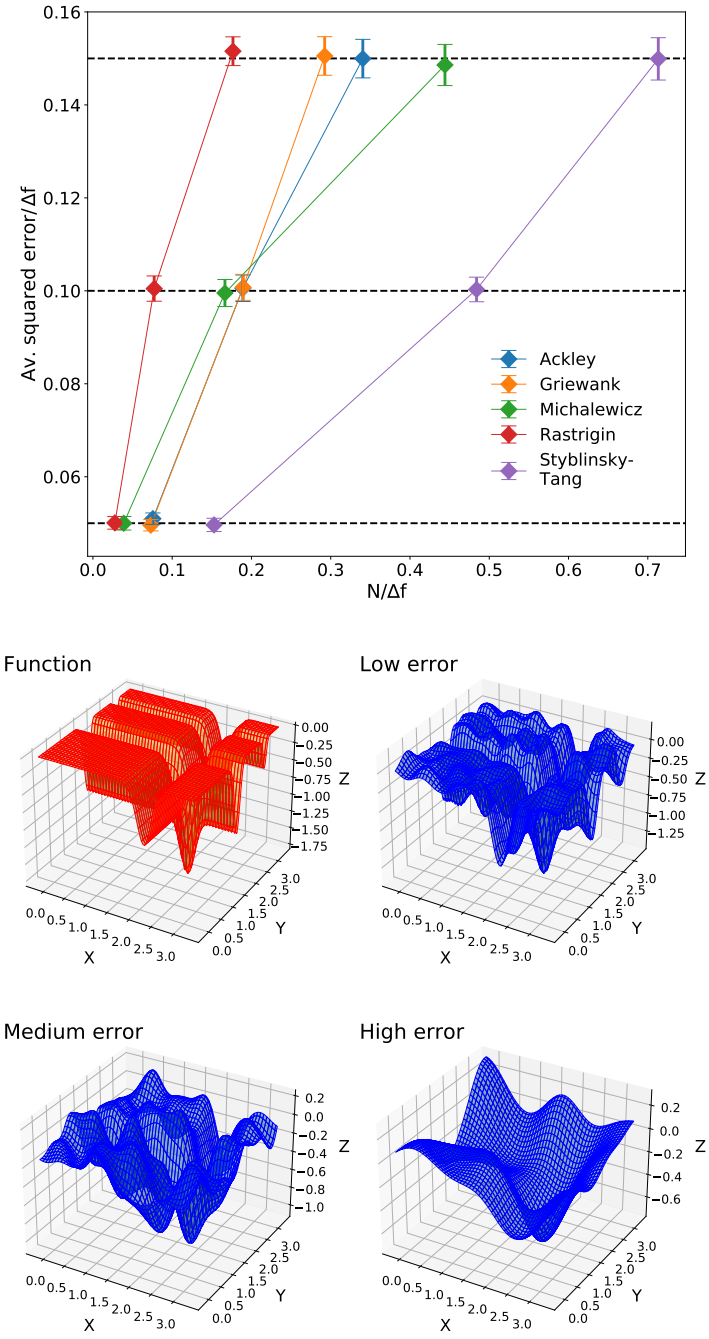

Figure 3.4: Top: the error parameters used to generate predictors for each function. N has been divided by Δf to partially normalize across the benchmarks. Bottom: example of predictors at the three error levels for the Michalewicz function.

3.5.2 Experimental parameters

A summary of all the parameters can be found in Tables 3.2 and 3.3 in this section. Both exclusion radius and discrepancy prediction methods have been benchmarked against the standard Bayesian optimisation method that does not use the predictor at all, and against a standard multi-fidelity approach that treats the predictor as a lower fidelity model.

For a fair comparison both our methods and multi-fidelity were run until the objective function f was queried a fixed number of times, so that all the methods could exploit an equal amount of real data. In all experiments, this number is set to 80 real queries. For all methods except multi-fidelity, this means that the number of steps is fixed, since those methods query one real point in each iteration, while for the multi-fidelity approach the number of iterations was unbounded, since any iteration that queried the predictor was not counted. For practical reasons however the multi-fidelity method was stopped after a maximum of 1000 iterations in total.

The quality of each method is measured according to the *regret* of the optimal point $\mathbf{x}_{\text{opt}}$ found by the method, which is defined as $R(t) = f(\mathbf{x}_{\text{opt}}(t)) - f_{\text{opt}}$, where t indicates the BO iteration step. As the $R(t)$ is averaged over all the experiments on the individual predictors, the actual measure of performance is the *averaged regret*:

$$\hat{R} = \langle f(\mathbf{x}_{\text{opt}}(t)) - f_{\text{opt}} \rangle = \langle f(\mathbf{x}_{\text{opt}}(t)) \rangle - f_{\text{opt}}. \tag{3.10}$$

For the exclusion radius method and discrepancy prediction method the optimum point $\mathbf{x}_{\text{opt}}$ is defined as the point that minimizes the mean of the surrogate model $\mathcal{M}$, while for the multi-fidelity method $\mathbf{x}_{\text{opt}}$ is defined as point that minimizes the mean of the high-fidelity surrogate model.

Standard Bayesian optimisation. The standard Bayesian optimisation method is initialized with five random points, and it is otherwise unaltered.

Exclusion radius method. The exclusion radius method is initialized with zero real points and 50 predicted points. The performance of this method is tested for a range of different values for the radius parameter r.

Each benchmark has a different sized domain, so absolute values of r cannot be compared across different benchmark functions. For this reason, the values of r are selected relative to the size of the search space. Since each benchmark function has a square shaped search space with side-length l (see Table 3.1), the values of r are chosen so that $r/l = \{0.05,\ 0.1,\ 0.15,\ 0.2,\ 0.3\}$, corresponding to 5%, 10%, 15%, 20%, and 30% of the size of the search space.

Additionally, the case where $r/l = 0$ was tested. Setting the radius to zero implies that no points will be deleted during the optimisation. This will allow us to

compare the exclusion radius method and the discrepancy prediction method against a baseline method that does not delete points.

Discrepancy prediction method. For the discrepancy prediction method we chose a GP with an RBF kernel as surrogate model $\mathcal{M}_\delta$, which was initialised with 5 points, which were randomly sampled from the search space, and for which both the objective function and the discrepancy $\delta(\mathbf{x}) = f(\mathbf{x}) - p(\mathbf{x})$ were evaluated.

The surrogate model for the objective $\mathcal{M}$ was initialised with a total of 50 points, which included the 5 points used to initialize $\mathcal{M}_\delta$. Unlike the exclusion radius method, discrepancy prediction method needs no other parameters.

Both the exclusion radius method and the discrepancy prediction method were implemented on top of the package for Bayesian optimisation GPyOpt [7].

The multi-fidelity method. To compare exclusion radius method and discrepancy prediction method to multi-fidelity Bayesian optimisation, the predictor and the actual objective function were treated as low and high fidelity model respectively. Since both our methods use one single predictor for each experiment, the multi-fidelity experiments reported were set with only two levels of fidelity.

Although one of the working assumptions for this work is that the the predictor is essentially free to query, applying the multi-fidelity method requires us to assign a finite and positive cost to both the objective and the predictor.

To be consistent with the hypotheses that the f is much more expensive than p, a much higher cost should be assigned to the first than to the second. However doing so caused the method to almost exclusively query p, and the stopping condition of 80 real observations was not reached in a reasonable amount of time. For this reason the cost of p was set to 1, and two testing values were chosen for the cost of f: 2 and 10.

The multi-fidelity method was initialized by using 5 random real points for the high-fidelity function, and 50 random points for the low-fidelity function. The code for the multi-fidelity experiment uses the implementation provided by the python package Emukit [84]. The cost aware acquisition function chosen for all experiments is obtained by dividing the standard entropy search by the cost of each model.

Method	**# Real**	**# Pred.**	$\hat{A}/\Delta f$	r/l	**Cost**
Standard	5	0	—	—	—
Exclusion radius	0	50	0.05, 0.1, 0.15, 0.2	0, 0.05, 0.1, 0.15, 0.2, 0.3	—
Discrepancy prediction	5	45	0.05, 0.1, 0.15, 0.2	—	—
Multi-fidelity	5	50	0.05, 0.1, 0.15, 0.2	—	2, 10

Table 3.2: Experimental setup for each method. # Real and # Pred. denote the number of real and predicted points used to initialize the method. $\hat{A}/\Delta f$ denotes the predictor accuracies that were tested. r/l gives the values of the radius parameter for the exclusion radius method, while cost denotes the costs that were tested for the multi-fidelity method.

Function	Error parameters (N)			Exclusion radius (r)					
	5%	10%	15%	0%	5%	10%	15%	20%	30%
Ackley	0.83	2.8	3.75	0	0.4	0.8	1.2	1.6	2.4
Griewank	0.12	0.31	0.48	0	1.0	2.0	3.0	4.0	6.0
Michalewicz	0.07	0.3	0.8	0	0.16	0.31	0.47	0.63	0.94
Rastrigin	2.23	6.22	14.22	0	0.51	1.02	1.54	2.05	3.07
Styblinski-Tang	50.18	158.91	234.18	0	0.5	1.0	1.5	2.0	3.0

Table 3.3: Absolute values of the set up parameters for the exclusion radius method and for each benchmark functions.

3.6 Results

Experimental results are shown in Figure 3.5, on a logarithmic scale, and in Figure 3.6 on a linear scale. The experiments with the exclusion radius and with the discrepancy prediction methods have been performed using a package built on top of the python library GPyOpt [7], while the results with the multi-fidelity method have been accomplished using the Emukit [84] library.

We observed that both the exclusion radius method and the discrepancy prediction method outperform standard Bayesian optimisation, especially during the early stages of the optimisation process. This is particularly clear in the linear-scale charts. However in the logarithmic scale it can be seen that multi-fidelity methods eventually catch up once we are very close to the optimal point.

Another measure of performance considered here is the number of real observations needed to get within 5% of the optimal value. This percentage is quoted relative to the average value of the benchmark function: we define $\hat{f}$ to be the average value of f, computed by sampling 10000 points according to a grid design, and then we set our target as $0.05 \times \left(\hat{f} - f_{\text{opt}}\right)$.

The plot in Figure 3.7 shows the first real observation at which each method achieves a regret that is better than this value. This data is reported in more detail, for each level of error of the predictors, in Tables 3.4, 3.5, and 3.6.

Analysis. For the exclusion radius method, it can be seen that as the error level of the predictor increases, the optimal values for r/l also increase, indicating that a higher number of predicted points need to be discarded in higher error regimes. Generally optimum values for the exclusion radius are between 0 and $0.15 \times l$ in the low error regime, between $0.05 \times l$ and $0.2 \times l$ in the medium error regime, and between $0.1 \times l$ and $0.2 \times l$ in the high error regime. In reality the level of error of the predictors is not known a priori, but the results shown in Figure 3.7 indicate that the choice $r/l = 0.1$ is suitable for all the three regimes.

Surprisingly, the benchmark obtained by setting the radius to $r/l = 0$, meaning that no points were deleted, was competitive, though not optimal, in the low and medium error regimes. However, the method performs very poorly in the high error regime. Hence, deleting points does have a positive effect on convergence speed. The results also show that, for low and medium error predictors, the discrepancy prediction method is competitive with the exclusion radius technique on four out of the five benchmark functions, with only the Styblinski-Tang function showing poorer convergence in the earlier stages of optimisation. In the high error regime the method is less consistent.

In summary, if an approximate estimation of the error level of the predictor is

available, then the parameter r can be fine-tuned to achieve excellent results. If the error level of the predictor is unknown, then either the exclusion radius method with $r/l = 0.1$, or the parameter-free discrepancy prediction method can be used, to achieve generally good results. The trends of the average regrets obtained for each function are plotted in logarithmic and in linear scale in Figure 3.5 and in Figure 3.6 respectively.

Computation time. We also found that our methods were substantially faster in computation time when compared to the multi-fidelity approach. Figure 3.8 shows the wall clock time that was taken for each method to reach 80 real observations. The multi-fidelity approach can be seen to be substantially slower, and we found that this was for two reasons: the multi-fidelity approach uses more iterations, and each iteration takes substantially more time.

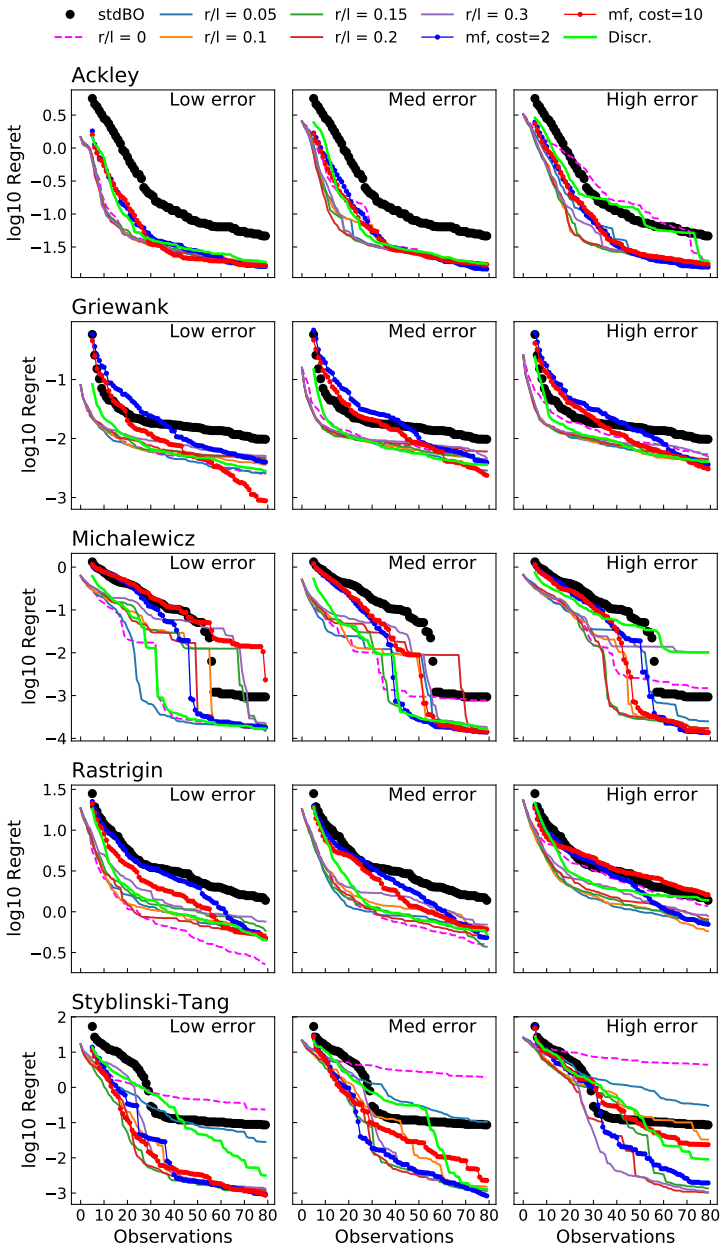

Figure 3.5: Experimental results for all methods on all benchmarks. The curves for standard BO, for the discrepancy prediction method, and for multi-fidelity Bayesian optimisation experiments start at observation 5, as they are all initialized with 5 real points. The curves for the exclusion radius method start from 0 as no real points are used for the initialization.

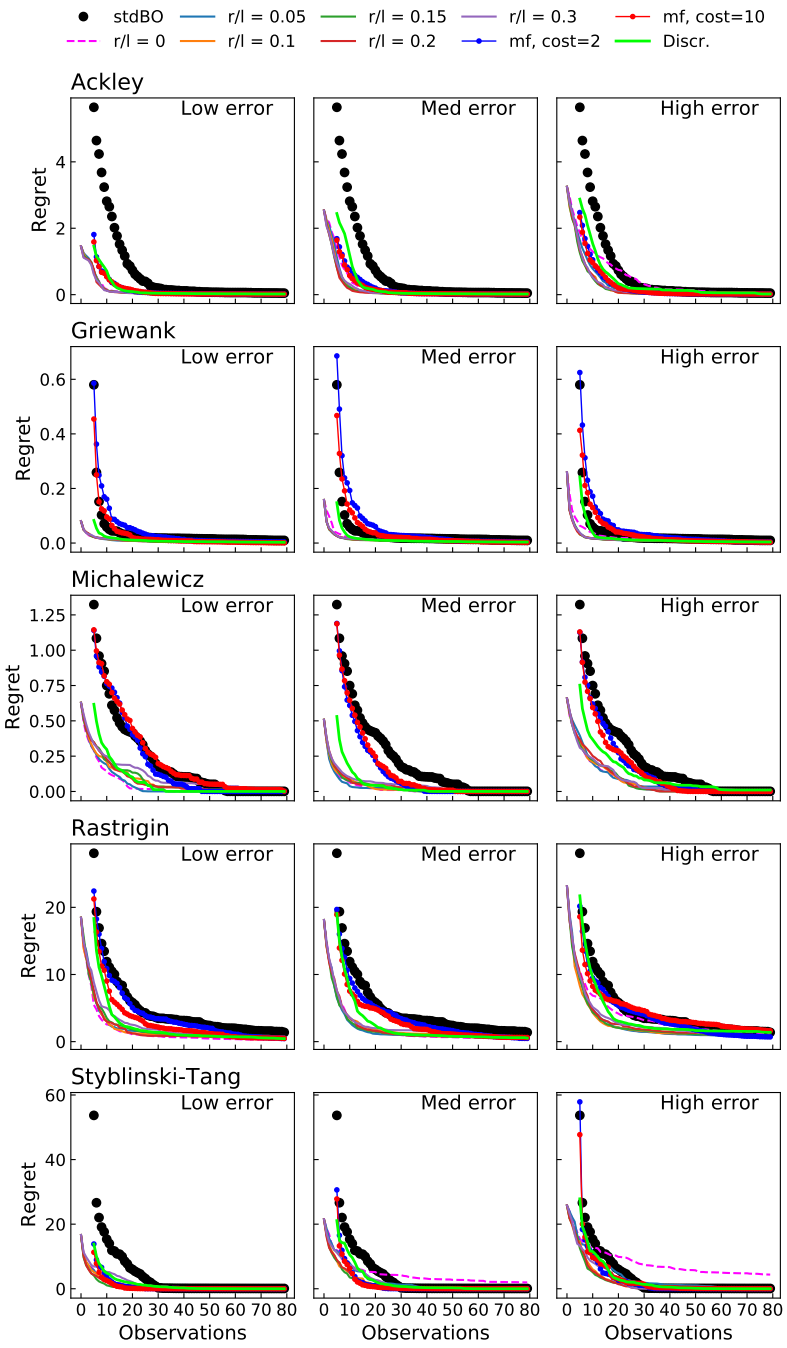

Figure 3.6: Here the average regrets shown in Figure 3.5 are plotted on a linear scale.

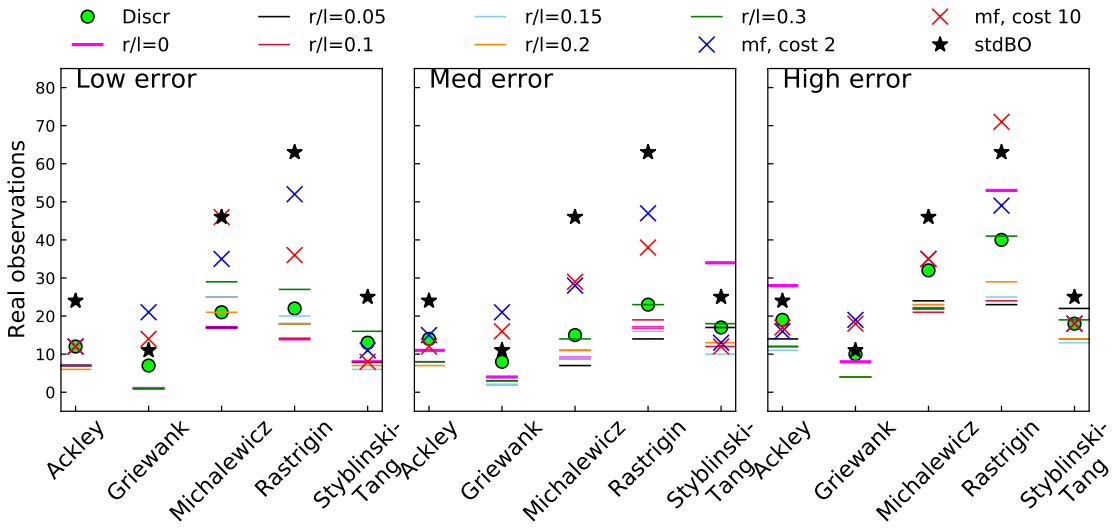

Figure 3.7: The number of real observations needed to get within 5% of the optimal value.

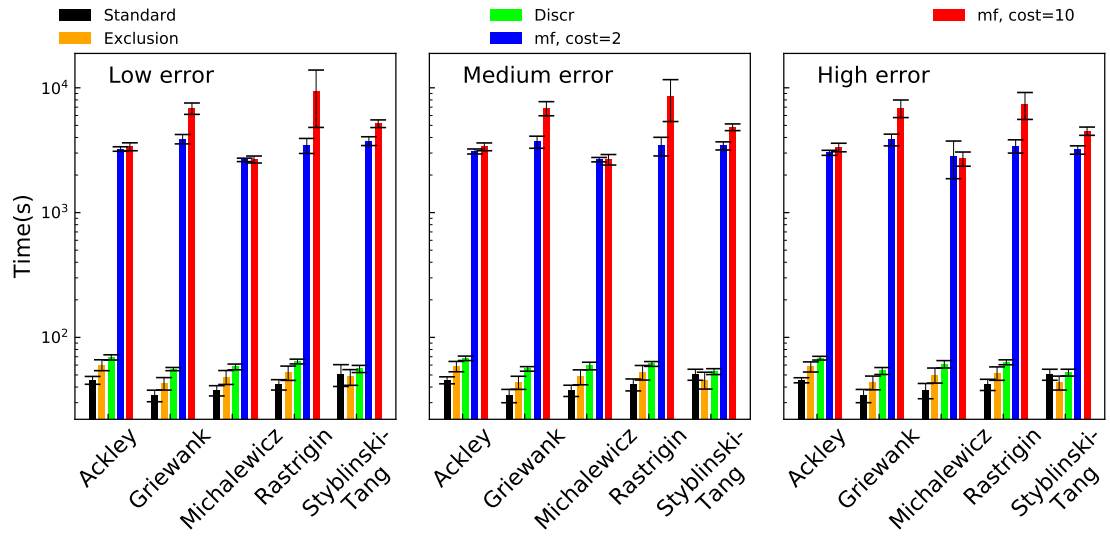

Figure 3.8: Total computational time in seconds for each method.

Low Error						
Function	r/l	Exc. radius	Standard	Discr.	MF cost 2	MF cost 10
Ackley	0	7	24	12	12	12
	0.05	7				
	0.1	7				
	0.15	7				
	0.2	**6**				
	0.3	7				
Griewank	0	**1**	11	7	21	14
	0.05	**1**				
	0.1	**1**				
	0.15	**1**				
	0.2	**1**				
	0.3	**1**				
Michalewicz	0	**17**	46	21	35	46
	0.05	**17**				
	0.1	25				
	0.15	25				
	0.2	21				
	0.3	29				
Rastrigin	0	**14**	63	22	52	36
	0.05	18				
	0.1	**14**				
	0.15	20				
	0.2	18				
	0.3	27				
Styblinski-Tang	0	8	25	13	11	8
	0.05	8				
	0.1	8				
	0.15	**6**				
	0.2	7				
	0.3	16				

Table 3.4: Number of steps needed to get within 5% of the optimal point for low error predictors.

Medium Error						
Function	r/l	Exc. radius	Standard	Discr.	MF cost 2	MF cost 10
Ackley	0	11	24	14	15	12
	0.05	8				
	0.1	8				
	0.15	8				
	0.2	**7**				
	0.3	8				
Griewank	0	4	11	8	21	16
	0.05	3				
	0.1	**2**				
	0.15	**2**				
	0.2	3				
	0.3	3				
Michalewicz	0	9	46	15	28	29
	0.05	**7**				
	0.1	11				
	0.15	9				
	0.2	11				
	0.3	14				
Rastrigin	0	17	63	23	47	38
	0.05	**14**				
	0.1	19				
	0.15	16				
	0.2	17				
	0.3	23				
Styblinski-Tang	0	34	25	17	13	12
	0.05	17				
	0.1	12				
	0.15	**10**				
	0.2	13				
	0.3	18				

Table 3.5: Number of steps needed to get within 5% of the optimal point for medium error predictors.

High Error						
Function	r/l	Exc. radius	Standard	Discr.	MF cost 2	MF cost 10
Ackley	0	28	24	19	16	17
	0.05	14				
	0.1	12				
	0.15	**11**				
	0.2	12				
	0.3	12				
Griewank	0	8	11	10	19	18
	0.05	**4**				
	0.1	**4**				
	0.15	**4**				
	0.2	**4**				
	0.3	**4**				
Michalewicz	0	22	46	32	35	35
	0.05	24				
	0.1	**21**				
	0.15	23				
	0.2	23				
	0.3	22				
Rastrigin	0	53	63	40	49	71
	0.05	**23**				
	0.1	24				
	0.15	25				
	0.2	29				
	0.3	41				
Styblinski-Tang	0	-	25	18	18	18
	0.05	22				
	0.1	14				
	0.15	**13**				
	0.2	14				
	0.3	19				

Table 3.6: Number of steps needed to get within 5% of the optimal point for high error predictors.

3.7 Conclusion

Two algorithms to accelerate Bayesian optimisation by exploiting predicted knowledge were proposed. Both methods are conceptually simple and they are competitive with state of the art methods like multi-fidelity optimisation, while requiring remarkably less computational time. Experimental findings show that a reasonable choice for the exclusion radius is $r/l = 0.1$, which is suitable for all the error levels that we considered. The discrepancy prediction method is overall less performant than the exclusion method, especially in the high error regime, but it has the advantage of not depending on any hyper-parameters.

Chapter 4

Enhanced Methods for Integrating Predicted Data into Bayesian Optimization

4.1 Introduction

This chapter describes variants of the methods reported in Chapter 3 to integrate predicted data into Bayesian optimisation. The two main methods proposed there are the *exclusion radius method* and the *discrepancy prediction* techniques. For each of these methods, here we explore several variants, with the aim to further improve the balance between the informativeness of the predicted data and its inaccuracy. For the exclusion radius method this means optimizing the rate at which the predicted points are deleted, by tuning the value of the radius r. For the discrepancy prediction method instead, higher performance can be potentially achieved by increasing the predictive accuracy of the surrogate model $\mathcal{M}_\delta$ which approximates the discrepancy. This can be achieved either by increasing the amount of training data or trying to improve the model $\mathcal{M}_\delta$ itself. The work reported in Chapter 3 has been performed using a Gaussian process as surrogate the model $\mathcal{M}_\delta$, but alternative regression models can be used to learn the discrepancy. Here the gradient boosting regressor (GBR) model has been tested, and a brief description of this model is given in Section 4.4.1.

In total four different approaches have been implemented:

1. Early switch exclusion radius method.
2. Adaptive radius exclusion method.
3. Discrepancy prediction method with a *gradient boosting regressor*.
4. Discrepancy prediction method with extra training points for $\mathcal{M}_\delta$.

All the experiments described in this chapter have been performed using the same benchmark functions, the same predictors and, where applicable, same the values of radius parameter r used for the results reported in Chapter 3. The performance of each method will be measured using the average regret, which has already been defined in Eq. (3.10):

$$\hat{R} = \langle f(\mathbf{x}_{\text{opt}}(t)) - f_{\text{opt}} \rangle.$$

It is worth here mentioning how $\mathbf{x}^{\text{best}}$ is evaluated: in Bayesian optimisation this is done simply by taking the best point among all those previously observed, i.e:

$$\mathbf{x}^{\text{best}} = \arg\min_{\mathbf{x} \in \mathcal{D}} f(\mathbf{x}). \tag{4.1}$$

In multi-fidelity BO however, $\mathbf{x}^{\text{best}}$ is evaluated by sampling the predictive mean of the surrogate model at the highest level of fidelity, $\mu_{\ell=0}$, and selecting the optimal sampled point:

$$\mathbf{x}^{\text{best}} = \arg\min_{\mathbf{x} \in X_{\text{sample}}} \mu_{\ell=0}(\mathbf{x}), \tag{4.2}$$

where $X_{\text{sample}} \subset \mathcal{X}$ This approach is followed for instance by Poloczek et al., [87], and it is also adopted for the experiments reported in Chapter 3. The reason for this procedure is that multi-fidelity BO aims to limit the queries of the objective function to minimise the evaluation cost, but querying a low fidelity model might result in acquiring poor quality observations. On the other hand more accurate values might be obtained by querying $\mu_{\ell=0}$, while still avoiding the cost of evaluating the actual objective function itself.

As the methods described in this chapter will also be compared to the multi-fidelity method, we use both Eq. (4.1) and Eq. (4.2) for $\mathbf{x}^{\text{best}}$. For clarity, in the rest of this chapter the regret R computed combining Eq. (3.10) with Eq. (4.1) will be called *predicted regret* and indicated simply with R, while the regret computed using Eq. (4.2) will be called *predicted regret*, as it is computed using the predictive mean, and it will be indicated with R_p. To compute R_p the predictive mean of the actual objective has been sampled over 10,000 points randomly sampled from the whole search space $\mathcal{X}$ according to a uniform distribution.

We point out however that using R_p as a measure of performance is due only for fair comparison with the multi-fidelity method. For some of the experiments reported below R_p is lower than the simple regret R: this indicates that, for these specific experiments, the surrogate model is a fairly accurate model of the objective, and that the acquisition function gives less accurate estimation of the position of the optimum point. We speculate that this happens because the acquisition function not only exploits the information coming from the surrogate model to find $\mathbf{x}^{\text{best}}$, but it also enforces exploration, which could result in detrimental performance if the surrogate model is accurate.

4.2 The early switch exclusion radius method

The main idea of the *early switch exclusion radius* method is to switch from exclusion radius method BO to standard BO before the optimisation process ends, by deleting all the remaining predicted points altogether at a given optimisation step t_{sw}.

From here on we will call this method simply the *early switch method*. The rationale behind this strategy is that, while the predicted points accelerate the convergence of the BO at its early stages, they could hinder the optimisation later on, because their inaccuracy could prevent the surrogate model from learning the objective function efficiently: for example a predictor could wrongly predict high values of the objective function in the proximity of the minimum. In this case the surrogate model would also return high expected values in that region, thus biasing the acquisition function to avoid sampling there. The consequence is that BO will not converge to the minimum. By removing misleading predicted points, the early switch method may help in that scenario.

The exclusion radius method is designed to remove predicted points along the way, but the amount and the location of the deleted points are not directly controlled during the experiment, as they depend on which points have been previously acquired. As a consequence, depending on the specific objective at hand, the number of predicted points remaining until the end of the BO process might still be too high, especially for the highest levels of predictors' error.

The early switch method was implemented to test this hypotheses and to investigate if and when the elimination of all predicted points is beneficial. The experiments performed using this method have the purpose to explore the possible experimental for each value of predictors' error, which will be described in Section 4.3.

4.2.1 Early switch method experimental settings

The early switch method has two parameters, the radius r and the *switch iteration* t_{sw}, at which the switch to standard BO happens. The radius r was set to $r/l = \{0.05,\ 0.10,\ 0.15,\ 0.20,\ 0.30\}$, and, for each value of r, the iteration t_{sw} was varied between 10 and 70 in steps of 10. Each single combination of experimental settings was repeated on a series of at least 20 predictors, over which the results have been averaged.

4.2.2 Results of the early switch method

The early switch method has been compared to the exclusion radius method simply by evaluating the difference between the respective average final regrets: $\Delta\hat{R} = \hat{R}_{switch} - \hat{R}_{excl}$, where the $\hat{R}_{switch}$ and $\hat{R}_{excl}$ are defined as in Eq. (3.10). The heat

maps in Figure 4.1 and Figure 4.2 report the values of $\Delta\hat{R}$ versus all the combinations of exclusion radius r and switch iteration t_{sw} for predicted and simple regrets respectively. Results are shown for all benchmark functions and for all the three levels of error of the predictors.

These figures show that the original exclusion radius method outperforms the early switch method in a large majority of experimental settings for r and t_{sw}, for all three regimes of predictors' error. Although the early switch method achieves a lower final regret at some specific combinations of r and t_{sw}, no clear trend can be inferred between these parameters and the observed values of ΔR, which prevents us from determining in which conditions the early switch method would generally perform better for most of the benchmarks. Also, the extent of such improvement varies widely among the benchmark functions, being almost irrelevant for some.

More importantly, at a given level of predictors' accuracy, the absolute best result is almost always achieved by the exclusion radius method, for an appropriate value of r. This is shown in Tables 4.1 and Tables 4.2 for simple and predicted regrets respectively. More detailed information is displayed in Tables A.1, A.2 and A.3 for simple regrets, and in Tables A.4, A.5 and A.6 for predicted regrets.

Each table reports the best final regrets observed with early switch method and the parameter t_{sw} at which they are achieved, compared with the results obtained with exclusion radius method in the same conditions.

A possible cause of the lower performance of the early switch method is the abrupt drop in the predictive accuracy of the surrogate GP model, observed after the elimination of the points. Here the predictive accuracy is defined as follows: first the objective function f is evaluated on a grid of 10,000 points, X_{grid}, then values of f predicted by the GP at X_{grid} are computed. The latter values are given by $\mu(X_{\text{grid}})$, where μ is the posterior mean of the GP, defined by Eq. (2.8). Finally the predictive accuracy A_p is defined as the Pearson correlation coefficient ρ between the actual values of the objective function, $f(X_{\text{grid}})$ and the predicted values $\mu(X_{\text{grid}})$:

$$A_p = \rho\left(f(X_{\text{grid}}),\ \mu(X_{\text{grid}})\right). \tag{4.3}$$

Compared to other possible ways of measuring the predictive accuracy, like for example mean-squared error, the Pearson correlation coefficient gives a quantity that is independent of how large values these functions can get in the search space where we studied them.

The values of A_p are plotted in Figure 4.3, Figure 4.4, and Figure 4.5. These plots show the values of A_p, evaluated at BO steps $n = \{11,\ 21,\ 31,\ 41,\ 51,\ 61,\ 71\}$, i.e. one optimisation stage after all the predicted points are deleted. It is possible to observe that:

1. A sharp drop in A_p occurs immediately after all the predicted points are deleted,

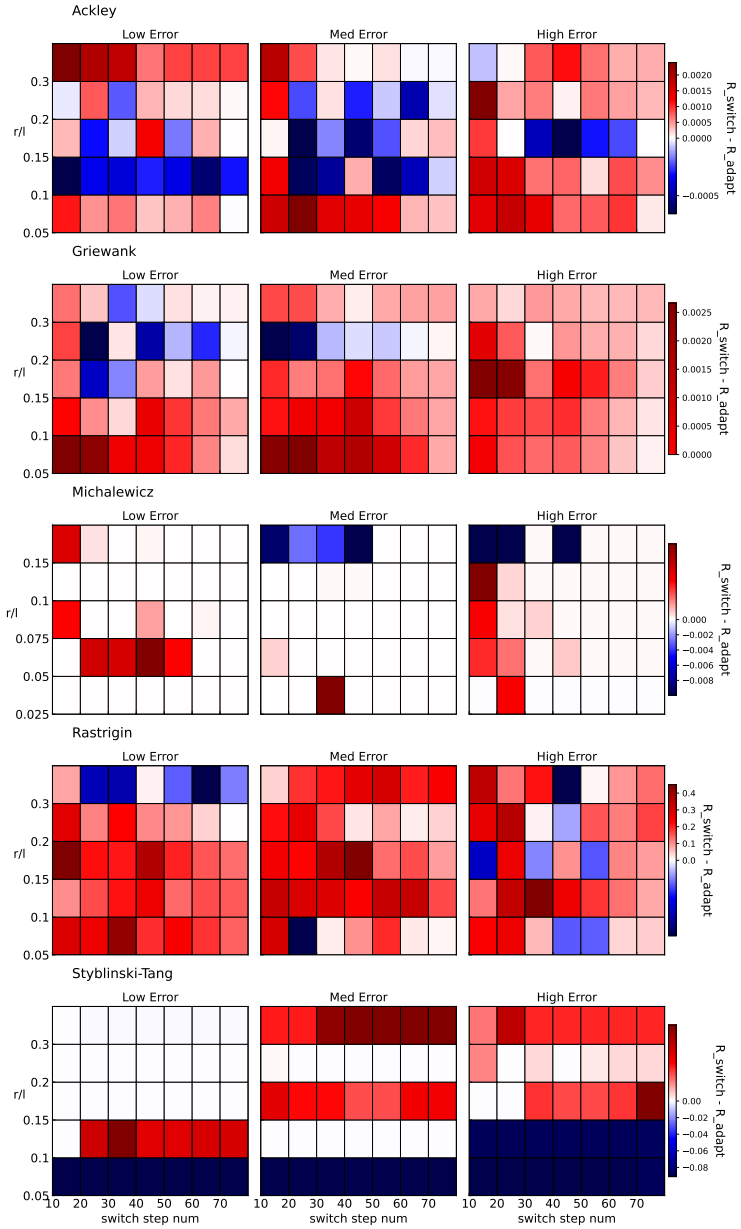

Figure 4.1: Final **predicted** regrets for different values of radius parameter r and early switch iteration t_{sw} for all the benchmark functions. Blue and red squares indicate experimental settings at which the early switch method performs better and worse than the exclusion radius method respectively.

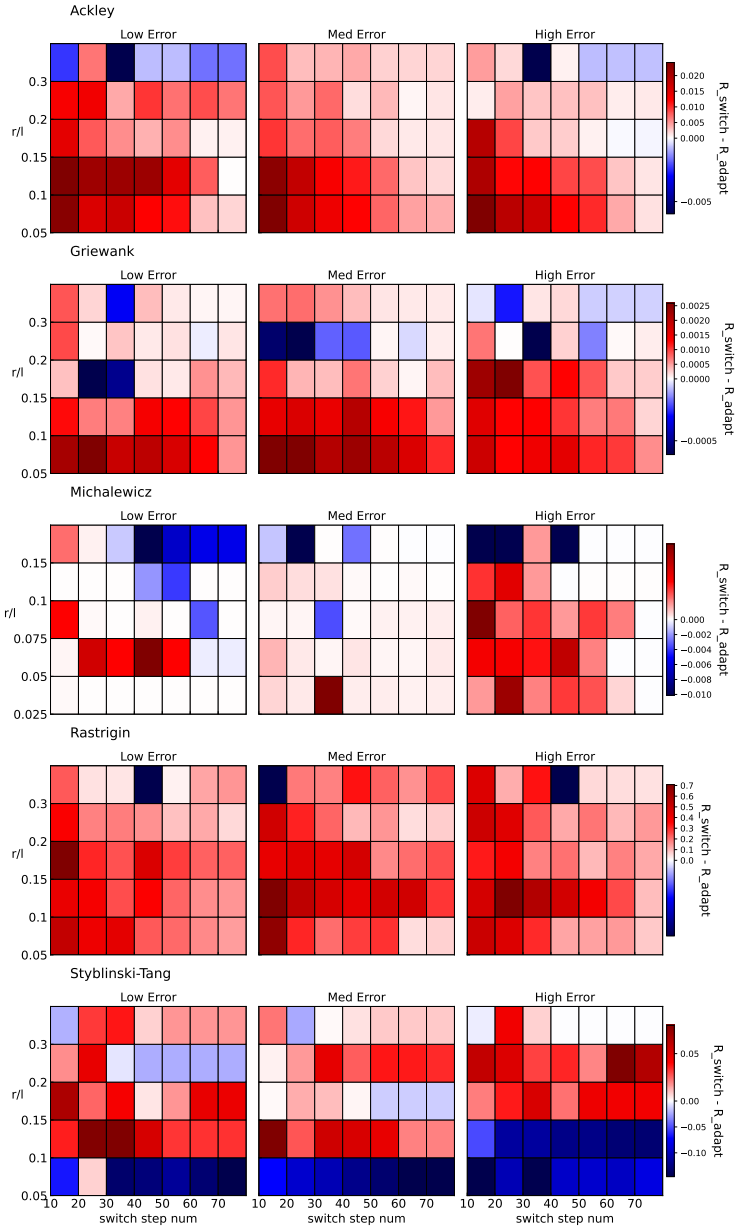

Figure 4.2: Final **simple** regrets for different values of radius parameter r and early switch iteration t_{sw} for all the benchmark functions. Blue and red squares indicate experimental settings at which the early switch method performs better and worse than the exclusion radius method respectively.

followed by a steady increase during the following optimisation steps.

2. The extent of this drop decreases with increasing t_{sw}, meaning that the later the predicted points are eliminated the lower is the deterioration of the accuracy of the surrogate model.

3. The recovery of A_p after the elimination of all predicted points presents a different behaviour across the different benchmark functions: for some, like Michalewicz and Styblinski-Tang the curves at all values of t_{sw} tend to overlap after the recovery starts. For others, like for Ackley, Griewank, and Rastrigin, the recovery rate varies with t_{sw}, and in some cases the predictive accuracy remains below the starting value until the end of the optimisation.

4. The highest accuracy is observed for $t_{sw} = 70$ in almost all cases, i.e. when all the predicted points are deleted the latest. This is a further confirmation that switching abruptly to standard BO is detrimental. The only exception to this trend occurs for the Styblinski-Tang function, which benefits from the early switch method at all error regimes.

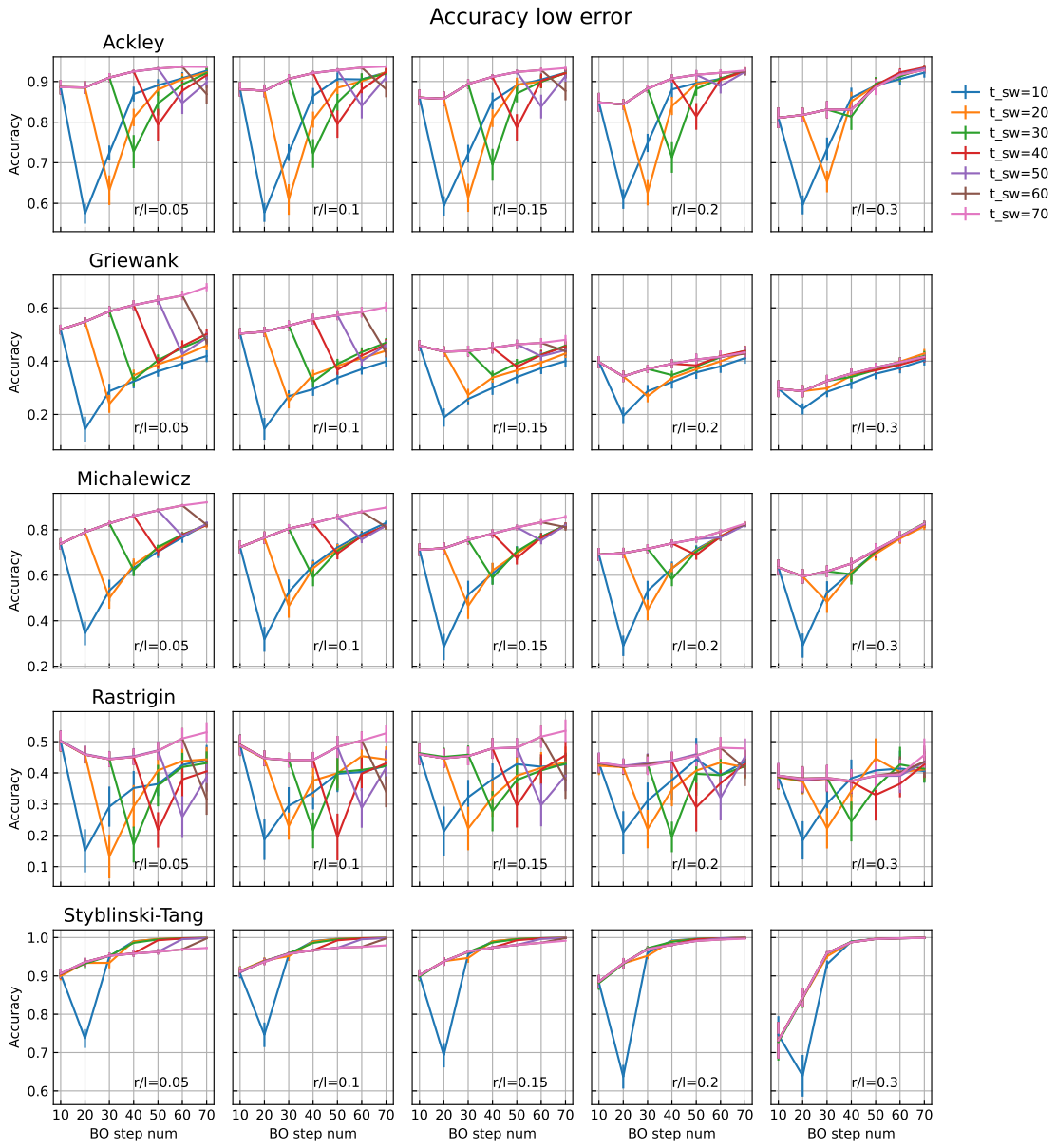

Figure 4.3: Predictive accuracy A_p for the early switch method, computed according to Eq. (4.3), in the **low** predictors' error regime. The values of A_p have been observed at BO step numbers 11, 21, 41, 51, 61 and 71. Each curve refers to experiment with a different setting for the parameter t_{sw}. The graphs show that, for each value of t_{sw}, A_p is minimum when the BO step number is equal to $t_{sw} + 1$, i.e. just after all the predicted points are deleted.

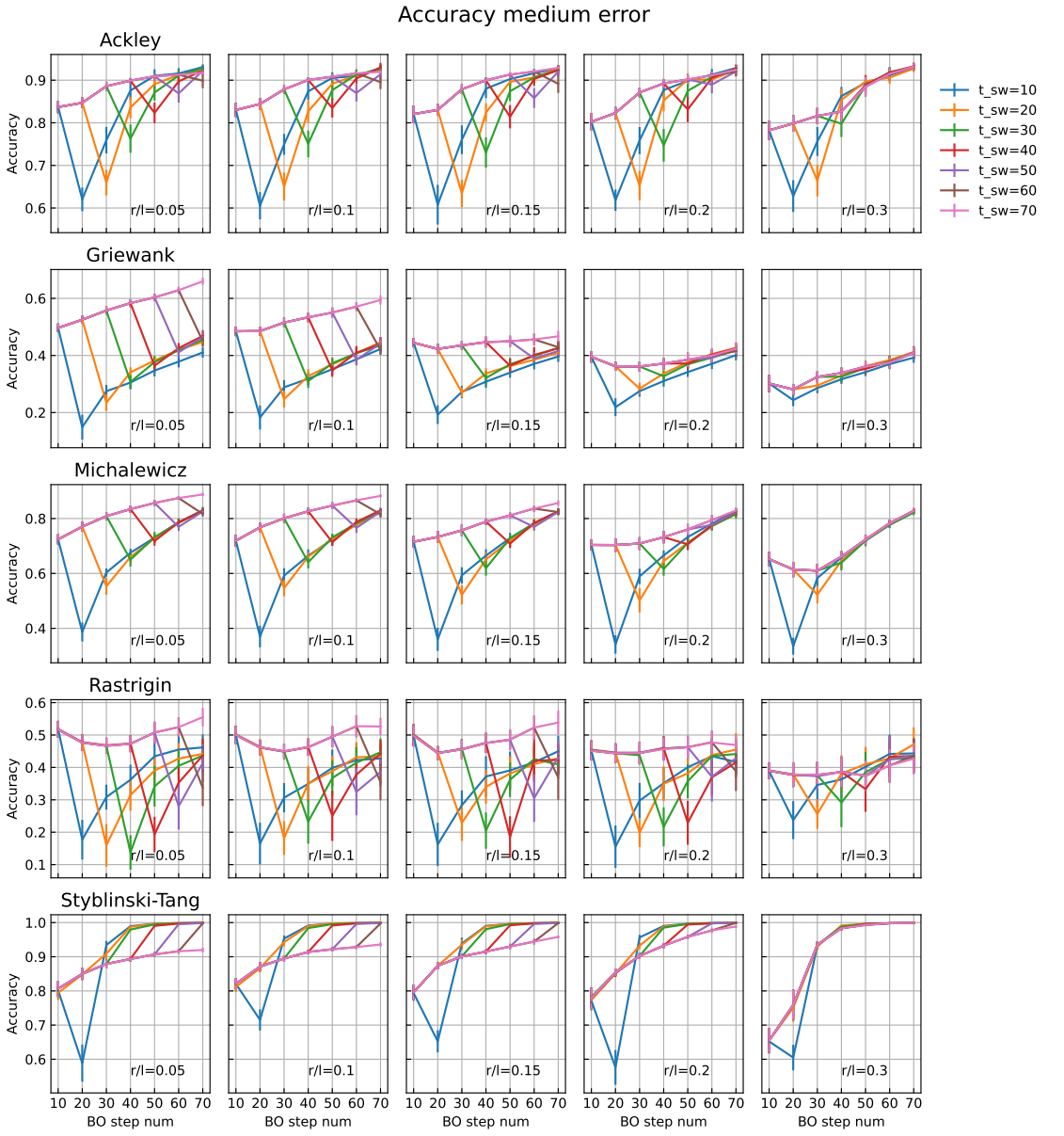

Figure 4.4: Predictive accuracy A_p for the early switch method, computed according to Eq. (4.3), in the **medium** predictors' error regime. The values of A_p have been observed at BO step numbers 11, 21, 41, 51, 61 and 71. Each curve refers to experiment with a different setting for the parameter t_{sw}. The graphs show that, for each value of t_{sw}, A_p is minimum when the BO step number is equal to $t_{sw} + 1$, i.e. just after all the predicted points are deleted.

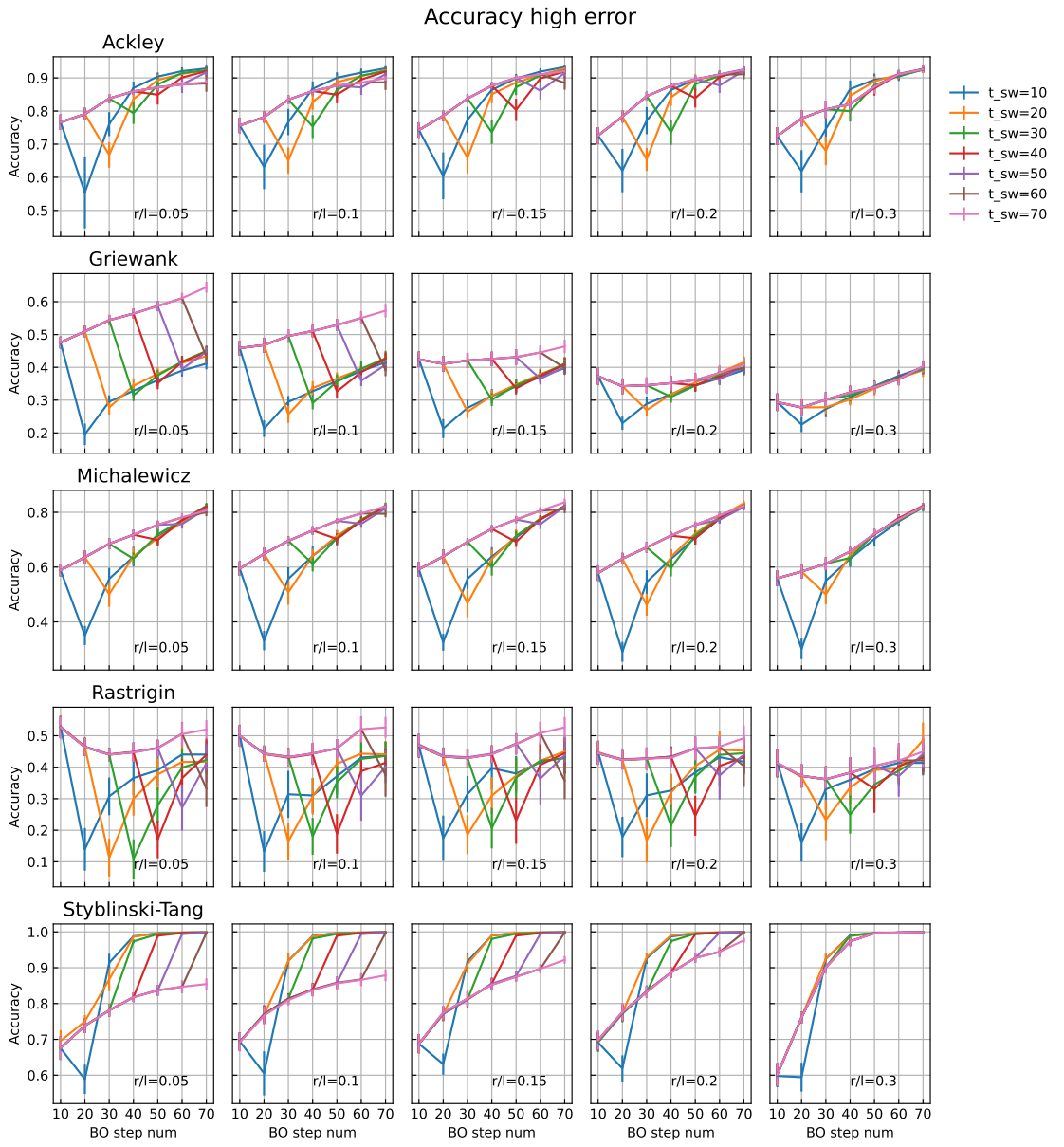

Figure 4.5: Predictive accuracy A_p for the early switch method, computed according to Eq. (4.3), in the **low** predictors' error regime. The values of A_p have been observed at BO step numbers 11, 21, 41, 51, 61 and 71. Each curve refers to experiment with a different setting for the parameter t_{sw}. The graphs show that, for each value of t_{sw}, A_p is minimum when the BO step number is equal to $t_{sw} + 1$, i.e. just after all the predicted points are deleted.

LOW ERROR				
Function	$\mathbf{R_{exclusion}}$	**r/l**	$\mathbf{R_{Switch}}$	**r/l**
Ackley	**3.282E-02 ± 4.084E-03**	0.05	3.363E-02 ± 2.246E-02	0.1
Griewank	**3.986E-03 ± 5.059E-04**	0.05	4.629E-03 ± 3.603E-03	0.05
Michalewicz	7.162E-04 ± 1.023E-04	0.1	**7.127E-04 ± 7.101E-04**	0.1
Rastrigin	**5.209E-01 ± 1.157E-01**	0.05	6.528E-01 ± 7.985E-01	0.05
Styblinski_Tang	**6.892E-02 ± 1.523E-02**	0.15	7.219E-02 ± 7.890E-02	0.15
MEDIUM ERROR				
Function	$\mathbf{R_{exclusion}}$	**r/l**	$\mathbf{R_{Switch}}$	**r/l**
Ackley	**3.697E-02 ± 3.155E-03**	0.05	3.936E-02 ± 2.782E-02	0.1
Griewank	**3.892E-03 ± 4.972E-04**	0.05	5.314E-03 ± 3.900E-03	0.05
Michalewicz	**6.505E-04 ± 7.988E-05**	0.05	7.198E-04 ± 6.687E-04	0.05
Rastrigin	**5.244E-01 ± 9.302E-02**	0.15	6.539E-01 ± 8.485E-01	0.15
Styblinski_Tang	6.940E-02 ± 1.349E-02	0.1	**6.806E-02 ± 5.470E-02**	0.05
HIGH ERROR				
Function	$\mathbf{R_{exclusion}}$	**r/l**	$\mathbf{R_{Switch}}$	**r/l**
Ackley	**3.273E-02 ± 2.864E-03**	0.05	3.407E-02 ± 2.496E-02	0.05
Griewank	**5.070E-03 ± 4.782E-04**	0.05	5.379E-03 ± 3.898E-03	0.15
Michalewicz	8.532E-04 ± 1.044E-04	0.1	**8.484E-04 ± 8.051E-04**	0.1
Rastrigin	**6.584E-01 ± 7.857E-02**	0.1	7.521E-01 ± 8.274E-01	0.1
Styblinski_Tang	7.358E-02 ± 1.210E-02	0.15	**6.823E-02 ± 7.255E-02**	0.05

Table 4.1: Comparison between the best average **simple** regrets obtained with the exclusion radius method and with the early switch method for each of the benchmark functions, at low, medium and high levels of error. The values in bold indicate the best method for a given function. The columns r/l report the the value of the parameter r/l at which the best results have been achieved for each of the two methods.

LOW ERROR				
Function	$\mathbf{R_{exclusion}}$	**r/l**	$\mathbf{R_{Switch}}$	**r/l**
Ackley	1.529E-02 ± 1.180E-03	0.05	**1.529E-02 ± 1.180E-03**	0.05
Griewank	**2.600E-03 ± 5.145E-04**	0.05	2.741E-03 ± 5.237E-04	0.05
Michalewicz	**1.503E-04 ± 2.567E-05**	0.05	1.629E-04 ± 2.709E-05	0.05
Rastrigin	**4.535E-01 ± 1.143E-01**	0.05	5.229E-01 ± 9.194E-02	0.2
Styblinski_Tang	9.200E-04 ± 1.523E-04	0.1	**8.097E-04 ± 1.644E-04**	0.15
MEDIUM ERROR				
Function	$\mathbf{R_{exclusion}}$	**r/l**	$\mathbf{R_{Switch}}$	**r/l**
Ackley	1.637E-02 ± 1.077E-03	0.05	**1.595E-02 ± 1.046E-03**	0.2
Griewank	**2.872E-03 ± 4.911E-04**	0.05	3.346E-03 ± 5.065E-04	0.05
Michalewicz	**1.494E-04 ± 2.039E-05**	0.05	1.545E-04 ± 2.032E-05	0.05
Rastrigin	**3.728E-01 ± 9.025E-02**	0.15	4.523E-01 ± 1.011E-01	0.15
Styblinski_Tang	1.073E-03 ± 1.558E-04	0.15	**1.019E-03 ± 2.457E-04**	0.1
HIGH ERROR				
Function	$\mathbf{R_{exclusion}}$	**r/l**	$\mathbf{R_{Switch}}$	**r/l**
Ackley	**1.594E-02 ± 9.688E-04**	0.05	1.604E-02 ± 9.780E-04	0.05
Griewank	**3.874E-03 ± 4.856E-04**	0.15	4.128E-03 ± 4.834E-04	0.15
Michalewicz	**1.718E-04 ± 2.895E-05**	0.15	1.731E-04 ± 2.892E-05	0.15
Rastrigin	**5.782E-01 ± 7.891E-02**	0.1	6.543E-01 ± 9.224E-02	0.1
Styblinski_Tang	**7.434E-04 ± 1.803E-04**	0.3	9.303E-04 ± 1.941E-04	0.15

Table 4.2: Comparison between the best average **predicted** regrets obtained with the exclusion radius method and with the early switch method for each of the benchmark functions, at low, medium and high levels of error. The values in bold indicate the best method for a given function. The columns r/l report the the value of the parameter r/l at which the best results have been achieved for each of the two methods.

4.2.3 Early switch: conclusions

The early switch method was proposed as a straightforward approach to accelerate the convergence pace of the exclusion radius method, by deleting all the predicted points before the end of the BO process. The aim of this strategy is to remove misleading data from the surrogate model, in order to improve its predictive accuracy. We tested the performance of the early switch method at different values of the parameter t_{sw}, which represents optimisation step at which the switch to standard BO occurs. Predicted and simple regrets were assessed, and compared with the values obtained for the original exclusion radius method. The expectation was a faster convergence for early switch BO for a suitable value of t_{sw}, depending on the level of error the predictors.

The experimental results however show that it is actually outperformed by the exclusion radius method, both in terms of predicted and simple regrets, provided an appropriate value of r is chosen. There are a few exceptions, observed mainly for the Styblinski-Tang and the Michalewicz functions, as well as for the predicted regrets obtained for the Ackley function at medium error regime.

Analysing the behaviour of the GP surrogate model, we found an abrupt decrease in its predictive accuracy just after all the predicted points are deleted. We ascribe the lower performance of the early switch method to this loss in accuracy.

4.3 The adaptive radius exclusion method

The core idea of the adaptive radius exclusion method is to progressively increase the deletion rate of the predicted points during BO with predictions, with the eventual elimination of all of them before the optimisation process ends. Similarly to the early switch method, this approach aims to eliminate inaccurate data that is potentially harmful for the convergence of the optimisation process.

However, given the finding from Section 4.2 that eliminating them abruptly is actually detrimental, and that the there is no evident correlation between the switch point and the resulting performance, here we propose to act on the parameter r, which was kept constant in the exclusion radius method and its extension the early switch method, while now it is increased by a constant amount at each iteration. This results in a gradual increase in the rate at which predicted points are deleted.

The hope is to achieve the same goals of the early switch method, but avoiding the deterioration of the GP's predictive accuracy. When $r = l/2$ all the remaining predicted points will be eventually deleted from the data set. Thus, a way to set the deletion rate is to determine at which BO step t^* this condition first occurs.

Setting the radius. The experiments reported in this chapter have been performed with a linear increase in the parameter r:

$$r = \begin{cases} r_0 & t < n \\ r_0 + \alpha t & t \geq n \end{cases} \tag{4.4}$$

where t is the current iteration, α is a coefficient which sets the increment rate for r, and n is the number of iterations during which the radius is kept constant. The value of the coefficient α is a positive real number, representing the increment rate of the radius r, and it is related to t^* according to:

$$\alpha = \frac{\frac{l}{2} - r_0}{t^*}. \tag{4.5}$$

In summary the adaptive radius exclusion method proceeds as follows:

1. Set the BO step number at which the condition $r = 0.5l$ must be reached. The value of α is then given by Eq. (4.5).
2. At each iteration, set the radius r according to Eq. (4.4).
3. Proceed with the exclusion radius method described in Algorithm 2 in Chapter 3.

4.3.1 Results of the adaptive radius exclusion radius method

Experimental settings

For the adaptive radius exclusion method the initial value of the radius, r_0, was fixed to $r_0 = 0.05l$, which is the minimum value chosen for the original exclusion method. We chose this value for r_0 because ideally we want to be able to increase r slowly, and test faster increase rates by varying the t^*: Eq. (4.5) shows that the increment rate of r is inversely proportional to t^*. Specifically, we varied t^* from 15 to 80 in steps of 10 ($t^* = \{15,\ 25,\ 35,\ 45,\ 55,\ 65,\ 75,\ 80\}$). To avoid deleting too many predicted points too early in the BO process, we kept r constant for the first 10 iterations by fixing $n = 10$.

The evolution of the predicted average regrets during optimisation is displayed in Figure 4.6 and Figure 4.7, while simple regrets are plotted in Figure 4.8 and Figure 4.9. To be able to compare the results across all the benchmark functions, the regret curves are plotted versus the parameter t^* rather than versus α, as the first one assumes the same values in all cases, while the second depends on the extension the search space of the specific function.

An overview of the final regrets reached for every value of t^* is given also in Figure 4.14 for predicted and simple regrets. Furthermore, the comparison with the original exclusion radius method is summarized in Table 4.3 for simple regrets and Table 4.4 for predicted regrets. Again, more detailed information can be found in Appendix A: Tables A.7, A.8 and A.9 show those for simple regrets, while Tables A.10, A.11 and A.12 show results for for predicted regrets.

The figures show that, regardless of the way the regrets are computed, generally the best final regret is obtained for $\alpha = 0$, which corresponds to the original exclusion radius method, and which is indicated as *exclusion* in the legends. Again, the only exception is represented by the Styblinski-Tang function, for which the exclusion radius method gives poor performance when the ratio r/l is set $r/l = 0.05$. Even when the adaptive radius exclusion method achieves lower regrets at intermediate optimisation steps, it is less effective than the exclusion radius method at the later stages of the optimisation process. In particular the values of the simple average regrets remain unvaried for quite a few BO steps.

There are at least two possible explanations for the lower performance of the adaptive radius exclusion method: firstly it is possible that the deletion of all remaining predicted points when $r/l = 1/2$ still causes a drop in the predictive accuracy of the surrogate model, similar to the early switch method. For simple regrets another possible explanation is that, when all the remaining predicted points get eliminated at $r/l = 1/2$, the uncertainty in the GP model increases, and it has more impact in the selection policy of the acquisition function: in other words exploration is promoted

over exploitation, which leads to a slow-down in the convergence of the optimisation process. This uncertainty is quantified by the variance of the GP, which here we indicate with *Sigma* and that is given by Eq. (2.9) in Chapter 2. However, this scenario does not apply to the predicted regrets, as these are selected by minimizing the predictive mean of the GP, without taking into account the variance.

To assess these hypotheses, we monitored the evolution of the accuracy A_p, defined by Eq. (4.3), at regular intervals t^*+1, analogously to what we did for the early switch method. The values of A_p are plotted in Figure 4.10, where each curve corresponds to different settings for t^*. All curves present a clear drop in the predictive accuracy just after the optimisation process reached the step $t = t^* + 1$.

The evolution of the variance of the GP, Σ, presents an opposite behaviour: in the bottom graphs of Figure 4.11, Figure 4.12 and Figure 4.13 we plotted Σ evaluated at the point $\mathbf{x}^*$ selected by the acquisition function. Each curve represents values of $\Sigma(\mathbf{x}^*)$ obtained in different experimental settings, corresponding to different values of t^*, and each curve shows a peak in correspondence of t^*, indicating that once BO reaches this step, and all the remaining predicted points are eliminated, the uncertainty on the point selected by the acquisition is maximum.

In conclusion, both hypotheses are confirmed, and thus both increases in variance and losses in predictive accuracy contribute to reducing the performance of the adaptive radius exclusion method.

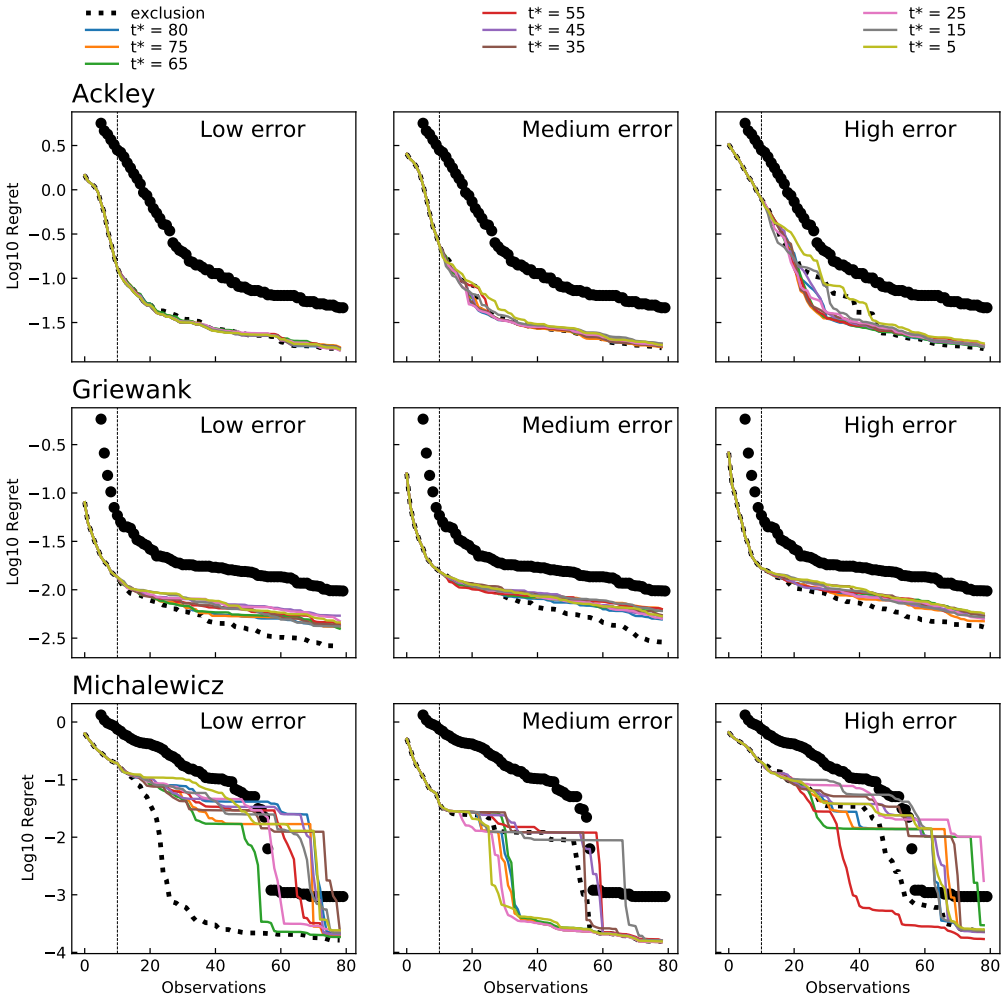

Figure 4.6: Average **predicted** regrets achieved by computing the best point at each iteration t among all the points returned by the acquisition function up to the t^{th} step. The results are plotted for the Ackley, Griewank and Michalewicz functions.

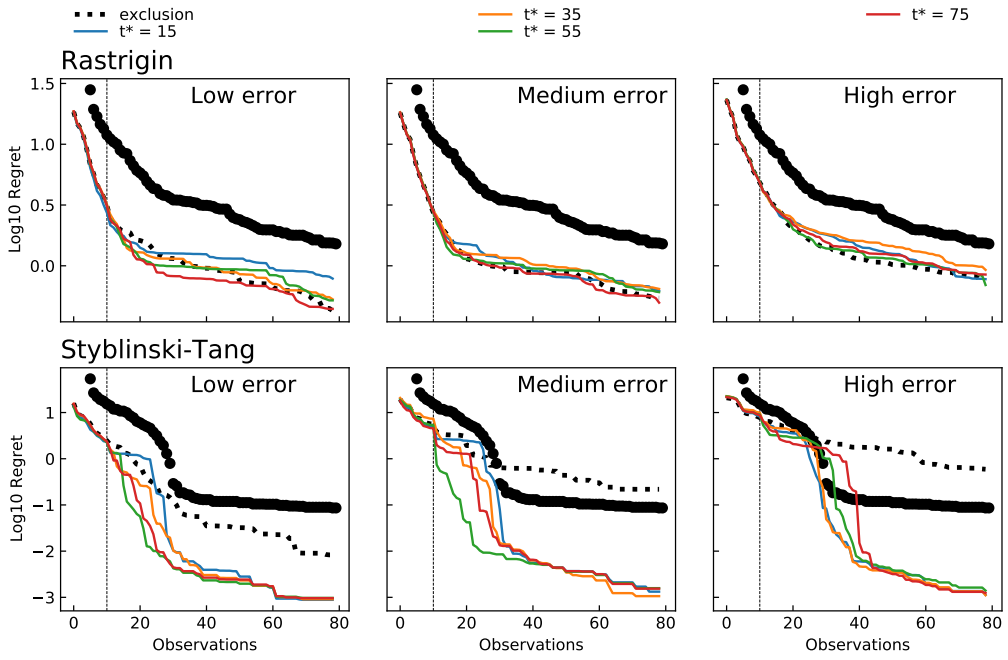

Figure 4.7: Average **predicted** regrets achieved by computing the best point at each iteration t among all the points returned by the acquisition function up to the t^{th} step. The results are plotted for the Rastrigin and Styblinski-Tang.

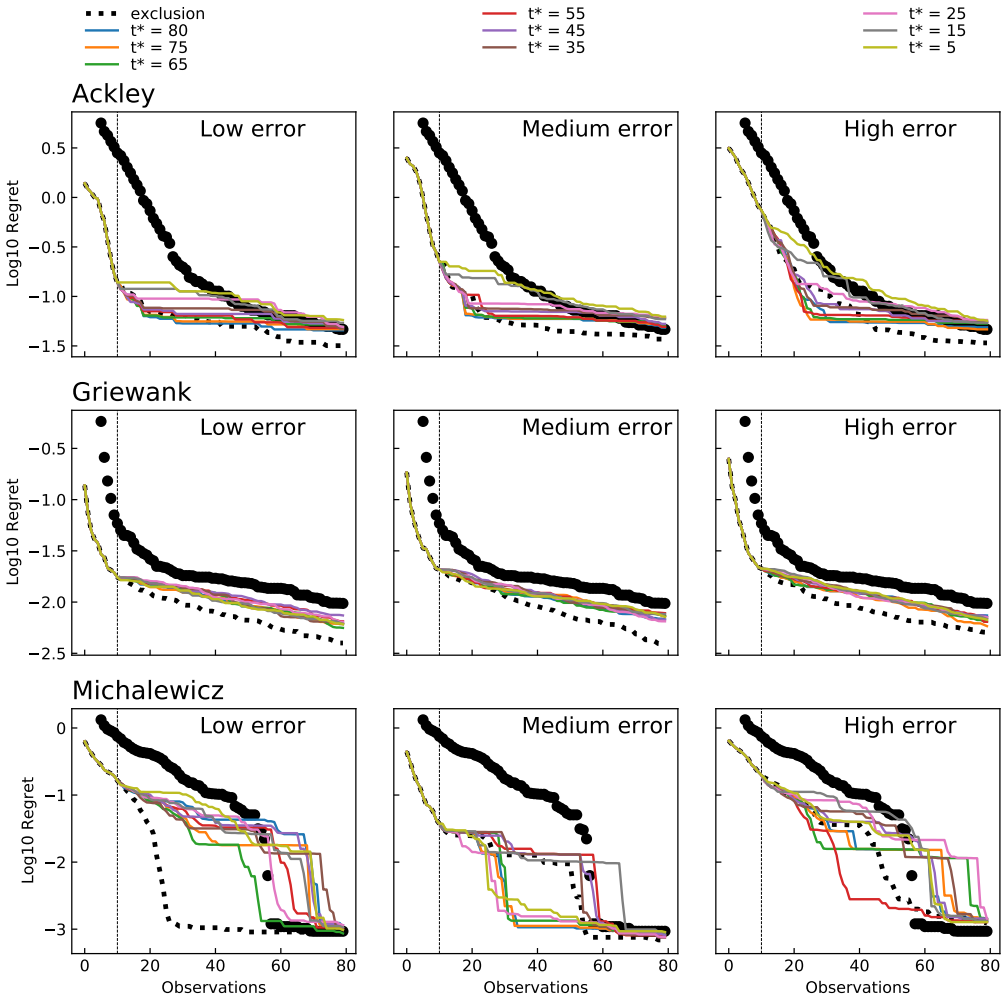

Figure 4.8: Average **simple** regrets achieved by computing the best point at each iteration t among all the points returned by the acquisition function up to the t^{th} step. The results are plotted for the Ackley, Griewank and Michalewicz functions.

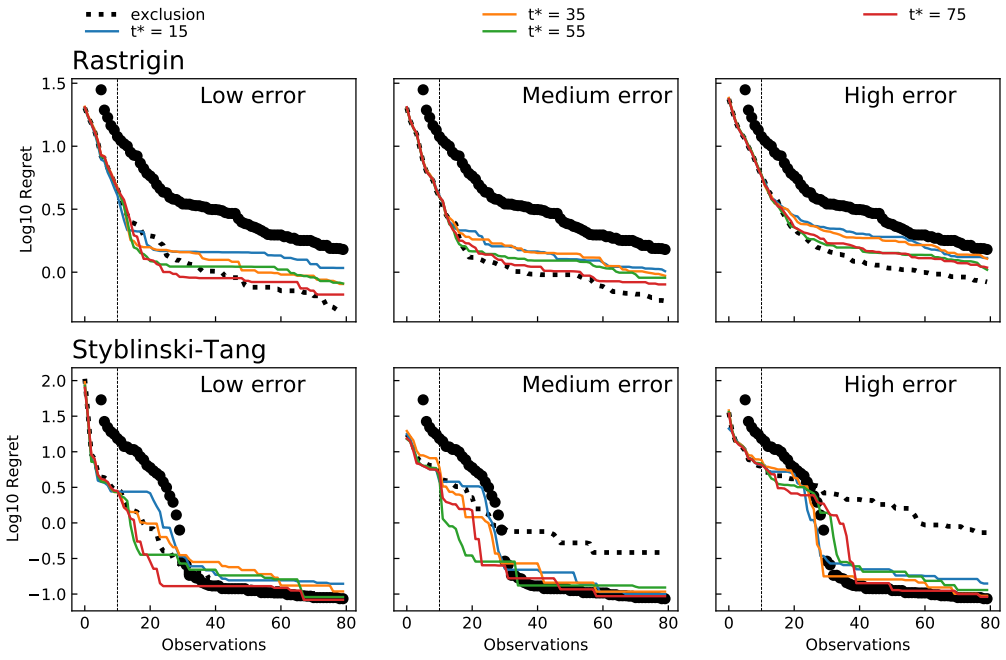

Figure 4.9: Average **simple** regrets achieved by computing the best point at each iteration t among all the points returned by the acquisition function up to the t^{th} step. The results are plotted for the Rastrigin and Styblinski-Tang.

LOW ERROR			
Function	$\mathbf{R_{exclusion}}$	**r/l**	$\mathbf{R_{Adaptive}}$
Ackley	**3.282E-02 ± 4.084E-03**	0.05	4.545E-02 ± 5.884E-03
Griewank	**3.986E-03 ± 5.059E-04**	0.05	5.599E-03 ± 5.954E-04
Michalewicz	**7.162E-04 ± 1.023E-04**	0.1	8.645E-04 ± 1.225E-04
Rastrigin	**5.209E-01 ± 1.157E-01**	0.05	6.644E-01 ± 9.768E-02
Styblinski_Tang	**6.892E-02 ± 1.523E-02**	0.15	8.212E-02 ± 1.616E-02
MEDIUM ERROR			
Function	$\mathbf{R_{exclusion}}$	**r/l**	$\mathbf{R_{Adaptive}}$
Ackley	**3.697E-02 ± 3.155E-03**	0.05	4.838E-02 ± 3.861E-03
Griewank	**3.892E-03 ± 4.972E-04**	0.05	6.486E-03 ± 5.942E-04
Michalewicz	**6.505E-04 ± 7.988E-05**	0.05	7.467E-04 ± 8.124E-05
Rastrigin	**5.244E-01 ± 9.302E-02**	0.15	7.914E-01 ± 9.453E-02
Styblinski_Tang	6.940E-02 ± 1.349E-02	0.1	**6.710E-02 ± 1.733E-02**
HIGH ERROR			
Function	$\mathbf{R_{exclusion}}$	**r/l**	$\mathbf{R_{Adaptive}}$
Ackley	**3.273E-02 ± 2.864E-03**	0.05	4.650E-02 ± 4.011E-03
Griewank	**5.070E-03 ± 4.782E-04**	0.05	5.825E-03 ± 5.581E-04
Michalewicz	**8.532E-04 ± 1.044E-04**	0.1	1.274E-03 ± 1.292E-04
Rastrigin	**6.584E-01 ± 7.857E-02**	0.1	9.599E-01 ± 1.076E-01
Styblinski_Tang	**7.358E-02 ± 1.210E-02**	0.15	9.137E-02 ± 1.664E-02

Table 4.3: Comparison between the best average **simple** regrets obtained with the exclusion radius method and with the adaptive radius exclusion method for each of the benchmark functions, at low, medium and high levels of error. The values in bold indicate the best method for a given function. The value of the parameter r/l at which the best results have been achieved for the original exclusion method have been reported in the column with the same name. The adaptive radius exclusion method has always been performed setting $r/l = 0.05$.

LOW ERROR			
Function	$\mathbf{R_{exclusion}}$	**r/l**	$\mathbf{R_{Adaptive}}$
Ackley	**1.529E-02 ± 1.180E-03**	0.05	1.529E-02 ± 1.144E-03
Griewank	**2.600E-03 ± 5.145E-04**	0.05	3.958E-03 ± 5.642E-04
Michalewicz	**1.503E-04 ± 2.567E-05**	0.05	1.847E-04 ± 2.943E-05
Rastrigin	4.535E-01 ± 1.143E-01	0.05	**4.418E-01 ± 8.848E-02**
Styblinski_Tang	9.200E-04 ± 1.523E-04	0.1	**8.480E-04 ± 1.817E-04**
MEDIUM ERROR			
Function	$\mathbf{R_{exclusion}}$	**r/l**	$\mathbf{R_{Adaptive}}$
Ackley	**1.637E-02 ± 1.077E-03**	0.05	1.671E-02 ± 1.144E-03
Griewank	**2.872E-03 ± 4.911E-04**	0.05	4.934E-03 ± 6.005E-04
Michalewicz	**1.494E-04 ± 2.039E-05**	0.05	1.499E-04 ± 1.771E-05
Rastrigin	**3.728E-01 ± 9.025E-02**	0.15	4.973E-01 ± 8.281E-02
Styblinski_Tang	1.073E-03 ± 1.558E-04	0.15	**1.061E-03 ± 2.826E-04**
HIGH ERROR			
Function	$\mathbf{R_{exclusion}}$	**r/l**	$\mathbf{R_{Adaptive}}$
Ackley	**1.594E-02 ± 9.688E-04**	0.05	1.672E-02 ± 1.025E-03
Griewank	**3.874E-03 ± 4.856E-04**	0.15	4.729E-03 ± 5.270E-04
Michalewicz	1.718E-04 ± 2.895E-05	0.15	**1.703E-04 ± 2.374E-05**
Rastrigin	**5.782E-01 ± 7.891E-02**	0.1	6.948E-01 ± 1.158E-01
Styblinski_Tang	**7.434E-04 ± 1.803E-04**	0.3	1.112E-03 ± 2.280E-04

Table 4.4: Comparison between the best average **predicted** regrets obtained with the exclusion radius method and with the adaptive radius exclusion method for each of the benchmark functions, at low, medium and high levels of error. The values in bold indicate the best method for a given function. The value of the parameter r/l at which the best results have been achieved for the original exclusion method have been reported in the column with the same name. The adaptive radius exclusion method has always been performed setting $r/l = 0.05$.

4.3.2 Adaptive radius exclusion method: conclusions

The adaptive radius exclusion method has been proposed with the intent to delete predicted points faster than in the original exclusion radius method, with the expectation that this could result in faster convergence. To address the issues observed for the early switch method, we chose to accelerate this deletion gradually instead of removing all the remaining predicted points abruptly.

However, an analysis of the evolution of the surrogate model, revealed that the predictive accuracy of the GP deteriorates also for the adaptive radius exclusion method. Additionally, we showed that just after all the the predicted points are eliminated, the optimisation process selects points with high variance, meaning that exploration is largely favoured over exploitation. The trade off between the two is recovered at the following steps, but overall the performance is negativcely effected.

We conclude that our strategies to accelerate the removal of predicted data is not beneficial, and that in general, the original exclusion radius method manages better the trade off between using using the approximated data when it is beneficial and discarding with it is detrimental.

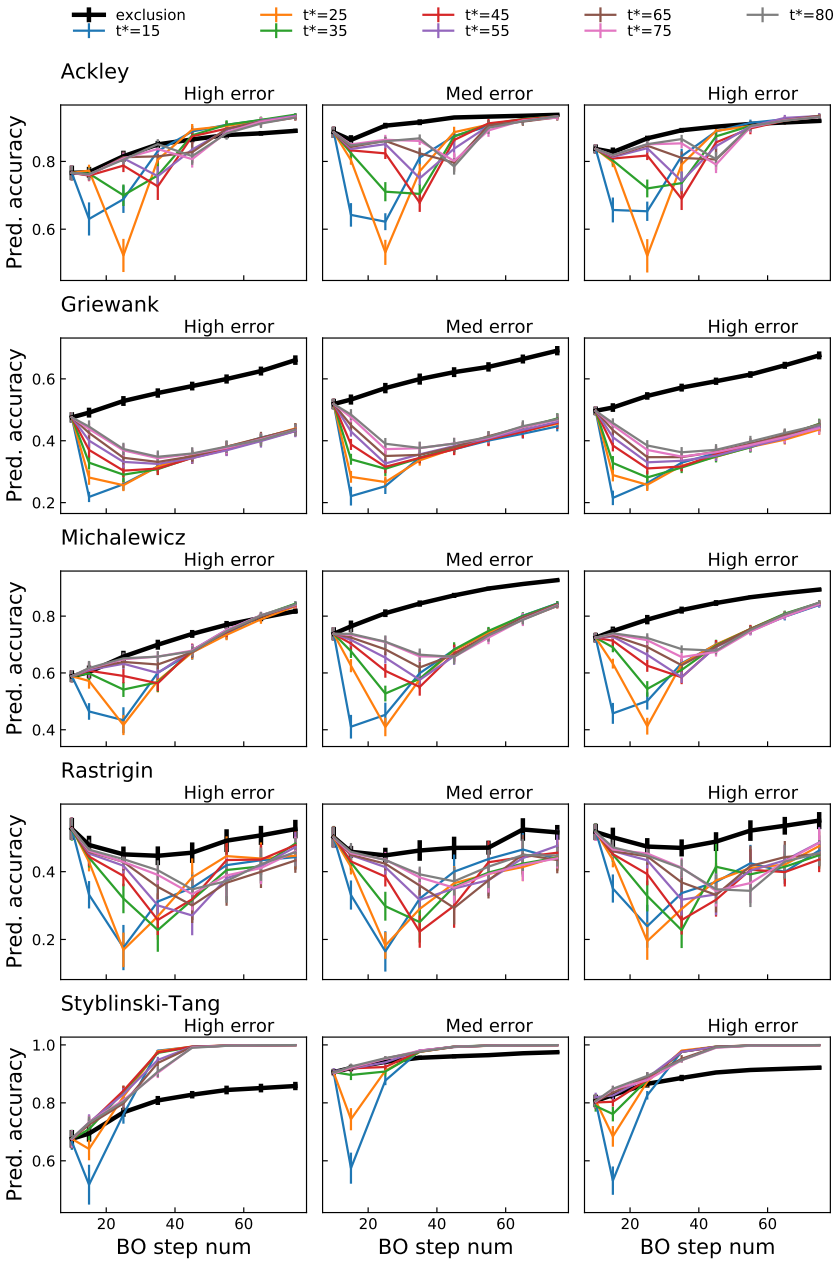

Figure 4.10: Predictive accuracy A_p for the adaptive radius exclusion method, computed according to Eq. (4.3), all predictors' error regimes. The values of A_p have been observed at BO step numbers 11, 21, 41, 51, 61 and 71. Each curve refers to experiment with a different setting for the parameter t_{sw}. The graphs show that, for each value of t_{sw}, A_p is minimum when the BO step number is equal to $t_{sw} + 1$, i.e. just after all the predicted points are deleted.

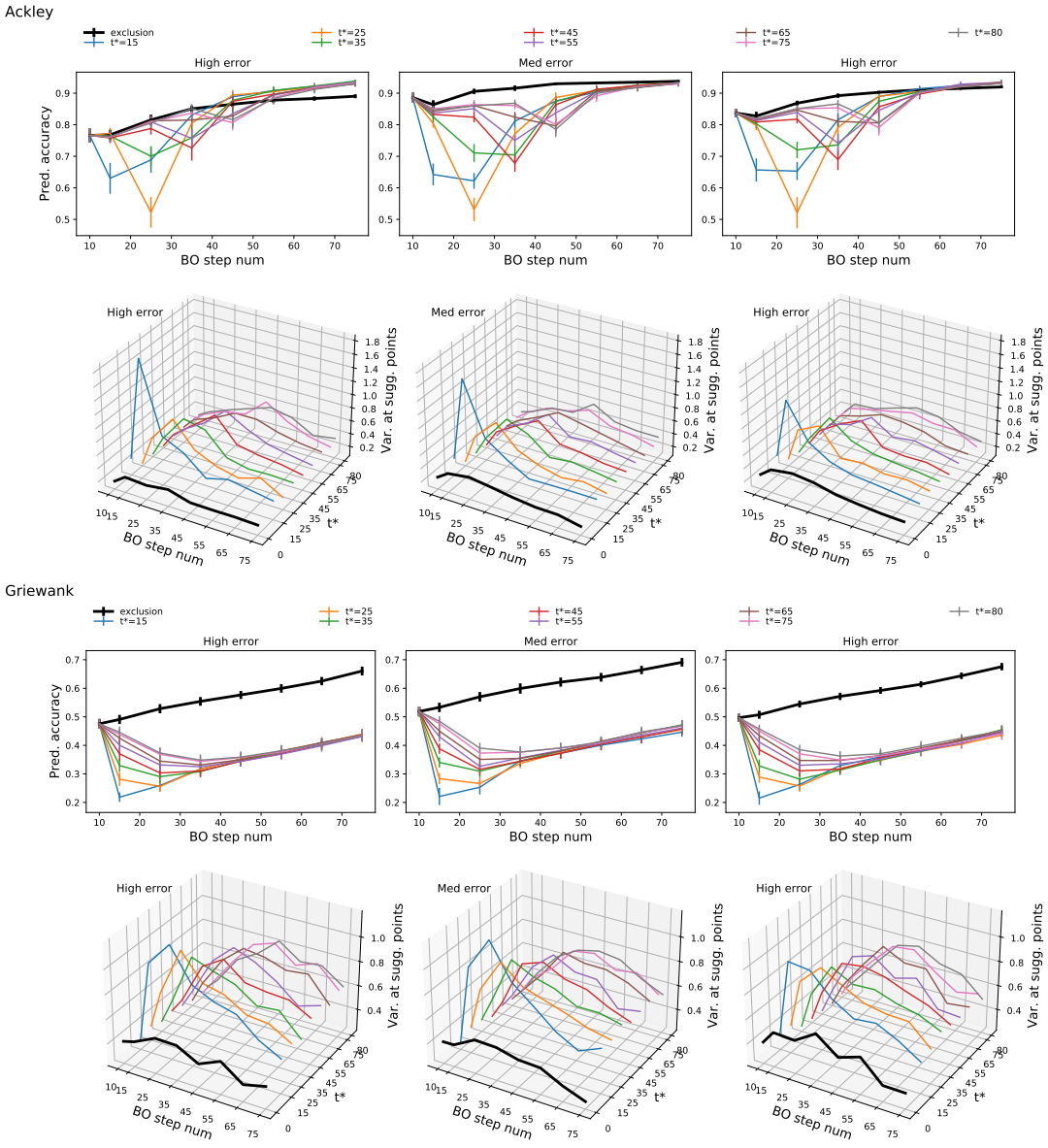

Figure 4.11: Predictive accuracy of the GP surrogate model for the adaptive radius exclusion method is displayed in the 2-dimensional graphs. The 3-dimensional graphs below show the posterior variance of the surrogate model, calculated at the point $\mathbf{x}^*$ returned by the acquisition function at BO step numbers $t^* = \{15,\ 25,\ 35,\ 45,\ 55,\ 65,\ 75,\ 80\}$, as indicated at Section 4.3. Both top and bottom graphs plot data obtained for the Ackley and Griewank functions respectively.

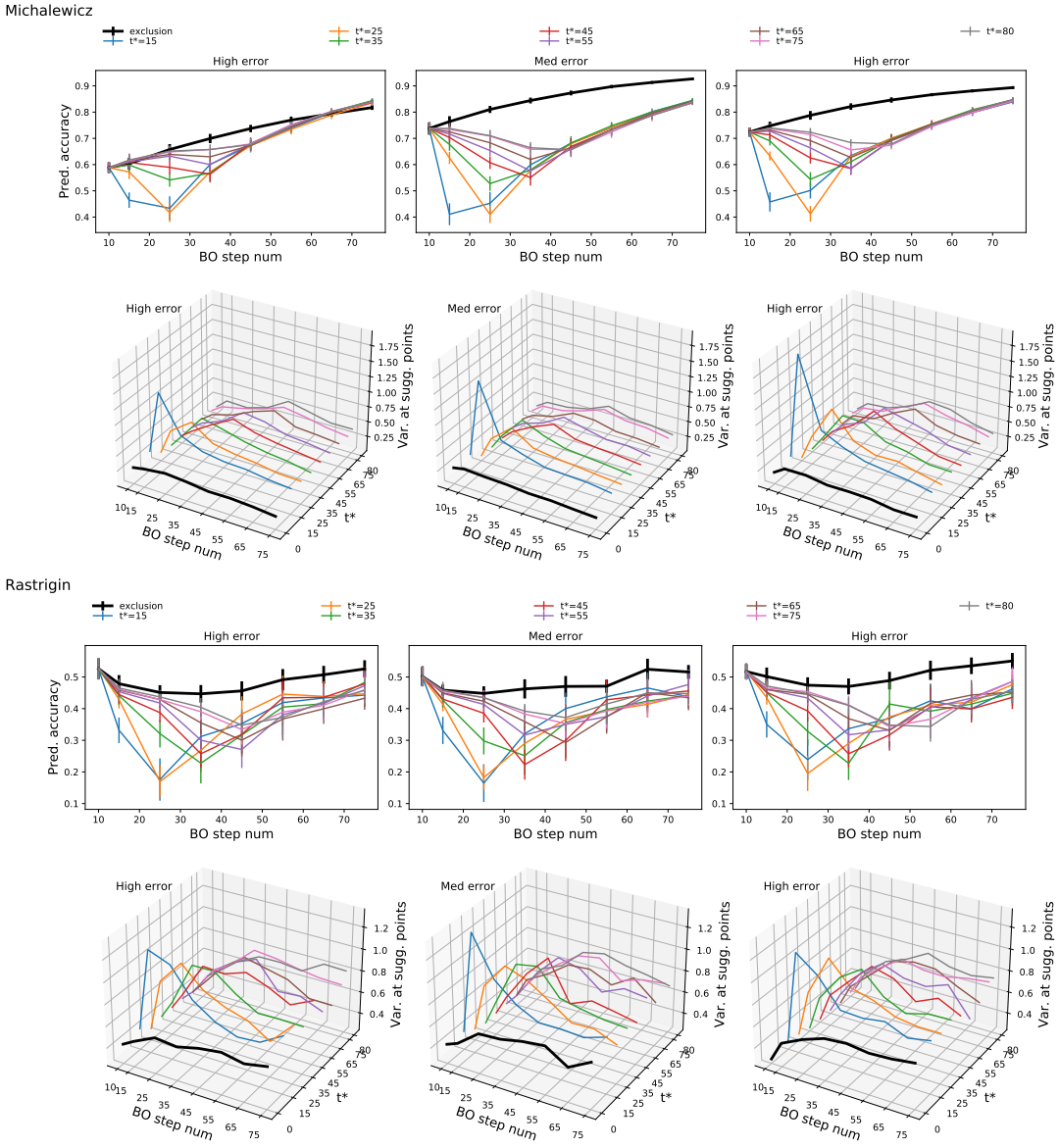

Figure 4.12: Predictive accuracy of the GP surrogate model for the adaptive radius exclusion method is displayed in the 2-dimensional graphs. The 3-dimensional graphs below show the posterior variance of the surrogate model, calculated at the point $\mathbf{x}^*$, returned by the acquisition function at BO step numbers $t^* = \{15,\ 25,\ 35,\ 45,\ 55,\ 65,\ 75,\ 80\}$, as indicated at Section 4.3. Both top and bottom graphs plot data obtained for the Michalewicz and Rastrigin functions respectively.

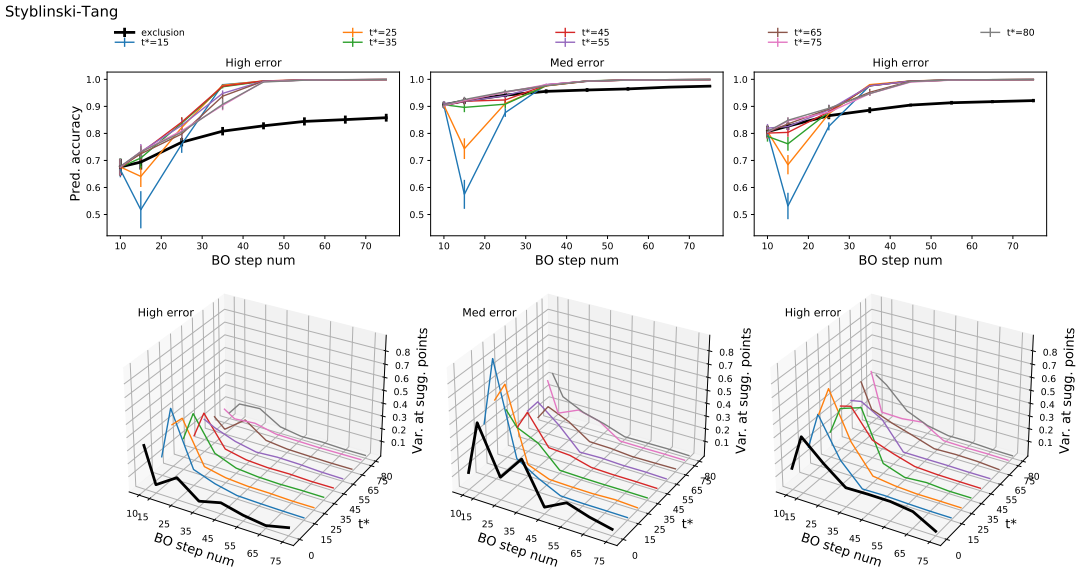

Figure 4.13: Predictive accuracy of the GP surrogate model for the adaptive radius exclusion method is displayed in the 2-dimensional graphs. The 3-dimensional graphs below show the posterior variance of the surrogate model, calculated at the point $\mathbf{x}^*$, returned by the acquisition function at BO step numbers $t^* = \{15,\ 25,\ 35,\ 45,\ 55,\ 65,\ 75,\ 80\}$, as indicated at Section 4.3. Both top and bottom graphs plot data obtained for the Styblinski-Tang function.

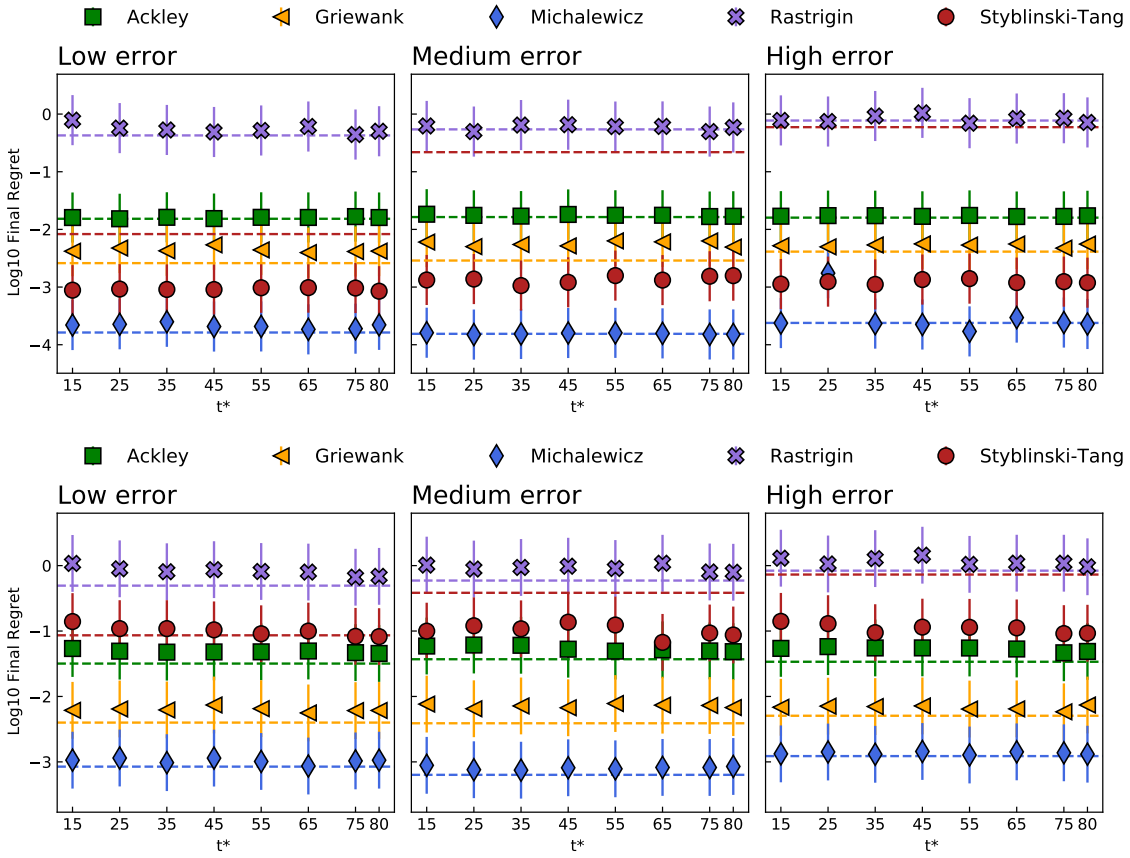

Figure 4.14: Final **predicted** average regrets (top) and **simple** average regrets (bottom) for each level of predictors' error. Dashed lines represent the final average predicted regret reached with exclusion radius method. Each colour refers to a distinct benchmark function.

4.4 Predicting the discrepancy with GBR

Gradient boosting regression (GBR) has gained increasing popularity in machine learning due its predictive ability, both in classification and regression tasks [30, 76]. For this reason we decided to employ it as a surrogate model to approximate the discrepancy between the predictors and the objective function. The Algorithm 3 reported in Section 3.4.2 is still valid, the only difference consisting in the definition of $\mathcal{M}_\delta$. In this section a very brief overview of GBR is given, while the specific settings of $\mathcal{M}_\delta$ will be described in the experimental settings section.

4.4.1 The gradient boosting regressor model

Gradient boosting is a machine learning method which uses an *ensemble* of *weak learners* to produce a prediction model. Weak learners are in general naive models, with poor predictive accuracy but low computational cost. Typically, weak learners are decision trees, and this is also the case for the work reported here. The idea behind ensemble machine learning methods is to build a model by combining a relatively large number of weak predictors, in order to achieve high accuracy. The advantage of this approach is the possibility to adjust the number of weak learners, and thus the number of model parameters, so as to reduce bias. An ensemble predictive model $F(\mathbf{x})$ is iteratively improved by adding a new weak learner $h(\mathbf{x})$ at each iteration. Thus, for each point $\mathbf{x}_i$, $F(\mathbf{x}_i)$ is given by:

$$F_{m+1}(\mathbf{x}_i) = F_m(\mathbf{x}_i) + \beta_m h_m(\mathbf{x}_i; \mathbf{a}_m), \tag{4.6}$$

where $h_m(\cdot\,; \mathbf{a}_m)$ is the new weak learner added at stage m, which is defined by the set of hyper-parameters $\mathbf{a}_m$. The parameter β_m is the weight assigned to each learner. The model $h_m(\cdot\,; \mathbf{a}_m)$ is evaluated so as compensate as much as possible the predictive error obtained at stage m, which is quantified by the *residual* $r_m = y_i - F_m(\mathbf{x}_i)$, where y_i is the actual value observed at $\mathbf{x}_i$. This is accomplished by optimizing the hyper-parameters $\mathbf{a}_m$ and the weights β_m so as to minimise the error function, or cost function, chosen to quantify the overall predictive error of F_m. A common choice for the cost function is the *mean squared error* (MSE), defined as:

$$L_{MSE} = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - F_m(\mathbf{x}_i)\right)^2 . \tag{4.7}$$

Thus the optimisation problem to solve is:

$$(\beta_m, \mathbf{a}_m) = \arg\min_{\beta, \mathbf{a}} \sum_{i=1}^{n} L\left(y_i, \beta h_m(\mathbf{x}_i,\ \mathbf{a})\right) . \tag{4.8}$$

The solutions of Eq. (4.8) are computed via gradient descent. Thus the solution at step m, F_m, is given by [14, 22]:

$$\begin{aligned} F_m(\mathbf{x}) =& F_{m-1}(\mathbf{x}) - \rho_m \nabla_{F_{m-1}} L\left(\mathbf{y}, F_{m-1}(\mathbf{x})\right) \\ &= F_{m-1}(\mathbf{x}) - \rho_m \nabla_{F_{m-1}} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(\mathbf{x}_i)\right) \\ &= F_{m-1}(\mathbf{x}) + \rho_m g_m(\mathbf{x}), \end{aligned} \tag{4.9}$$

where ρ_m is the *learning rate*, and

$$-g_m(\mathbf{x}_i) = -\left[\frac{\partial L\left(y, F(\mathbf{x}_i)\right)}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x}_i)=F_{m-1}(\mathbf{x}_i)}. \tag{4.10}$$

The parameters of the weak learner are given by:

$$\mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^{n} \left[-g_m(\mathbf{x}_i) - \beta h(\mathbf{x}_i;\mathbf{a})\right]^2. \tag{4.11}$$

The parameter ρ_m is computed via line search, and it is given by:

$$\rho_m = \arg\min_{\rho} \sum_{i=1}^{n} L(y_i, F_{m-1}(\mathbf{x}_i) + \rho h(\mathbf{x}_i;\mathbf{a})). \tag{4.12}$$

Once the parameters $\mathbf{a}_m$ and ρ_m have been optimised, the model F_m becomes:

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x};\mathbf{a}_m) \tag{4.13}$$

Gradient boosting tree regression

In this section a few more details about gradient boosting regression are given in the specific case regression trees are chosen as weak predictive models, as we do for the discrepancy prediction method. Regression trees are themselves additive models:

$$h(\mathbf{x};\{b_j R_j\}_{j=1,\dots,J}) = \sum_{j=1}^{J} b_j \mathbb{1}(\mathbf{x} \in R_j), \tag{4.14}$$

where J is the number of terminal node of a given tree, R_j with $j = 1, \dots, J$ are the disjoint regions that collectively cover the whole space of the input variables, and b_j is a constant value predicted by the tree at the region R_j. Each region corresponds to a terminal node. Also, each region is defined by two types of parameters: its boundaries and the assigned value b_j, which represent the hyper-parameters of this

specific learner, $\mathbf{a}_m$. The equation (4.6) for the boosting gradient tree regressor can be rewritten as:

$$\begin{aligned} F_{m+1}(\mathbf{x}_i) &= F_m(\mathbf{x}_i) + \rho_m \sum_{j=1}^{n} b_{jm} \mathbf{1}(\mathbf{x}_i \in R_j) \\ &= F_m(\mathbf{x}_i) + \gamma_{j,m} \mathbf{1}(\mathbf{x}_i \in R_j). \end{aligned} \tag{4.15}$$

The workflow of gradient boosting, originally published in [31], is described in Algorithm 4.

Algorithm 4: Gradient boosting descent

Input: Training data set $\mathcal{D}_{train} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, differentiable loss function $L(\mathbf{y}, F(\mathbf{x}))$, number of iterations m

Output: $F_M(\mathbf{x})$

1 Initialize the model F with a constant value b_0: $F_0 = \arg\min_b \sum_{i=1}^{n} L(y_i, b_0)$

2 **for** *m=1,...,M* **do**

3 Compute the pseudo residuals: $\tilde{y}_i = -g_m(\mathbf{x}_i) = -\left[\frac{\partial L(y, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}$

4 Using the train data set $\mathcal{D}_{train}$ optimise the parameters of the weak learner $\mathbf{a}_m$: $\mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^{n} \left[\tilde{y}_i - \beta h(\mathbf{x}_i; \mathbf{a})\right]^2$

5 Compute the linear coefficient ρ_m: $\rho_m = \arg\min_{\rho} \sum_{i=1}^{n} L(y_i, F_{m-1}(\mathbf{x}_i) + \rho h(\mathbf{x}_i; \mathbf{a}))$

6 Update the model: $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h_m(\mathbf{x})$

7 **end**

4.4.2 Experimental results

The average predicted and simple regrets for all the benchmark functions are plotted in Figure 4.15 and Figure 4.16 respectively. The green and red curves represent respectively the results achieved using a GP or a GBR as a predictive model for the discrepancy. Predicted average regrets show comparable performance for both choices, with the exception of the Michalewicz and Styblinski-Tang functions at high error regime: in the first case a GBR model leads to better convergence, with a final regret lower than standard BO, while in the second case the opposite is observed.

An analogous behaviour is observed for simple regrets, although these last ones are significantly higher than predicted regrets, similarly to what happened for the previously described methods. Figure 4.17 and Figure 4.18 show a comparison between the final regrets obtained with the discrepancy prediction methods, with GP

and with GBR as surrogate models, and those obtained with standard BO. While a the results show a large variability among the different benchmark functions, no significant difference can be discerned between the performances achieved with either of the two models, which result comparable.

Furthermore, all the results concerning predicted regrets show that generally the discrepancy prediction method outperforms standard BO, with the only exceptions of the Michalewicz and Styblinski-Tang functions at high error regime. This is consistent with the results reported for this BO method in Chapter 3.

On the other hand, simple regrets at the high error regime show lower performance of the discrepancy prediction method with respect to standard BO for the majority of the benchmark functions, whether a GP or a GBR method is chosen to predict the discrepancy.

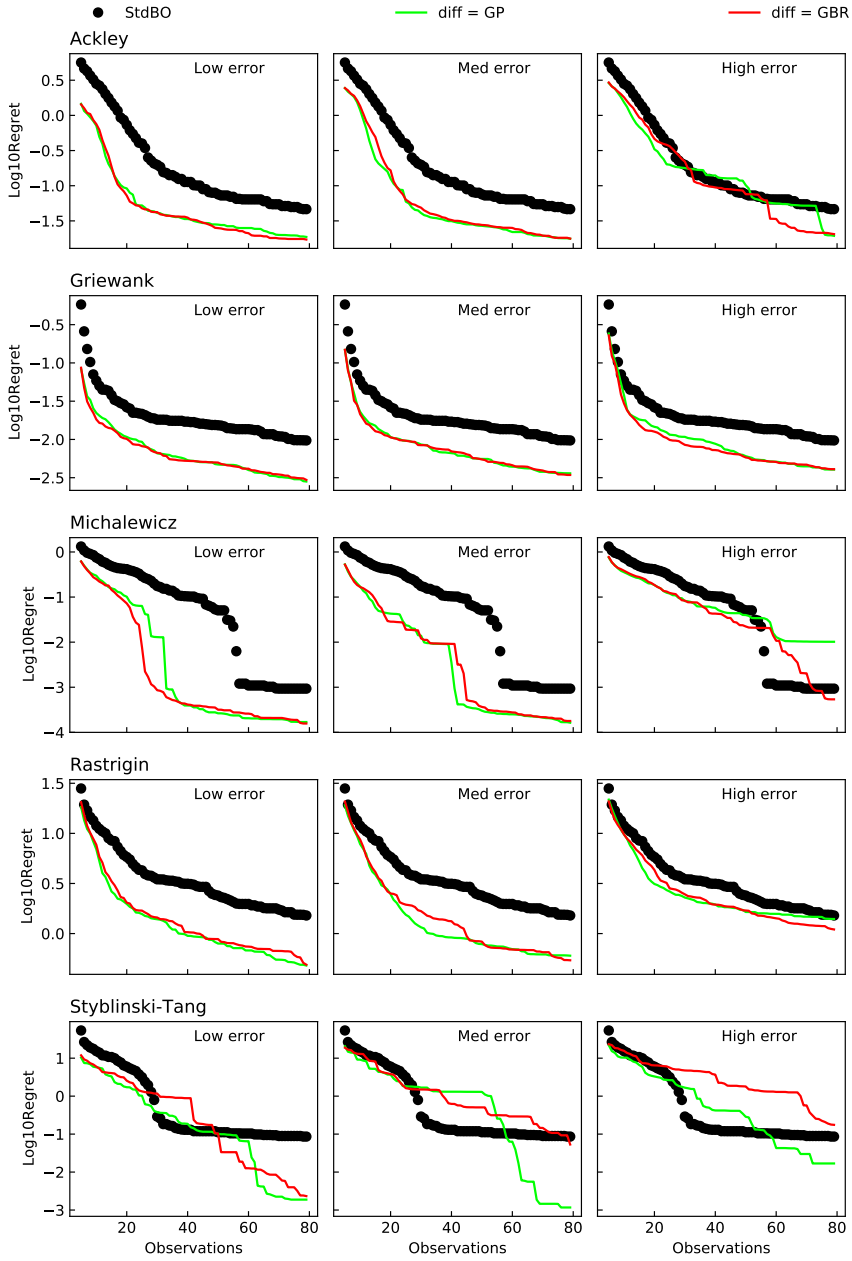

Figure 4.15: Comparison between the average **predicted** regrets obtained using a GP or a GBR surrogate model for the discrepancy. The results are plotted for all three predictors' error regimes for all the benchmark functions and compared to standard BO.

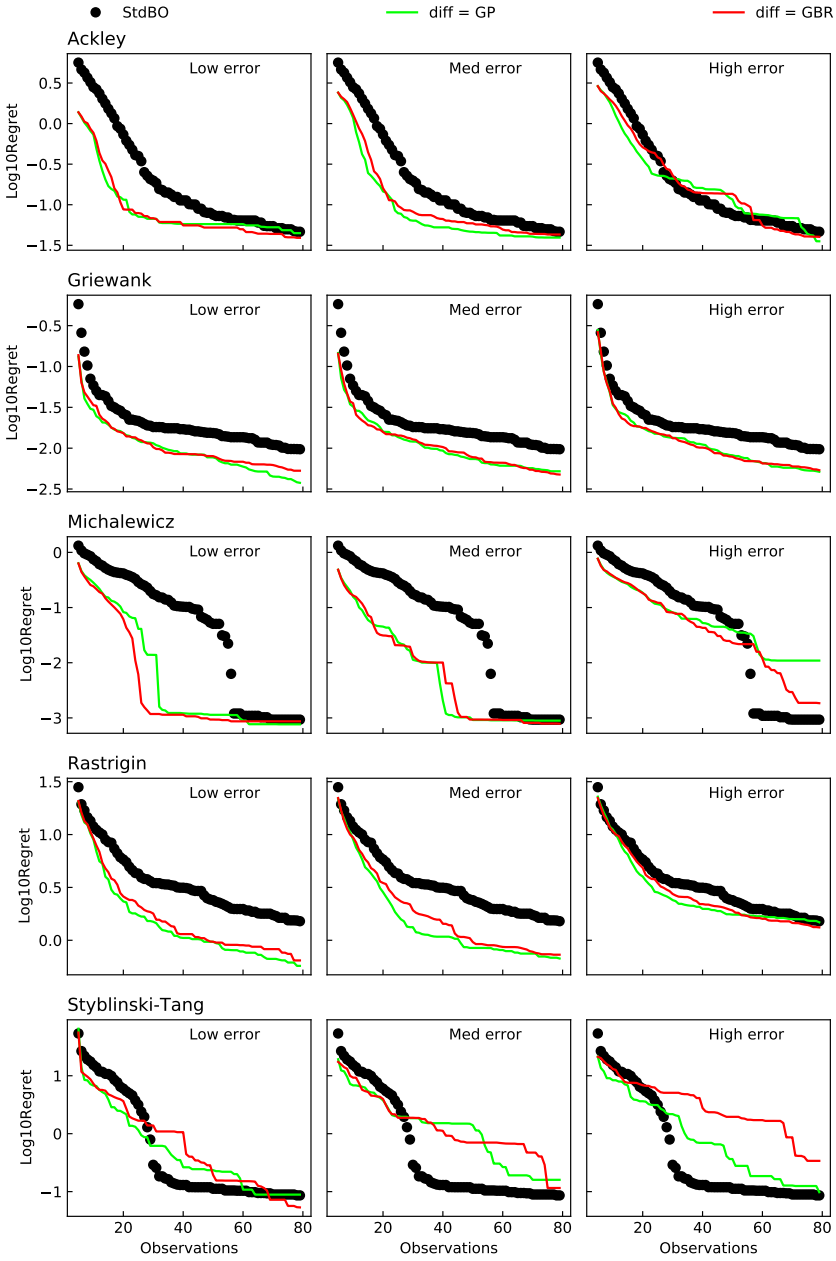

Figure 4.16: Comparison between the average **simple** regrets obtained using a GP or a GBR surrogate model for the discrepancy. The results are plotted for all three predictors' error regimes for all the benchmark functions and compared to standard BO.

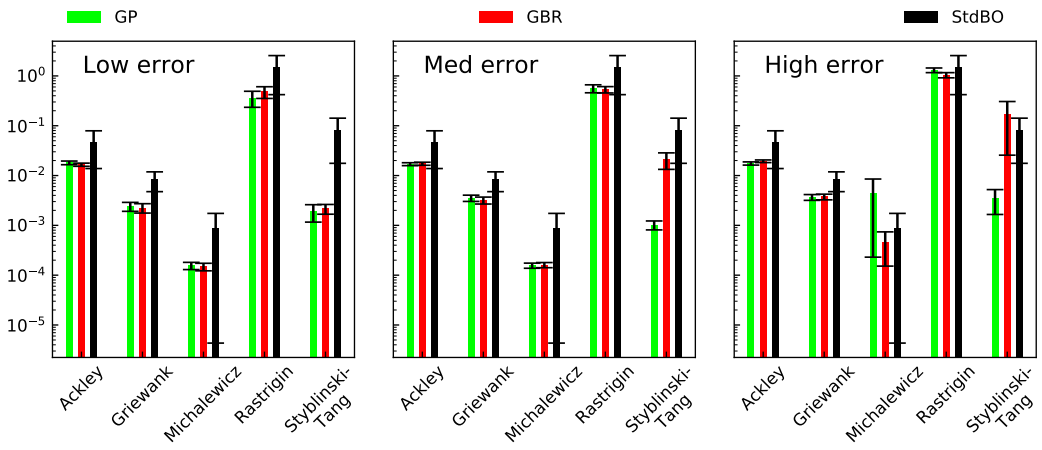

Figure 4.17: Comparison between final average **predicted** regrets using a GP or a GBR surrogate model for the discrepancy. The results are plotted for all three predictors' error regimes for all the benchmark functions and compared to standard BO.

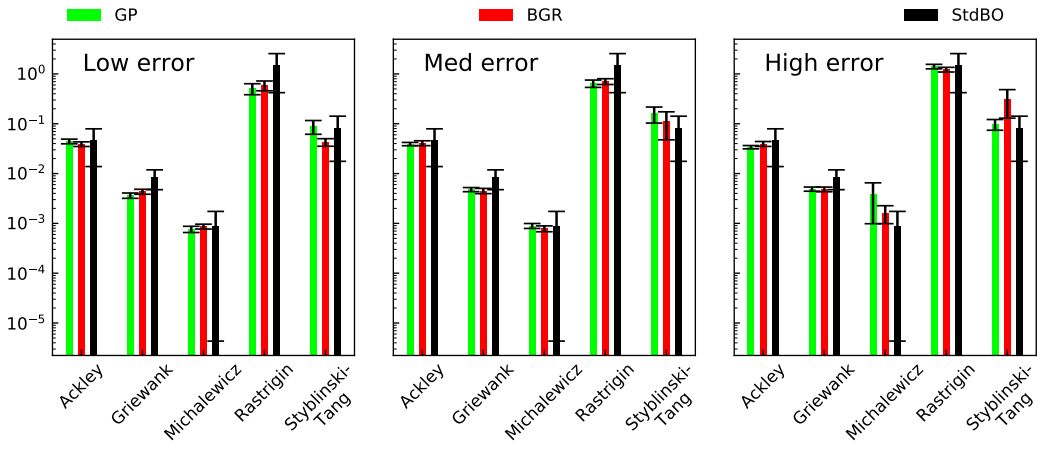

Figure 4.18: Comparison between final average **simple** regrets using a GP or a GBR surrogate model for the discrepancy. The results are plotted for all three predictors' error regimes for all the benchmark functions and compared to standard BO.

4.4.3 Discrepancy prediction method with GBR: conclusions

The analysis of the regrets discussed above shows that, on average, using a Gaussian process or a Gradient boosting regressor as predictive models for the discrepancy between the predictor and the actual objective function does not results in a significant difference in the performance of the discrepancy prediction method.

This leads to the conclusion that both models have an overall comparable predictive accuracy. Typically gradient boosting regressors have the capability to handle complex and high-dimensional data sets, but in our specific study they did not present advantages over Gaussian processes. We ascribe this to the fact that the data sets used for our studies were low dimensional and the number of points was not very large, which might have lead the GBR model to overfitting, outweighing potential benefits. The results of our experiments show that using a GBR surrogate model to predict the discrepancy is not a particularly encouraging method. Despite this we would not rule out the use of ensemble learning models when the discrepancy prediction method is applied to materials design optimisation tasks. Indeed, as it will be shown in Chapter 5, this type of problems are typically defined in search spaces with dimensionality much higher than 2. Furthermore overfitting can be mitigated by applying regularisation. For example if the weak learners are decision trees as in our experiments, Lasso regularization [39, 104] could be applied to the weights of the trees. Alternatively, more recent and effective regularisation methods, like DART [108], could be employed.

Other ensemble learning algorithms could also be explored, like the extreme gradient boosting regressor [20], or LightGBM [55], which have proven to be more robust than traditional Gradient Boosting Regressor against overfitting [92]. We leave the investigation of these options for future work.

4.5 Discrepancy prediction method with augmented real data

This method aims to improve the original discrepancy prediction method by augmenting the training data set of the surrogate model for the discrepancy $\mathcal{M}_\delta$. However this requires extra evaluations of the objective function f, which is contrary to our overall goal. A compromise can be reached by acquiring extra real observations only at some optimisation steps, for example according to a probabilistic rule. For this work a random number $\beta \in [0, 1]$ is sampled according to a uniform distribution, and extra observations are acquired only if n is below a given threshold, T. The criterion to select the location where to evaluate f is related to the location of the best point at the current step t, $\mathbf{x}_s^{\text{best}}(t)$, as defined in Eq. (4.1). Ten points are randomly sampled from the search space according to a uniform distribution, within a square of side 2λ, centered at $\mathbf{x}_s^{\text{best}}(t)$. Finally, out of these ten points only the farthest from $\mathbf{x}_s^{\text{best}}(t)$ is selected.

The rationale behind this policy, which we call *FarNeighbour*, is to improve the prediction of the discrepancy in the portions of the search space with high probability to find the optimum: the idea is that, by spending the budget of real observations in a limited region, the predictive accuracy of the surrogate model of the discrepancy becomes particularly good in that region. We expected this to be increasingly beneficial as optimisation proceeds, as BO is more and more likely to focus in proximity of the optimum. In the meantime we want to avoid sampling too close to previous observations, which would lead to redundancy. The parameters of this method are the threshold T and λ. The FarNeighbour sampling method is described in the Algorithm 5 below.

The full algorithm of the discrepancy prediction method with augmented real data is described in Algorithm 6.

4.5.1 Experimental results

Experimental settings

For the parameter λ five values were tested, so that $\lambda/l = \{0.1,\ 0.2,\ 0.3,\ 0.4,\ 0.5\}$, while three values have been chosen for the threshold T, $T = \{-1, 0.3,\ 0.5\}$ corresponding respectively to 0%, 30% and 50% probability to acquire one extra observation at each BO step.

The last value of λ, $\lambda = l/2$, implies that the ten points can be selected anywhere in the search space, disregarding the location of $\mathbf{x}_s^{\text{best}}$ completely. Furthermore the same method has been implemented using both a Gaussian process and a gradient boosting regressor as the surrogate models. It is important to notice that in all

Algorithm 5: FarNeighbour

Input: Current best point $\mathbf{x}_s^{\text{best}}(t)$, sampling threshold T, sampling distance λ, number of points to sample N_{extra}

Output: Point $\mathbf{x}^{\text{extra}}$

Sample a number $\beta \in [0,\ 1]$ randomly from a uniform distribution.

if $\beta < T$ **then**

  Sample N_{extra} points $\{\mathbf{x}_1, \dots \mathbf{x}_{N_{\text{extra}}}\}$ from a uniform distribution, s.t. $\mathbf{x}_{j,d} \in [\mathbf{x}_d^{\text{best}}(t) - \lambda, \mathbf{x}_d^{\text{best}}(t) + \lambda]$ where $d = \{1,\ 2\}$ is the spatial component of the space.

  Select the point $\mathbf{x}^{\text{extra}}$ as:

$$\mathbf{x}^{\text{extra}} = \underset{\mathbf{x} \in \{\mathbf{x}_1, \dots \mathbf{x}_{N_{\text{extra}}}\}}{\arg\max} d(\mathbf{x}, \mathbf{x}^{\text{best}}(t))$$

end

experiments the extra points acquired around the best point so far are counted against the maximum number of real observations. Thus the number of real data points in the discrepancy prediction method with augmented data is actually the same as for all the other methods, i.e. 80.

Results

As for the previous methods, also for the discrepancy prediction with augmented real data the average predicted and simple regrets have been computed.

Convergence curves observed with all the variants of this method are plotted in Figures 4.19 and 4.20 for simple regrets, and $T = 0.3$: they show respectively the results achieved using a GP or a GBR as surrogate model for the discrepancy. The converge curves for the other experimental settings are provided in Appendix B, while Table 4.5 and Table 4.6 report a summary of the final average regrets for the original discrepancy prediction method, for the discrepancy prediction method with GBR, and for the discrepancy prediction method with augmented real data. For this last one, the tables show the best combination of experimental settings that give the best results, i.e. the best choice of the surrogate model for the discrepancy, $\mathcal{M}_\delta$, as well as the best values for the parameters T and λ/l. The values reported in both tables indicate that the variant of the discrepancy prediction method with augmented real data is the most performant in majority of the experiments, for appropriate settings of $\mathcal{M}_\delta$, T, and λ/l. In particular we observe that the choice of a Gaussian process as surrogate model $\mathcal{M}_\delta$ is preferable for the high error regime.

Algorithm 6: Discrepancy prediction method with augmented real data

Input: Predictor p, number of real observations N_{obs}, number of points to sample N_{extra}, sampling threshold T, sampling distance λ

Output: Optimum point $\mathbf{x}_{opt}$

Initialize the real points data set $\mathcal{R}_0$, the predicted points data set $\mathcal{P}_0$, the discrepancy data set $\mathcal{C}_0$ and the set $\mathcal{S}_0 = \mathcal{R}_0 \cup \mathcal{P}_0$

for $i \leftarrow 0, 1, \ldots$ **do**

  Update the surrogate model $\mathcal{M}$ using the data in $\mathcal{S}_i$.

  Select a new point $\mathbf{x}_{i+1}$ by optimizing the acquisition function a: $\mathbf{x}_{i+1} = \arg\max_{\mathbf{x}\in\mathcal{D}} a(\mathbf{x}|\mathcal{D}_i)$

  Select the input point for the extra real observation: $\mathbf{x}_i^{\text{extra}} \leftarrow$ FarNeighbour

  Query f to obtain $\mathbf{y}_{i+1} = f(\mathbf{x}_{i+1}),\ \mathbf{y}_{i+1}^{\text{extra}} = f(\mathbf{x}_{i+1}^{\text{extra}})$

  Create a new real data set $\mathcal{R}_{i+1} = \mathcal{R}_i \cup \left\{(\mathbf{x}_{i+1}, \mathbf{y}_{i+1}), \left(\mathbf{x}_i^{\text{extra}}, \mathbf{y}_{i+1}^{\text{extra}}\right)\right\}$

  Set $\mathcal{C}_{i+1} = \mathcal{C}_i \cup ((\mathbf{x}_{i+1}, f(\mathbf{x}_{i+1}) - p(\mathbf{x}_{i+1}))) \cup (\mathbf{x}_{i+1}^{\text{extra}}, f(\mathbf{x}_{i+1}^{\text{extra}}) - p(\mathbf{x}_{i+1}^{\text{extra}}))$, and then retrain $\mathcal{M}_\delta$ on $\mathcal{C}_{i+1}$

  Create $\mathcal{P}_{i+1} = \{(\mathbf{x},\ p(\mathbf{x}) + \hat{\delta}(\mathbf{x}))\ :\ \mathbf{x}$ is a point in $\mathcal{P}_0\}$ where $\hat{\delta}$ is the predicted discrepancy.

  Set $\mathcal{S}_{i+1} = \mathcal{R}_{i+1} \cup \mathcal{P}_{i+1}$

  if *number real observations* $= N_{obs}$ **then**

    break

  end

end

On the other hand, combining a GBR with the discrepancy prediction method with augmented real data generally gives lower or comparable predicted regrets with respect to the the original method. Final simple and predicted regrets are also plotted respectively in Figures B.7 to B.8 and in Figures B.9 to B.10, leading to analogous conclusions.

Unfortunately there is no evident trend between the experimental settings and the performance of the discrepancy prediction with augmented data.

This analysis however is partial, as it focuses on the best final regrets only. We can have a broader overview of the experimental results form the converge curves plotted in Figure 4.19, in Figure 4.20 and in in Figures B.1 to B.6. These show that, for both simple and predicted regrets, the fastest convergence is obtained with the original discrepancy prediction method, displayed with a black dashed line across all the graphs. Even when the version of this method with augmented real data reach the lowest final regret, it shows noticeably slower convergence compared to the

original method at previous stages of Bayesian optimisation.

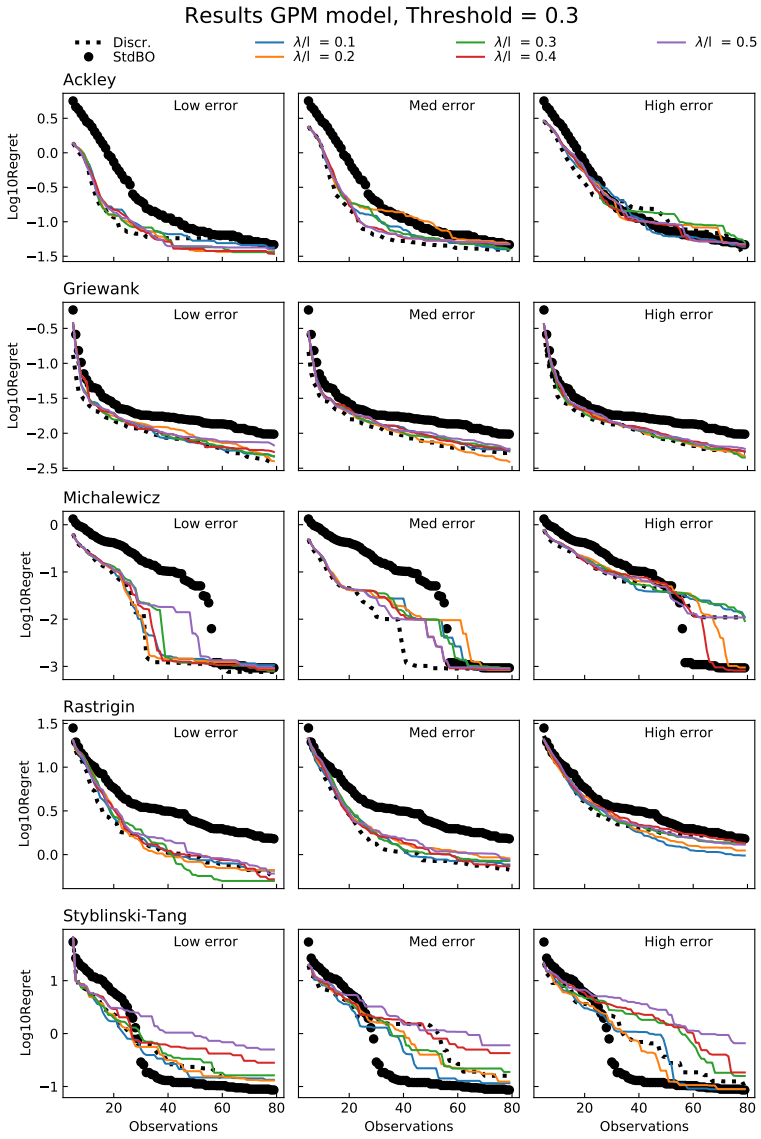

Figure 4.19: Average **simple** regrets for all the benchmark functions, using a **GP** surrogate model and a sampling threshold **T=0.3**. Each curve corresponds to a different value of the ratio λ/l. The black dotted curve refers to the discrepancy prediction method without any additional real data.

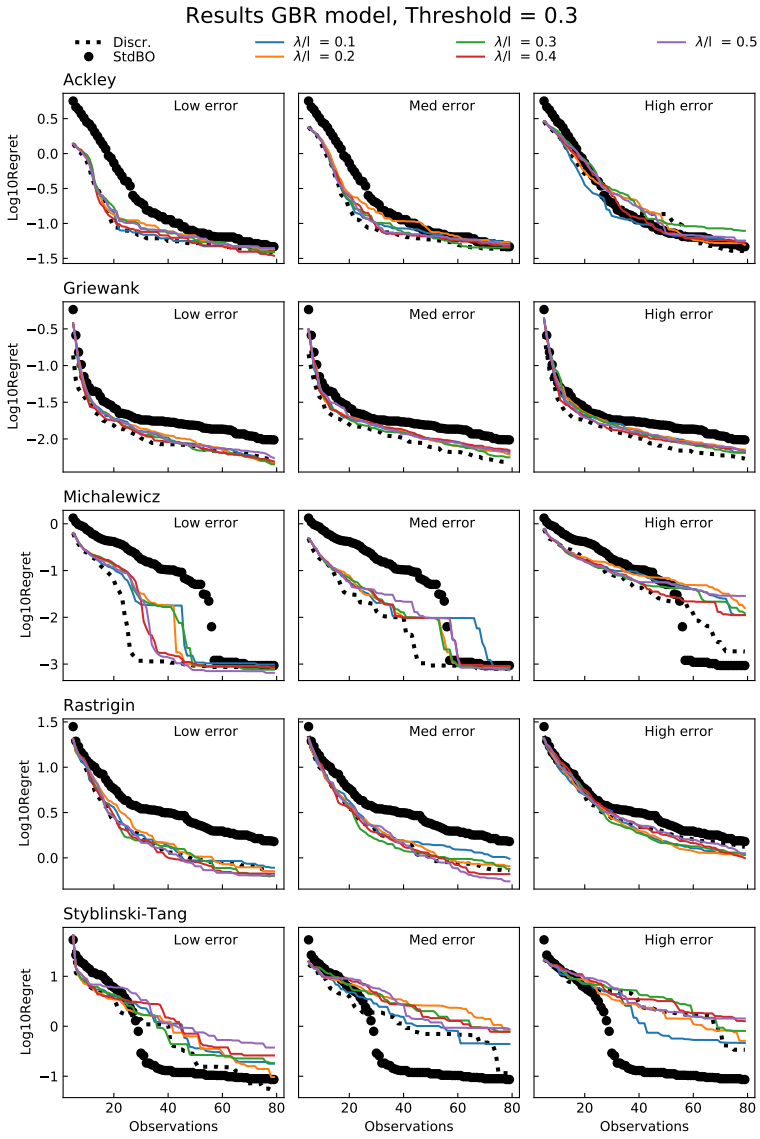

Figure 4.20: Average **simple** regrets for all the benchmark functions, using a **GBR** surrogate model and a sampling threshold **T=0.3**. Each curve corresponds to a different value of the ratio λ/l. The black dotted curve refers to the discrepancy prediction method without any additional real data.

LOW ERROR						
Function	$\mathbf{R_{discr}}$	$\mathbf{R_{GBR}}$	$\mathbf{R^{best}_{agument}}$	**Method**	**T**	$\boldsymbol{\lambda}$**/l**
Ackley	**3.17E-02 ± 4.45E-03**	4.10E-02 ± 5.65E-03	3.23E-02 ± 3.24E-03	GPM	0.3	0.3
Griewank	3.96E-03 ± 4.66E-04	**3.91E-03 ± 4.64E-04**	3.94E-03 ± 4.46E-04	BGR	0.3	0.2
Michalewicz	8.05E-04 ± 1.01E-04	8.84E-04 ± 1.11E-04	**6.51E-04 ± 8.53E-05**	BGR	0.3	0.5
Rastrigin	8.61E-01 ± 1.76E-01	6.72E-01 ± 1.13E-01	**4.67E-01 ± 9.11E-02**	GPM	0.3	0.3
Styb.-Tang	8.92E-02 ± 2.07E-02	1.10E-01 ± 3.64E-02	**8.64E-02 ± 1.61E-02**	GPM	0.5	0.1
MEDIUM ERROR						
Function	$\mathbf{R_{discr}}$	$\mathbf{R_{GBR}}$	$\mathbf{R^{best}_{agument}}$	**Method**	**T**	$\boldsymbol{\lambda}$**/l**
Ackley	4.22E-02 ± 3.36E-03	4.18E-02 ± 3.42E-03	**3.79E-02 ± 2.80E-03**	GPM	0.5	0.5
Griewank	4.26E-03 ± 5.18E-04	4.47E-03 ± 5.54E-04	**3.75E-03 ± 4.89E-04**	GPM	0.3	0.2
Michalewicz	8.48E-04 ± 1.11E-04	7.88E-04 ± 9.58E-05	**7.39E-04 ± 7.52E-05**	BGR	0.3	0.5
Rastrigin	1.02E+00 ± 1.73E-01	1.13E+00 ± 1.94E-01	**5.20E-01 ± 7.29E-02**	BGR	0.3	0.5
Styb.-Tang	1.54E-01 ± 2.25E-02	**4.12E-02 ± 1.48E-02**	9.07E-02 ± 1.68E-02	GPM	0.5	0.2
HIGH ERROR						
Function	$\mathbf{R_{discr}}$	$\mathbf{R_{GBR}}$	$\mathbf{R^{best}_{agument}}$	**Method**	**T**	$\boldsymbol{\lambda}$**/l**
Ackley	**3.74E-02 ± 2.88E-03**	4.94E-02 ± 4.60E-03	4.05E-02 ± 2.95E-03	GPM	0.3	0.1
Griewank	4.69E-03 ± 5.02E-04	4.47E-03 ± 4.75E-04	**4.20E-03 ± 4.53E-04**	GPM	0.3	0.3
Michalewicz	9.73E-04 ± 1.15E-04	1.31E-03 ± 4.26E-04	**8.13E-04 ± 1.01E-04**	GPM	0.3	0.4
Rastrigin	1.20E+00 ± 1.11E-01	1.62E+00 ± 1.65E-01	**8.39E-01 ± 6.46E-02**	GPM	0.5	0.1
Styb.-Tang	**7.68E-02 ± 1.59E-02**	7.50E-01 ± 3.50E-01	8.87E-02 ± 1.45E-02	GPM	0.3	0.1

Table 4.5: Comparison between the final **simple** average regrets obtained with the original discrepancy prediction method, the discrepancy prediction method with gradient boosting regressor, and the version of the same method with augmented real data. The regrets for each of these methods are indicated respectively with R_{discr}, R_{GBR} and $R^{\text{best}}_{\text{agument}}$. For this last one, only the best results are displayed, and the corresponding choice of the surrogate model for the discrepancy is reported in the column Method, where the values GPM or GBR indicate a Gaussian process or a gradient boosting regressor respectively. The values of the parameters T and λ/l giving $R^{\text{best}}_{\text{agument}}$ are also shown in the columns T and λ/l.

LOW ERROR						
Function	$\mathbf{R_{discr}}$	$\mathbf{R_{GBR}}$	$\mathbf{R^{best}_{agument}}$	**Method**	**T**	$\boldsymbol{\lambda}$**/l**
Ackley	1.75E-02 ± 1.48E-03	1.96E-02 ± 1.53E-03	**1.64E-02 ± 1.33E-03**	BGR	0.3	0.1
Griewank	2.31E-03 ± 4.93E-04	2.38E-03 ± 4.88E-04	**8.12E-04 ± 2.73E-04**	BGR	0.3	0.2
Michalewicz	1.47E-04 ± 2.58E-05	**1.43E-04 ± 2.50E-05**	1.49E-04 ± 2.25E-05	BGR	0.3	0.5
Rastrigin	7.27E-01 ± 1.59E-01	5.06E-01 ± 1.06E-01	**4.16E-01 ± 9.05E-02**	GPM	0.3	0.3
Styb.-Tang	**1.13E-03 ± 2.91E-04**	7.52E-03 ± 3.95E-03	1.44E-03 ± 2.99E-04	GPM	0.3	0.1
MEDIUM ERROR						
Function	$\mathbf{R_{discr}}$	$\mathbf{R_{GBR}}$	$\mathbf{R^{best}_{agument}}$	**Method**	**T**	$\boldsymbol{\lambda}$**/l**
Ackley	**1.72E-02 ± 1.19E-03**	1.77E-02 ± 1.08E-03	1.82E-02 ± 1.25E-03	BGR	0.3	0.5
Griewank	2.98E-03 ± 4.95E-04	3.69E-03 ± 5.13E-04	**1.36E-03 ± 3.79E-04**	GPM	0.3	0.2
Michalewicz	1.58E-04 ± 2.08E-05	1.59E-04 ± 1.91E-05	**1.38E-04 ± 1.55E-05**	BGR	0.3	0.2
Rastrigin	8.45E-01 ± 1.65E-01	8.74E-01 ± 1.86E-01	**4.45E-01 ± 7.27E-02**	BGR	0.3	0.5
Styb.-Tang	**1.33E-03 ± 3.28E-04**	7.67E-03 ± 3.28E-03	1.70E-03 ± 3.11E-04	GPM	0.5	0.1
HIGH ERROR						
Function	$\mathbf{R_{discr}}$	$\mathbf{R_{GBR}}$	$\mathbf{R^{best}_{agument}}$	**Method**	**T**	$\boldsymbol{\lambda}$**/l**
Ackley	**1.80E-02 ± 1.16E-03**	2.00E-02 ± 1.12E-03	1.91E-02 ± 1.10E-03	GPM	0.3	0.1
Griewank	4.08E-03 ± 4.86E-04	3.34E-03 ± 4.79E-04	**2.22E-03 ± 4.21E-04**	GPM	0.3	0.3
Michalewicz	1.94E-04 ± 3.35E-05	**1.88E-04 ± 4.25E-05**	2.46E-04 ± 3.93E-05	GPM	0.3	0.4
Rastrigin	1.15E+00 ± 1.10E-01	1.29E+00 ± 1.58E-01	**5.78E-01 ± 7.60E-02**	BGR	0.3	0.1
Styb.-Tang	2.70E-03 ± 1.48E-03	4.14E-01 ± 2.12E-01	**1.83E-03 ± 3.84E-04**	GPM	0.3	0.1

Table 4.6: Comparison between the final **predicted** average regrets obtained with the original discrepancy prediction method, the discrepancy prediction method with gradient boosting regressor, and the version of the same method with augmented real data. The regrets for each of these methods are indicated respectively with R_{discr}, R_{GBR} and $R^{\text{best}}_{\text{agument}}$. For this last one, only the best results are displayed, and the corresponding choice of the surrogate model for the discrepancy is reported in the column Method, where the values GPM or GBR indicate a Gaussian process or a gradient boosting regressor respectively. The values of the parameters T and λ/l giving $R^{\text{best}}_{\text{agument}}$ are also shown in the columns T and λ/l.

4.5.2 Discrepancy prediction method with augmented real data: Conclusions

The application of the discrepancy prediction method with augmented real data proved to be more performant than the original one only in terms of final average regret, although bar plots in Figure B.10 show that using this method, i.e. applying the FarNeighbour policy is beneficial in majority of the experiments if a GBR is used as a model to predict the discrepancy, and if we consider the predicted regrets only.

One possible drawback of this variant is the fact that sampling data in vicinity of the current best point privileges exploitation over exploration compared to the original method, where the policy to query the next point is dictated by the acquisition function alone. This potentially deteriorates the overall optimisation process.

In order to overcome this issue we decided to skip the sampling of extra points for the first 10 optimisation step. Additionally we tuned the parameter λ/l which determines how close to $\mathbf{x}_s^{\text{best}}(t)$ the extra points are going to be sampled. However no discernable trend con be determined for the impact of this parameter on the performance of the method.

4.6 Summary and Final Conclusions

In this chapter we described several variants for both the exclusion radius method and the discrepancy prediction method. Each variant has been designed for the purpose of improving the way of dealing with the inaccuracy of the predicted data.

The exclusion radius methods For the exclusion radius method the main goal was to explore better criteria to keep the predicted data as long as it is informative for the surrogate model and to discard it when it is detrimental. We designed two heuristic strategies to discard all the predicted data before the end of the optimisation process: the early switch method is a naive strategy were all the remaining predicted points are discarded at once at a pre-establish step, which is a parameter of the method. Although this simple approach gives better results than the original exclusion radius method for some benchmark functions for appropriate values of the radius r and of the switch iteration t_{sw}, overall it is always possible to find a radius for which exclusion radius method performs the best. Furthermore, the result we observe do not allow us to establish a criterion to select r and t_{sw}.

This outcome was not surprising due the simplicity of this approach, which was meant as an exploratory precursor for the adaptive radius exclusion method. Nevertheless we analysed the behaviour of the surrogate model before and after the predicted data was completely eliminated, by monitoring the changes in its predictive accuracy. Indeed a sharp decrease in the predictive accuracy of the GP surrogate model was observed immediately after the switch to standard BO.

As this was ascribed to the abrupt deletion of the predicted points, in the adaptive radius exclusion method we delete them progressively by increasing the radius by a constant amount after each iteration, except for the first 10, when few real data is available yet. Similarly to the early switch method however, we aim to remove all the predicted data before the budget of iterations is reached: the specific step at which this happens is an hyper parameter of the adaptive radius exclusion method, and the increase rate of the radius is set accordingly.

The results summarised in Tables A.10 to A.9 show that the original exclusion radius method achieves better or comparable final regrets, if an appropriate value r is selected, with the only exception of the Styblinski-Tang function. An analysis of the predictive accuracy of the surrogate model before and after the complete deletion of all the predicted points, similar to the one carried on for the early switch method, revealed a similar behaviour. The same analysis performed on the results obtained by keeping r fixed did not reveal any analogous drop in accuracy.

The main difference between the exclusion radius method and the adaptive radius method is that in the first one it is possible to keep a portion of the predicted points during all the optimisation process, if it happens that the acquisition function leaves

large portions of the search space unexplored, while in the second one the radius r is forced to increase so that at a given step all the predicted data is lost. Potentially this can leave regions of space completely depleted of data points, which explains the deterioration of predicted accuracy.

Although we were concerned that the presence of predicted points could hinder the performance of the adaptive radius exclusion method, the study reported in this chapter leads to the conclusion that generally forcing their removal is not advisable.

The discrepancy prediction methods In Section 4.4 we explored the performance of the discrepancy prediction method using a Gradient boosting regressor as a predictive method for the discrepancy. This approach is motivated by the fact that GBR is an ensemble method known for its capacity to model complex data sets. The hope was that, if the discrepancy between predictor and the actual objective is significantly different in different portions of the space, a GBR model can capture this complexity. The results show that GBR and GPs provide comparable performance for the functions we investigated. As the main advantage of GBR over GPs is the capacity to deal with high dimensional data, we hypothesize that the reason why we cannot discern any significant difference between the two configurations is the low dimensionality of our data.

In Section 4.5 we introduce an additional criterion to select the training points for the surrogate model for the discrepancy: a stochastic policy determines if, at the current optimisation step, a point within a given distance from the best point so far is added to the data set instead of the point suggested by the acquisition function. The average frequency at which this happens is given by the threshold parameter T.

The idea is to improve the predictive accuracy for the discrepancy in a region where we are most likely to find the optimum. A comparison between the experimental results obtained with the discrepancy prediction method with and without augmented data revealed that, although for majority of the benchmark functions it is possible to find experimental settings for this last version of the method, generally the convergence observed with augmented data is poor compared to the original discrepancy method. Furthermore, both simple and predicted regrets showed slightly higher values when we set $T = 0.5$ instead of $T = 0.3$. All these observations lead to the conclusion that introducing the FarNeighbour sampling method is detrimental in this configuration.

On the other hand the results obtained with augmented data and with a GBR as a predictive model for the discrepancy showed lower predicted regrets compared to the discrepancy prediction method without augmented real data. Simple regrets however were lower for the original method.

The overall conclusion is that this last version of the discrepancy prediction

method is better than the original only if a GBR is used and only if the best point at each step is computed from the predictive mean of the surrogate model of the objective function, as formalised in Eq. (4.2).

Finally, for all the other methods, we conclude that the original versions proposed in Chapter 3 perform generally better and are more straightforward to use as they depend on fewer hyper parameters.

Chapter 5

Batched Bayesian Optimization for Molecular Structures

5.1 Introduction

Batch Bayesian optimisation (BBO) is widely employed to accelerate discovery in the physical sciences [35, 43, 63, 74]. Indeed BBO allows us to benefit from the acquisition of multiple real observations, which is possible in majority of the modern experimental setups. When the objective function is evaluated via computations, multiple function evaluations are achieved via parallel computation.

However, the application of BBO to the design of experiments presents several challenges. As mentioned in Chapter 2, Section 2.4, the choice of the *batching policy* is crucial. Ideally this should maximise the informativeness of the sampled points while also avoiding redundancy, but the design of batching policies that achieve this balance is still an active topic of research. Furthermore, in problems arising from the design of experiments, the objective function is frequently defined in a discrete space: for example if the objective function is a physical or chemical property resulting from molecules and their structure, the search space is often discrete, as molecules are inherently discrete entities. This represents a further challenge for BBO: while the acquisition functions designed for sequential BO and introduced in Chapter 2 can deal with discrete variables, the acquisition functions proposed in literature for BBO are mostly designed to work in continuous search space only [37, 94, 116]. The development of batching policies for BBO in a discrete space has been so far relatively little investigated.

One state-of-the-art batching approach for discrete spaces is the LAW2ORDER method, proposed by Oh et. al. [79] to solve permutation problems. The batching policy of this method is based on *determinantal point processes* [61] (DPPs). As

we will describe further in Section 5.2.2, batching policies based on DPPs have the advantage of being suitable for discrete spaces in addition to guaranteeing diversity among the batch points.

However, these BBO approaches relying on DDPs present the drawback that the selection of the batch points is driven by the maximisation of diversity only, while the informativeness of the points, and their utility with respect to the optimisation problem under consideration are not taken into account.

In order to address this issue, the LAW2ORDER method uses a sampling strategy that combines DDPs with a sequential acquisition function, which chooses the next point to evaluate based on its informativeness. Specifically, the batch points are selected by maximising a quantity, called the *acquisition weighted kernel*, which is defined as the product of two terms. One is the posterior variance of the GP surrogate model, which promotes diversity among the batch points. The other term depends on the acquisition function, and thus it encourages exploitation. The contribution of each of these terms to the batching policy can be tuned explicitly by a weighting function. A more detailed description of LAW2ORDER is provided in Section 5.2.3.

Application of LAW2ORDER to materials design. The work described in this chapter aims to investigate the applicability of LAW2ORDER to the field of materials design. This choice is motivated by the consideration that both permutation problems and optimisation of materials' properties deal with extensive and discrete spaces. For this purpose we studied four distinct materials' properties optimisation problems on three different data sets of small organic molecules.

Using the LAW2ORDER BBO method on a chemical data set requires us to adapt the original method in two ways: first an appropriate numerical representation of the input space must be chosen, secondly a suitable kernel function must be used. The input space here is the set of molecules composing the whole search domain, and each single molecule is defined by the set of its atoms and their relative spatial coordinates.

In order to be usable for BO, this raw data representation must be transformed into a mathematical representation that can be fed into a Gaussian process. This is possible using *chemical descriptors*, which generate a numerical encoding of molecular properties and which will be introduced in Section 5.3. Molecular descriptors must be valid input data for the specific similarity kernel used to define the GP. This means that, in general, the choices of descriptors and kernels are not independent.

Here two combinations of descriptors and similarity kernels have been used: the same set of experiments has been repeated using Coulomb descriptors [73, 91] combined with an RBF kernel [115] and SOAP kernel descriptors combined with a regularized entropy match kernel (REMatch) [23] respectively. The REMatch kernel is

specifically designed to quantify similarities between molecules, described by a SOAP kernel.

All the experimental results have been compared to the performance achieved using *Thompson Sampling* (TS) as a batching policy, as this method has previously been successfully applied in this area [43, 53]. Our results indicate that LAW2ORDER is always competitive with Thompson sampling across all of the domains, with significant advantages observed in some domains. Our results also indicate that Coulomb descriptors significantly outperform SOAP descriptors across all of our experimental domains, both in terms of computational cost and in the performance of BBO.

Structure of the chapter. The content of this chapter is structured as follows:

1. In Section 5.2 we provide a short description of the previous work more closely related to our work:
   (a) In Section 5.2.1 we introduce BBO based on Thompson sampling method. This approach has been used for optimisation tasks related to Materials Science and we also use it as a benchmark against which to compare the performance of the LAW2ORDER.
   (b) In Section 5.2.2 we introduce DPPs, explaining why they encourage diversity when used to sample points.
   (c) Section 5.2.3 provides a description of the LAW2ORDER BBO method, as originally proposed by Oh et al [79].
2. In Section 5.3 we introduce the concept of *chemical descriptors*, mathematical entities that encode chemical and structural properties of molecules, which are the input variables in our BBO experiments. After giving a general overview, we describe the two types of descriptors that we use for the experiments, Coulomb matrices and SOAP kernel descriptors.
   (a) Coulomb descriptors are treated in Section 5.3.1.
   (b) Section 5.3.2 illustrates the main concepts related to the SOAP descriptors and the REMatch similarity kernel. This last one is specifically designed to quantify similarity between molecules when these ones are represented by SOAP descriptors.
3. In Section 5.4.1 the specific molecular data sets used for the experiments are described, together with the related optimisation tasks.
4. Results are reported in Section 5.5.

5.2 Previous work

Here we give a short overview of methods employing batching policies based on stochastic sampling, which are the most closely related to our work. Specifically, we focus on *Thompson sampling BBO* (TS-BBO), which is also used as a benchmark method, and the *BBO based on determinantal point processes*, which was the precursor of the *LAW2ORDER* method. A more general overview of BBO methods for discrete spaces is given in Section 2.4.

5.2.1 Thompson sampling BBO: TS-BBO

Thompson sampling (TS) is a randomized algorithm for choosing actions sequentially under uncertainty, originally designed to solve multi-armed bandit problems [3, 103]. More recently TS has been successfully employed as a batching policy for BBO, in the fields of Engineering, Medicine and Material Science [43, 53]. This approach, which will be referred here as *TS-BBO*, consists in drawing a function g from the surrogate model.

The function g, which is one of the possible values of the actual objective, is used to populate the batch, by selecting from the search space the input point that optimises it. This process is repeated k times, where k is the size of the batch. Typically the surrogate model is a GP, in which case $g \sim \mathcal{GP}(\boldsymbol{\mu}, K_{\text{post}})$, where $\boldsymbol{\mu}$ and K_{post} are respectively the predictive mean and posterior covariance matrix of the GP. The objective function is then evaluated at these k points, and BBO proceeds as for sequential BO (see Algorithm 1). The batching policy based on TS is described in Algorithm 7.

TS is conceptually simple and computationally inexpensive. Importantly, it can be applied to both continuous and discrete search spaces. Due to the relatively wide use of TS-BBO in design of experiments problems [43, 53], we benchmark all our experimental results against this method.

TS-BBO enforces mainly exploitation, because the function g is sampled from a distribution which is insensitive to the variance of the observations $y = g(\mathbf{x})$, as the posterior variance K_{post} depends only on the input variables. The process of sampling g introduces a variance, and by virtue of this exploration is introduced in TS-BBO . However, sampling methods designed to explicitly promote exploration and to allow us to control the extent at which the space is explored are generally more convenient for BBO. As will be shown soon, the approach that we applied satisfies these requirements.

Algorithm 7: Thompson Sampling Batching

Input: Available data set of size $|N|$, $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1,\dots,N}$, prior GP $\mathcal{GP}(0, K)$, batch size k

Output: Batch of k points $|B| = k$

Initialize batch $B = \emptyset$
Compute the posterior distribution $\mathcal{GP}_{\text{post}} = \mathcal{GP}(\boldsymbol{\mu}, K_{\text{post}})$
for $j \leftarrow 1$ **to** k **do**
    Draw a function $g(\mathbf{x}) \sim \mathcal{GP}(\boldsymbol{\mu}, K_{\text{post}})$
    g as $\mathbf{x}_j = \arg\min_{\mathbf{x}\in\mathcal{X}} g(\mathbf{x})$
    Update the batch B: $B := B \cup \{\mathbf{x}_j\}$
end

5.2.2 BBO based on Determinantal Point Processes (DPP-BBO)

A more diversity oriented BBO strategy is to use determinantal point processes (DPPs) to sample batch points. DPPs are stochastic processes giving the probability of sampling a subset S of the search space Y as proportional to the determinant of a symmetric, positive definite matrix L that measures pairwise similarities among the points of the subset itself. The matrix L is called *ensemble*. The probability distribution over the possible subsets of Y is given by:

$$P_L(S) = \frac{det(L)}{det(L+I)} = \frac{det(L)}{\sum_{S\subset Y} det(L_S)}. \tag{5.1}$$

The ensemble L is related to the similarity kernel among the points in S, K, via:

$$K = (L+I)^{-1}L. \tag{5.2}$$

Remarkably, the subsets with higher probability to occur are those that have higher diversity, as measured by L [61, 62]. Thus DPPs are particularly suitable for sampling diverse batches. As BBO usually requires that the batches have finite and constant size k, in order to use DDPs for BBO it is necessary to apply the restriction that the sampled sets have precisely k points:

$$P_L^k(S) = \frac{det(L_S)}{\sum_{|S'|=k} det(L'_S)}. \tag{5.3}$$

The most diverse batch of points is given by:

$$S^* = \arg\max_{S\subset\Omega_k} P_L^k(S). \tag{5.4}$$

where Ω_k is the set of all the subsets with size k in the search space. The process defined by Eq. (5.3) is called *k-DPP* [62]. The essential feature of DDP-BBO is the choice of the similarity kernel K, which determines the ensemble L. Kathuria and coauthors [54] proposed that we should use the posterior covariance matrix of the surrogate model, K_{post}, as the ensemble for the DPP: this has the advantage that L can be learned during the optimisation process, and does not need to be defined a priori. Thus the batch selection criterion becomes:

$$B = \underset{B \subset \Omega_k}{\arg\max} \det(K_{\text{post}}). \tag{5.5}$$

In the aforementioned work the authors initialize the batch via a sequential acquisition function: they compare results obtained with EST, introduced in Section 2.3.2, and upper confidence bound (UCB) [4, 6], while the following points are selected by solving Eq. (5.5). As this maximisation problem is NP hard, they proposed two methods to solve it: a greedy algorithm and a sampling one, and this last one was shown to give higher performance.

More recently, Wang et al. [112] and Nava et al. [77] proposed respectively Gibbs sampling and Thompson sampling to solve the maximisation problem (5.3).

5.2.3 LAW2ORDER

The DPP-BBO methods described above have, as a drawback, the fact that they focus on diversity, at expense of the quality of the batch points. LAW2ORDER aims to address this issue by defining a novel diversity gauge, called the *L-ensemble with Acquisition Weights* (LAW), which combines the posterior variance with a sequential acquisition function, a, for example expected improvement (EI). The explicit form of the LAW gauge is:

$$L^{\text{AW}}(\mathbf{x}_1, \mathbf{x}_2) = w(a(\mathbf{x}_1)) \cdot K_{\text{post}}(\mathbf{x}_1, \mathbf{x}_2) \cdot w(a(\mathbf{x}_2)). \tag{5.6}$$

The introduction of the acquisition function in the DPPs process has the advantage of adding local-exploitation in the selection of the batch points, alongside to the global-exploration promoted by the variance. Thus the determinant of the gauge L_S in Eq. (5.3) becomes

$$\begin{aligned} \det\left(\left[L^{\text{AW}}\right]_{i,j=1,\dots,k}\right) = \\ \det\left(\left[L(\mathbf{x}_i, \mathbf{x}_j)\right]_{i,j=1,\dots,k}\right) \prod_{i=1}^{k} w(a(\mathbf{x}_i))^2 = \\ \det\left(\left[K_{\text{post}}(\mathbf{x}_i, \mathbf{x}_j)\right]_{i,j=1,\dots,k}\right) \prod_{i=1}^{k} w(a(\mathbf{x}_i))^2 \end{aligned} \tag{5.7}$$

where k is the batch size, i and j start from 2 because the first batch point is selected simply by maximising the sequential acquisition function a. This shows that $\det(L^{\text{AW}})$ can be maximised increasing either the determinant of the posterior variance or the product term, or both in a balanced way. The impact of the acquisition function is regulated by the weight function $w(\cdot)$. The batch point returned at each iteration of the LAW batching policy is given by [79]:

$$\mathbf{x}_b = \arg\max_{\mathbf{x}\in\mathcal{X}} \log((K_{\text{post}}(\mathbf{x}|\{\mathbf{x}_i\}_{i=1,\dots,k-1}) \cdot w(a(\mathbf{x})^2). \tag{5.8}$$

The whole batching method is illustrated in Algorithm 8.

Algorithm 8: LAW batching method

Input: Weight function w, prior similarity kernel K to define the L_ensembles, acquisition function a, batch size k

Output: Batch B of k points, $|B| = k$

Initialize the data set $\mathcal{D}_0$.

Select the first batch point:

$$\mathbf{x}_1 = \arg\max_{\mathbf{x}\in\mathcal{X}} a(\mathbf{x})$$

for $b \leftarrow 2$ **to** k **do**

  $\mathbf{x}_b = \arg\max_{\mathbf{x}\in\mathcal{X}} \log((K_{\text{post}}(\mathbf{x}|\{\mathbf{x}_i\}_{i=1,\dots,b-1}) \cdot w(a(\mathbf{x}))^2$

end

5.3 Molecular descriptors and similarity kernels

Molecular descriptors give a mathematical representations of molecules' properties: the chemical information contained in a symbolic representation of a molecule is transformed into numerical values via algorithms. A formal definition of a molecular descriptor is provided by Todeschini and Consonni as [105]:

The molecular descriptor is the final result of a logic and mathematical procedure which transforms chemical information encoded within a symbolic representation of a molecule into a useful number or the result of some standardized experiment.

Molecular descriptors can be derived from measured physical and chemical properties, or built based on theoretical values of these properties. In both cases they must satisfy the following requirements [72]:

1. Being non-degenerate: two different molecules must have different descriptors.

2. Being invariant to atom labelling and numbering (permutation invariance).
3. Being invariant to spatial rotations and translations.
4. Being defined by an unambiguous algorithm.

In addition to being non degenerate, ideally a descriptor should allow us to distinguish between isomers, i.e. molecules with the same atomic composition but different geometric structure. Furthermore, in order to be useful, it should also have:

5. Structural interpretation.
6. Good correlation with at least one experimental property.

From point 6 it follows that different molecular descriptors can result in different levels of accuracy in the prediction of the objective function, depending on which property they are related to, and how relevant this property is to the optimisation problem at hand.

It is important to notice that the choice of a molecular descriptor partially constrains the availability of similarity kernel suitable for that particular descriptor, although for each descriptor more than one option is usually available. For example we will use Coulomb and SOAP kernel descriptors, introduced in Section 5.3.1 and in Section 5.3.2 respectively. For the first ones kernels commonly used for Gaussian regression like RBF or Matérn, are applicable, while this is not the case for SOAP descriptors, as these ones are not unidimensional vectors. For SOAP descriptors we used a custom kernel proposed in previous literature, called the *REMatch* kernel, briefly described in Section 5.3.2.

5.3.1 Coulomb descriptors and RBF kernels

A Coulomb matrix is a molecular descriptor which represents the electrostatic interaction between the nuclei of the atoms composing a molecule [73]. It is defined as:

$$M_{ij} = \begin{cases} 0.5 Z_i^{2.4} & \text{for } i = j \\ \frac{Z_i Z_j}{|\mathbf{R}_i - \mathbf{R}_j|} & \text{for } i \neq j \end{cases} \tag{5.9}$$

where Z_i and Z_j are the atomic charges of atoms i and j respectively, while the vectors $\mathbf{R}_i$ and $\mathbf{R}_j$ represent the positions of these atoms in the three dimensional space, and $|\mathbf{R}_i - \mathbf{R}_j|$ is the distance between the vectors $\mathbf{R}_i$ and $\mathbf{R}_j$[1]. The off-diagonal elements of the matrix defined by Eq. (5.9) encode the Coulomb repulsion between the nuclei of atoms i and j, while diagonal elements can be seen as the interaction of an atom with itself. The Coulomb matrix is invariant to translations and rotations

[1]In Eq. (5.9) the distance between two atoms is indicated with the operator $|\cdot|$, which is a common notation in physical sciences for the Euclidean norm, and it is equivalent to using $\|\cdot\|_2$

of the molecule in 3D space, but not invariant to permutations of the index of the atoms

Permutation invariance can be achieved by choosing the eigenspectrum $\mathbf{C}$ of the Coulomb matrix as a descriptor: $\mathbf{C} = (\lambda_1, \ldots, \lambda_d)$, where λ_i are the eigenvalues of the matrix defined in Eq. (5.9), and d is the number of columns, corresponding to the number of atoms present in the molecule. Another advantage of using the eigenspectrum representation is the reduction in the original dimensions. As mentioned before, Coulomb eigenspectrum descriptors have been used in combination with the radial basis function similarity kernel (RBF), defined as:

$$K(\mathbf{X}_i, \mathbf{X}_j) = \exp\left(-\frac{\|\mathbf{X}_i - \mathbf{X}_j\|_2^2}{2\sigma^2}\right). \tag{5.10}$$

5.3.2 SOAP descriptors and REMatch kernels

SOAP descriptors [9, 23] represent molecules in terms of *local environments*: each atom j in a given molecule M can be associated to a cluster of neighbours in the same molecule, which are within a predefined distance r_{cut} from j. The region containing this cluster is a local environment, here indicated as H_j, while r_{cut} is called the *cut-off radius*. The density of each atom in the environment H_j is approximated with a Gaussian function with variance σ^2 and centered in j. A local environment is characterised by the *local density of atoms*, which is the sum of the densities of all the atoms it contains, and which is given by:

$$\rho_H(\mathbf{r}) = \sum_{j \in H} \left(-\frac{(\mathbf{x}_j - \mathbf{r}_{\text{cut}})^2}{2\sigma^2}\right). \tag{5.11}$$

The SOAP descriptor for a molecule M is defined as the pairwise similarity kernel K^{SOAP} between all the local environments in M. Each element of the matrix K^{SOAP} is defined as the overlap of two environments, H_i and H_j, over all the 3D rotations, $\hat{\mathbf{R}}$:

$$K^{\text{SOAP}}_{i,j} = \int d\hat{\mathbf{R}} \left| \int \rho_{H_i}(\mathbf{r}) \rho_{H_j}(\hat{\mathbf{R}}\mathbf{r}) d\mathbf{r} \right|^2. \tag{5.12}$$

In order to make K^{SOAP} rotationally invariant, the density of atoms $\rho_H(\mathbf{r})$ is transformed using orthonormal functions based on by spherical harmonics $Y_{lm}(\mathbf{r})$ and radial basis functions $g_n(\mathbf{r})$. Thus Eq. (5.11) becomes:

$$\rho_H(\mathbf{r}) = \sum_{nlm} c_{nlm} g_n(|\mathbf{r}|) Y_{lm}(\hat{\mathbf{r}}). \tag{5.13}$$

With this transformation K^{SOAP} can be expressed as a function of the power spectrum of the local density of the environment H, $p(H)$. Remarkably, the power

spectrum is rotationally invariant and, for an environment with only two atoms with atomic numbers Z_1 and Z_2 respectively[2], it is given by:

$$p_{n,n',l}(H) = \pi \sqrt{\frac{8}{2l+1}} \sum_m c_{nlm}^{Z_1}(\mathbf{r}) c_{n'}^{Z_2}(\mathbf{r}). \tag{5.14}$$

The coefficients $c_{nlm}^{Z_i}$ are given by:

$$c_{nlm}^{Z_i} = \int\int\int_{\mathbb{R}^3} dV\, g_n(\mathbf{r}) Y_{lm}(\mathbf{r}) \rho^{Z_i}, \tag{5.15}$$

where $\rho^{Z_i} = \sum_i^{Z_i} e^{-\alpha|\mathbf{r}-\mathbf{R_i}|^2}$, Z_i and R_i being respectively the atomic number and the position of a generic atom.

By concatenating elements of the power spectrum into a unit vector $\hat{\mathbf{p}}(H)$, it is possible to express the SOAP kernel as the dot product:

$$K_{i,j}^{\text{SOAP}} = K^{\text{SOAP}}(H_i, H_j) = \hat{\mathbf{p}}(H_i) \cdot \hat{\mathbf{p}}(H_j) = \hat{\mathbf{p}}_i \cdot \hat{\mathbf{p}}_j \tag{5.16}$$

or, in its normalized form:

$$K^{\text{SOAP}}(\hat{\mathbf{p}}_i, \hat{\mathbf{p}}_j) = \left(\frac{\hat{\mathbf{p}}_i \cdot \hat{\mathbf{p}}_j}{\sqrt{\hat{\mathbf{p}}_i \cdot \hat{\mathbf{p}}_i \; \hat{\mathbf{p}}_j \cdot \hat{\mathbf{p}}_j}} \right)^{\xi}, \tag{5.17}$$

where ξ can be any positive integer. The vector $\hat{\mathbf{p}}(H)$ has the form:

$$\hat{\mathbf{p}}(H) = \{p_i\}_{i=1,\ldots,N},$$

where each p_i is given by Eq. (5.14).

The REMatch kernel

Differently from Coulomb descriptors, SOAP descriptors are not suitable to be used with the more common similarity kernels like RBF or Matérn kernels, because additional computational steps are needed to define a similarity measure between entire structures. A kernel specifically designed to compute the similarity between molecular structures using SOAP kernel descriptors is the *regularized entropy match kernel* (REMatch) kernel [23]. For two molecular structures with the same number of atoms N, the REMatch kernel is defined as:

$$K^{\gamma}(A, B) = \text{Tr}\mathbf{P}^{\gamma}\mathbf{C}(A, B) \tag{5.18a}$$

$$\mathbf{P}^{\gamma} = \underset{\mathbf{P} \in \mathcal{U}(N,N)}{\arg\min} \sum_{ij} P_{ij}(1 - C_{ij} + \gamma \ln P_{ij}) \tag{5.18b}$$

[2]The atomic number Z is equal to the number of protons in the nucleus of an atom. Z is used to uniquely identify chemical elements.

where $\mathbf{A}$ and $\mathbf{B}$ are two distinct molecular structures, $C(\mathbf{A}, \mathbf{B})$ is the covariance matrix between all the possible pairings of environments between the systems $\mathbf{A}$ and $\mathbf{B}$. Each element of $C(\mathbf{A}, \mathbf{B})$ is given by:

$$C_{i,j} = K^{\text{SOAP}}(H_i^A, H_j^B)$$

$\mathbf{P}$ is a $N \times N$ matrix that must satisfy the property that its rows and columns sum to $1/N$, i.e. $\sum_i P_{ij} = \sum_j P_{ij} = 1/N$. Furthermore, $\mathbf{P}$ must fulfil the constraint expressed by Eq. (5.18b), where $\mathcal{U}(N, N)$ is the set of all the $N \times N$ matrices whose rows and columns sum to $1/N$.

It is important to notice that the term $E(\mathbf{P}) = -\sum_i P_{ij} \ln P_{ij}$ in Eq. (5.18a) is an entropy term, which acts also as regularization term. Thus the parameter γ determines the contribution of information entropy to the kernel: for $\lambda \to 0$ the entropic penalty becomes negligible, with the consequence that only the best matching local environments are taken into account, while for $\lambda \to \infty$ all the environments contribute to $K(\mathbf{A}, \mathbf{B})$.

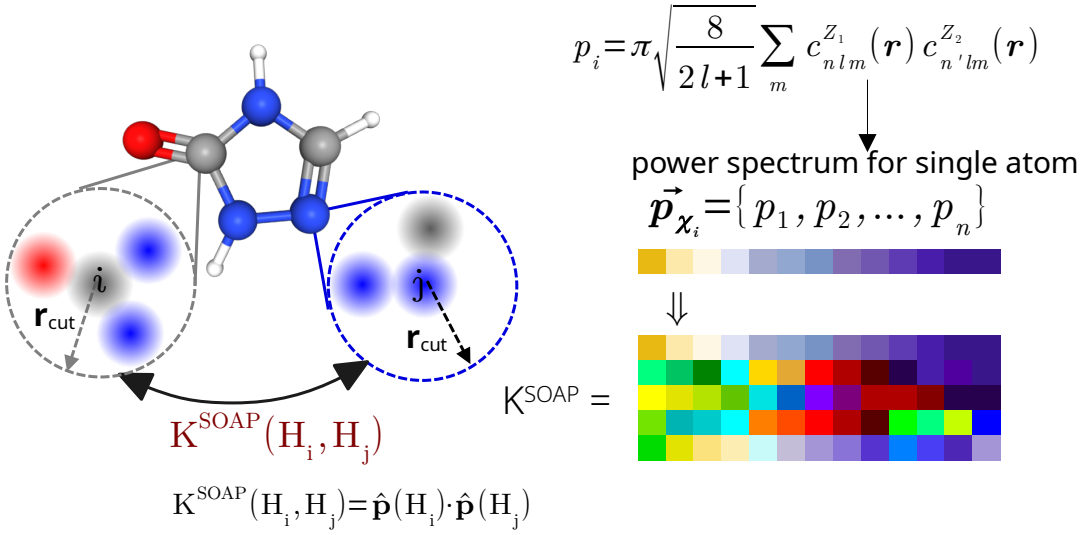

Figure 5.1: Illustration of how a SOAP kernel descriptor is built: on the left side an element of the K^{SOAP} matrix is built from the local environments of the atoms i and j, H_i and H_j respectively. On the right side it is shown the whole SOAP kernel as a matrix where each row corresponds to an atom in the molecule, and it corresponds to $\mathbf{p}$, introduced in Eq. (5.16). The vector $\mathbf{p}$ is given by the concatenation of the pairwise power spectra defined in Eq. (5.14).

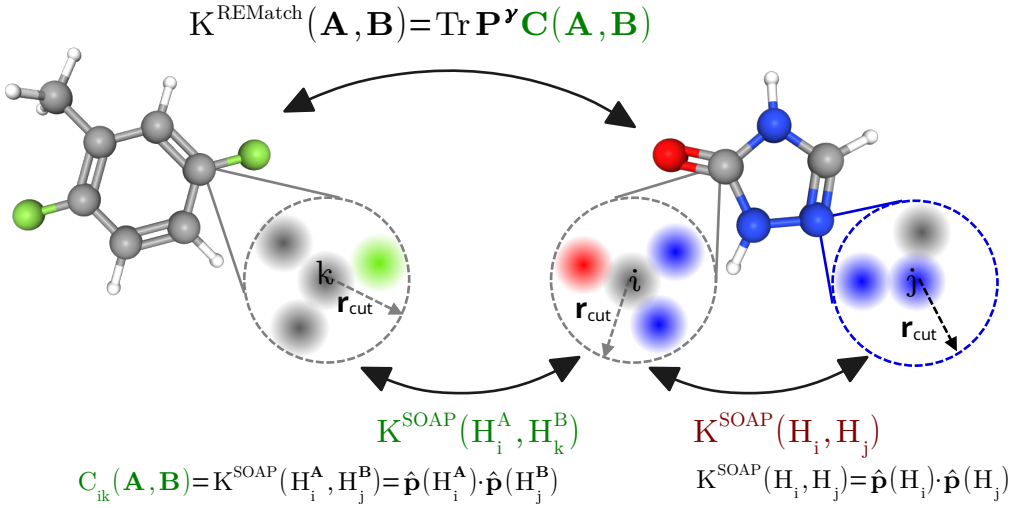

Figure 5.2: REMAtch kernel between two molecular structures, $\mathbf{A}$ and $\mathbf{B}$, which is related to the covariance matrix $\mathbf{C}(\mathbf{A},\mathbf{B})$ via Eq. (5.18a). Each element of $\mathbf{C}(\mathbf{A},\mathbf{B})$ is given by the SOAP similarity kernel between a local environment of $\mathbf{A}$ and a local environment of $\mathbf{B}$.

5.4 Experiments

The LAW2ORDER and the Thompson BBO methods have been evaluated on four optimisation tasks arising from three different publicly available data sets: QM7, QM7b and QM9. These data sets contain molecules' composition and geometric structure, along with their chemical properties.

QM7. This data set contains the atomization energies of 7,165 small organic molecules, computed via DFT simulations [91]. All the molecules are composed of four different atomic species: carbon, hydrogen, nitrogen and oxygen. The optimisation task for this data set consists of finding the molecule with the lowest atomization energy.

QM7b. The QM7b data set includes 14 properties of 7,211 small organic Here we focus on the energy gaps, defined as the difference between the *homo* and *lumo* energetic levels of the molecules: $\Delta E = E_{\text{homo}} - E_{\text{lumo}}$. The optimisation task is to find the closest value to a hypothetical optimum value, here arbitrarily set to be $E^{\text{opt}} = 3.5\text{Hartree}^3$. Thus our objective function is $f = \{\text{abs}(\Delta E^{\text{opt}} - \Delta E)\}$.

[3]Hartree is a unit of energy in the Hartree atomic units system, corresponding to $\approx 2625\ kJ\ mol^{-1}$, or $\approx 27\ eV$.

QM9. This data set contains energetic, electronic and thermodynamic properties for 133,885 small organic molecules, including polarizability, and homo and lumo energies [89]. We study two different optimisation problems over this data set:

1. **QM9polar:** maximize the polarizability of the molecule.

2. **QM9gaps:** optimisation of the electronic band gap $\Delta E = E_{\text{homo}} - E_{\text{lumo}}$.

Since the QM9 data set is significantly larger than the other two data sets, in order to have our experiments complete in a reasonable amount of time, reduced data sets have been generated for both the QM9polar and QM9gap tasks, containing 7,020 and 7,021 molecules respectively. The reduced data sets were constructed by selecting molecules from the whole QM9 data set in such a way that the distributions of the values of energy gaps and of polarizability approximate the respective original distributions as closely as possible. More information about this is given in Section 5.4.1.

Data set	**Points**	**Objective** f	**Task**
QM7	7,165	E_{atom}	min
QM7b	7,211	$\lvert\Delta E - 3.5 \text{ Hartree}\rvert$	min
QM9gaps	7,021	$\lvert\Delta E - 3.5 \text{ Hartree}\rvert$	min
QM9polar	7,021	E_{pol}	max

Table 5.1: A summary of our data sets.

5.4.1 Experimental setup

All the experimental results reported in this chapter have been achieved setting the batch size $k = 3$. This choice was imposed by the computational cost of the LAW2ORDER method with SOAP descriptors, and we selected equal values of k for Coulomb descriptors for fair comparison. Each experiment was initialized with 10 points, randomly selected from the search space according to a uniform distribution, excluding the 100 best molecules. Input variables have been L2 normalized, i.e. each input variable has been scaled so that its Euclidean norm is equal to 1.

LAW2ORDER parameters. For the LAW2ORDER method we used the expected improvement acquisition function, and, in keeping with the original work [79], we chose a linear weight function, $w(a(\mathbf{x})) = c + b \cdot a(\mathbf{x})$, where $a(\mathbf{x})$ denotes the

acquisition function. The batching criterion expressed by Eq. (5.8) thus becomes[4]:

$$\arg\max_{\mathbf{x}\in\mathcal{X}} L_t^{\mathrm{AW}}(X) = \arg\max_{\mathbf{x}\in\mathcal{X}} K_{\mathrm{post}}(\mathbf{x}|\{\mathbf{x}_i\}_{i=1,\dots,k-1}) \cdot (c + ba(X))^2, \tag{5.19}$$

where the acquisition function $a(X)$ is the expected improvement, defined in Eq. (2.34) in Chapter 2. The function w has the role of balancing the variance controlled by the value c, and the acquisition function, controlled by b. To evaluate the effect of this trade off, different values of the parameter b were tested for each data set: specifically we set $b \in \{0, 1, 10, 100, 500\}$. The parameter c was kept fixed at $c = 1$ throughout all experiments, since it can be seen from Eq. (5.19) that the maximum point of $L^{\mathrm{AW}}(X)$ depends only on relative sizes of c and b.

Evaluation metrics. We measure the performance of the methods by evaluating the *simple regret*, R, defined as:

$$R = f_{\mathrm{best}} - f_{\mathrm{opt}}, \tag{5.20}$$

where f_{best} is the best observed value of the objective function, and f_{opt} is the absolute optimum. All experiments are run independently 30 times over different random initializations. All of these runs are then averaged to give the *average simple regret* $\hat{R}$.

Construction of SOAP descriptors. The SOAP descriptors for all the data sets were built before running the experiments and saved to disk. They were computed starting from the atomic coordinates, using the Python package DScribe [44]. As mentioned in Section 5.3.2, SOAP descriptors are computed as a function of the power spectrum of the atomic density $\mathbf{p}$, which in turn requires that the atomic density is expanded in a basis composed of spherical harmonics and a set of orthogonal radial basis functions, according to Eq. (5.11). The DScribe software allows different options for the computation of the radial basis functions, among which we chose the default option of *spherical Gaussian orbitals* [49], or GTOs. Thus each function $g_n(\mathbf{r})$ in Eq. (5.11) is calculated as the orthonormal composition of GTOs, as follows[5]:

$$g_{n\ell}(r) = \sum_{n'=1}^{n_{\max}} \beta_{nn'\ell} r^\ell e^{-\alpha_{n'\ell} r^2}, \tag{5.21}$$

[4]The logarithm is a monotonous function, so $\arg\max log(\mathbf{x}) = \arg\max x$.

[5]A single spherical Gaussian orbitals is defined as $g_\ell(\mathbf{r}) = A(\ell, \alpha) r^\ell e^{-\alpha r^2}$, where $\alpha \in \mathbb{R}$, and ℓ is a parameter of the function.

where n indicates a specific radial basis function, and ℓ is a parameter of the GTOs. The values of n and ℓ are limited up to $n_{\max}$ and $l_{\max}$ respectively, which are hyper-parameters of the SOAP descriptors, and which are set a priori. The factors $\beta_{nn'\ell}$ are multiplicative coefficients to make the composition orthogonal and normalised. For our experiments we set $n_{\max} = 5$, $\ell_{\max} = 8$. We set these values as a compromise between values that have been reported in literature [9, 19] for molecular databases similar to the ones we studied, and the necessity to contain the dimensionality of the descriptors, to keep the BBO process computable in a reasonable amount time.

Another important parameter for the SOAP descriptors is α, which is related to the width of the GTOs used to smooth the atomic coordinates, σ: $\alpha = \frac{1}{2}\sigma^2$. We set $\sigma = 0.2$. For the REMatch kernel, which we used in combination with SOAP descriptors, the parameter γ in Eq. (5.18a) is set to $\gamma = 0.1$.

Selection of molecules for QM9gaps, QM9polar. To reduce the QM9 data set for the **QM9gaps** and the **QM9polar** optimisation tasks, the following procedure was used:

1. The desired number of points in the reduced data was set to $n^{\text{red}} = 7,000$.
2. The histogram of the molecules' properties was computed.
3. From each bin b a number of points was selected, proportionally to the number n_b^{orig} of molecules contained in the b^{th} bin of the original data set:

$$n_b^{\text{red}} = \frac{n_b^{\text{orig}}}{n^{\text{red}}}. \tag{5.22}$$

The actual number of points in the reduced data sets is $n = \sum_{b=1}^{N} n_b^{\text{red}}$. If $n_b^{\text{red}} < 1$, still one molecule is selected from bin b, and this is the reason why the actual number of molecules in reduced data sets is slightly higher than $7,000$.

5.5 Results

Experimental results obtained for the two types of descriptors are shown in Figure 5.3, where simple regrets are plotted for Coulomb descriptors with an RBF kernel, and for SOAP descriptors with a REMatch kernel. All average regrets are averaged over 30 runs, for both Thompson and LAW2ORDER BBO. The best average regrets achieved by each method are also reported in Table 5.2 and Table 5.3. For Coulomb descriptors, the optimisation process has been performed over 500 steps and since our batch size $k = 3$, this means that 1500 queries were made to the objective function.

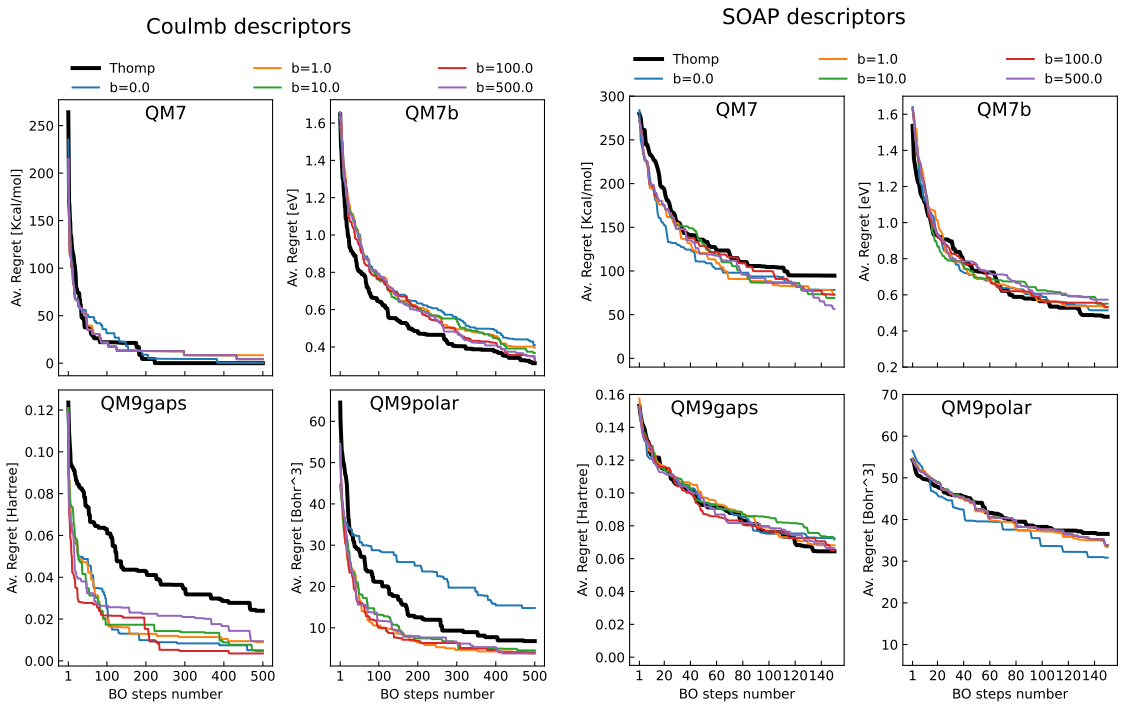

Figure 5.3: Comparison of the average regrets curves obtained with Coulomb descriptors and RFB kernel (left) and SOAP descriptors and REMatch kernels. Experimental results achieved with LAW2ORDER method at different value of the parameter **b** are compered with the results given by Thompson method (black line)

Experiments with SOAP descriptors have been restricted to 150 iterations, due to the much higher computational cost of Gaussian inference in this configuration. The reason behind this disparity is the different dimensionality of the descriptors: while Coulomb eigenvalues descriptors have maximum dimensionality $d = 29$, SOAP descriptors can easily exceed $d = 1000$. As a consequence, while 500 optimisation steps with LAW2ORDER method and Coulomb descriptors require on average 16 hours, around 104 hours are necessary to complete 150 steps with the same method but using SOAP descriptors and a REMatch kernel.

The regret curves in Figure 5.3 show that LAW2ORDER is always competitive with Thompson sampling, and it can be significantly better. In particular, LAW2ORDER clearly considerably outperforms Thompson sampling on QM9gaps and QM9polar with Coulomb descriptors. The only data set for which Thompson can be seen to outperform LAW2ORDER is QM7b, where it performs better for both kernel choices.

As mentioned earlier, the parameter b for LAW2ORDER controls how much

weight is allocated to the acquisition function, and how much is allocated to variance. It can be seen that in many of the experiments this parameter has little effect, with the exception of the results obtained for the QM9Gaps optimisation task, using Coulomb descriptors. However, the poor result for $b = 0$ (where no weight is put on the acquisition function) in QM9polar on Coulomb descriptors indicates that this setting should be avoided.

As a complement to Figure 5.3, Tables 5.2 and 5.3 report the best average regrets achieved by each method, using Coulomb and SOAP descriptors respectively, showing how at the late stages of the optimisation process the Thompson method catches up or even outperforms the LAW2ORDER method. Finally, it is interesting to compare results across the kernels: BBO with Coulomb descriptors appears to outperform BBO with SOAP descriptors. Although only a limited number of steps are executed for SOAP descriptors, due to the high computational cost, it can be seen that the results for Coulomb descriptors have lower average regret after 150 steps. Tables 5.2 and 5.3 show the best average regrets achieved by our methods.

Coulomb descriptors			
Data set	b_value	LAW Av. Regret	Thompson Av.Regret
qm7	0	0.241 ± 0.000	**0.005 ± 0.001**
	1	8.408 ± 0.468	
	10	4.273 ± 0.242	
	100	4.278 ± 0.340	
	500	4.281 ± 0.142	
qm7b	0	0.409 ± 0.103	**0.315 ± 0.088**
	1	0.396 ± 0.106	
	10	0.368 ± 0.083	
	100	0.330 ± 0.115	
	500	0.332 ± 0.161	
qm9gaps7K	0	0.005 ± 0.009	0.026 ± 0.019
	1	0.009 ± 0.014	
	10	0.005 ± 0.003	
	100	**0.004 ± 0.003**	
	500	0.009 ± 0.011	
qm9polar7K	0	14.784 ± 1.342	6.939 ± 1.290
	1	3.803 ± 0.438	
	10	4.548 ± 0.725	
	100	4.066 ± 0.742	
	500	**3.793 ± 0.587**	

Table 5.2: Comparison final simple regrets obtained with the LAW2ORDER and Thompson BBO methods, using Coulomb descriptors and RBF kernel. The best result for each data set is reported in bold

SOAP descriptors			
Data set	b_value	LAW Av. Regret	Thompson Av.Regret
qm7	0	**52.068 ± 1.858**	77.404 ± 1.884
	1	64.789 ± 1.960	
	10	69.010 ± 1.956	
	100	60.223 ± 1.914	
	500	52.299 ± 1.868	
qm7b	0	0.464 ± 0.109	**0.457 ± 0.119**
	1	0.511 ± 0.088	
	10	0.547 ± 0.082	
	100	0.528 ± 0.091	
	500	0.561 ± 0.086	
qm9gaps7K	0	0.065 ± 0.011	**0.058 ± 0.016**
	1	0.062 ± 0.017	
	10	0.071 ± 0.011	
	100	0.062 ± 0.017	
	500	0.062 ± 0.019	
qm9polar7K	0	**27.977 ± 2.776**	36.753 ± 1.512
	1	33.826 ± 2.341	
	10	31.399 ± 2.328	
	100	31.399 ± 2.328	
	500	31.399 ± 2.328	

Table 5.3: Comparison final simple regrets obtained with the LAW2ORDER and Thompson BBO methods, using SOAP descriptors and REMatch kernel. The best result for each data set is reported in bold

LAW2ORDER combined with Thompson sampling as acquisition function

In this section we report results of preliminary experiments for future work, where we applied the LAW2ORDER BBO method using Thompson sampling as a sequential acquisition function. We call this approach *Thomp-LAW2ORDER*, while here we will refer to the LAW2ORDER BO method used so far *EI-LAW2ORDER*.

The purpose of this experiment is to explore possible options for the sequential acquisition function $a(\mathbf{x})$, which, as can be observed from Eq. (5.6) can strongly influence the L-ensemble, and thus the performance of the LAW2ORDER BBO method. Already in the original work, Oh and coauthors [79] tested multiple type of acquisition functions, like Multi-objective acquisition ensemble (MACE) [67] and EST [111],

as well as more common ones like upper confidence bound. However some of them can be remarkably more expensive than expected improvement. As the LAW2ORDER BBO method has proven to be quite computationally intensive in the space of molecular descriptors, their application might be challenging.

Alternatively, we explored the suitability of replacing EI with Thompson sampling: in this case the acquisition function $a(\mathbf{x})$ is obtained by sampling a function from the surrogate model, analogously to what is shown in Algorithm 7. Such a function is evaluated over the search space. The rest of the LAW2ORDER batching policy the proceeds as described in Algorithm 8. This choice is motivated by the fact that Thompson is a widely used sampling method, it is computationally inexpensive and conceptually simple.

The results obtained for Coulomb descriptors with the Thomp-LAW2ORDER method are compared with those achieved with EI-LAW2ORDER, and the respective average simple regrets are plotted in linear and logarithmic scale in Figures 5.4 and 5.5 respectively.

The first approach can be seen to be advantageous for QM7 and, to a lower extent, for QM7b optimisation problems, while using EI is preferable for QM9gaps and QM9polar problems. In general, the results reveal that the acquisition function has lower impact when Thomp-LAW2ORDER is applied. In particular no discernible difference can be observed between different values of the parameter b, for the QM7 data set.

However an important advantage of the Thomp-LAW2ORDER is the lower computational time compared to EI-LAW2ORDER, as shown in Figure 5.6, where the time necessary to complete 500 iterations is compared for both these two methods and for the Thompson BO method. We found that time required by the Thomp-LAW2ORDER BBO method is intermediate between the other two methods. Overall the improvements observed with Thomp-LAW2ORDER are not yet sufficient to comfortably apply the LAW2ORDER BBO method to materials optimisation tasks, particularly using SOAP descriptors. It can be still worthwhile to employ Thomp-LAW2ORDER for Coulomb descriptors, as it gives comparable or better performance than EI-LAW2ORDER in significantly shorter time, but for the SOAP descriptors other strategies need to be tested.

We suggest that another possible direction for future research is to combine LAW2ORDER batching policy with BBO methods which have been proven to be more performant in high dimensional space. As BO is known to have poor scalability with the dimensionality of the search space [32] a few methods have been proposed in literature to address this problem [21, 25, 26, 68]. Among these, the TURBO method [26] is particularly promising, as it is applicable in the discrete space.

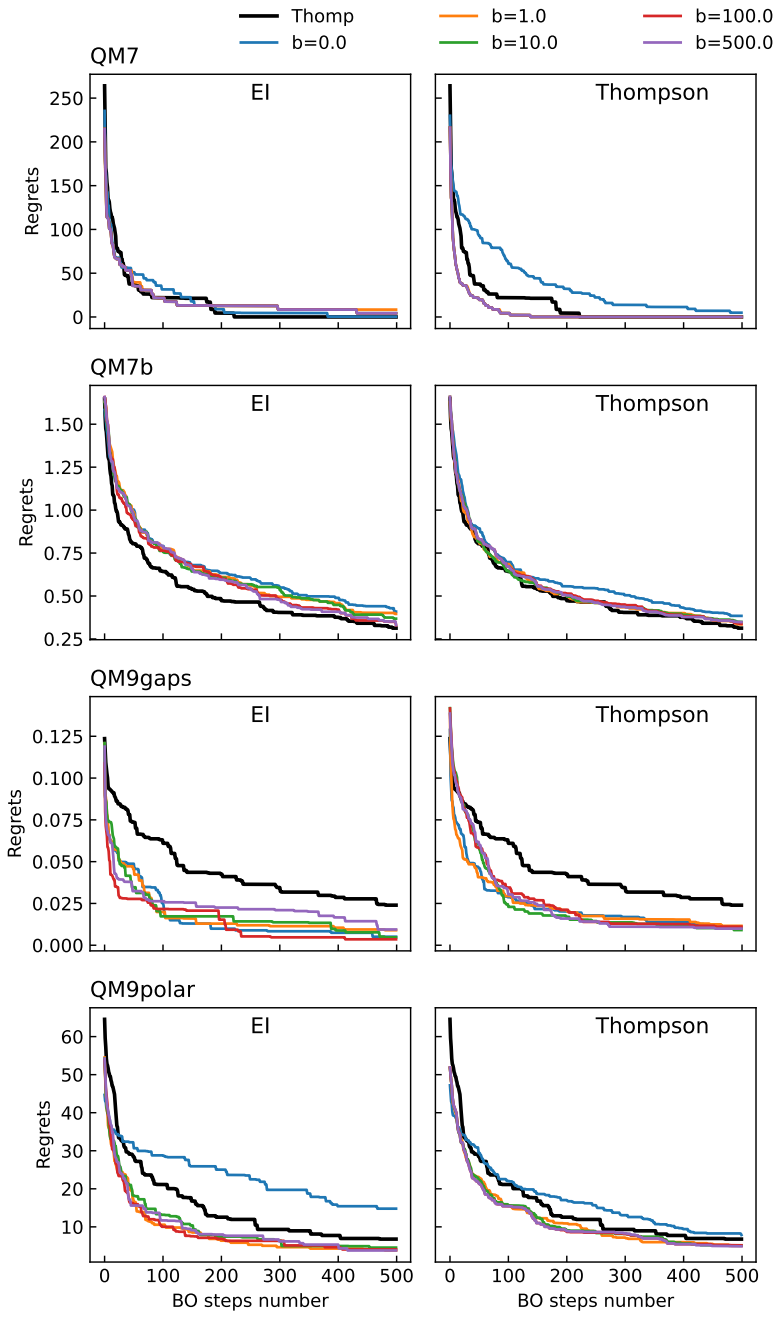

Figure 5.4: Average final regrets for all the optimisation tasks, for Coulomb descriptors, in linear scale. The curves plotted in the left column and in the right refer to results obtained with EI-LAW2ORDER and Thomp-LAW2ORDER BBO methods respectively.

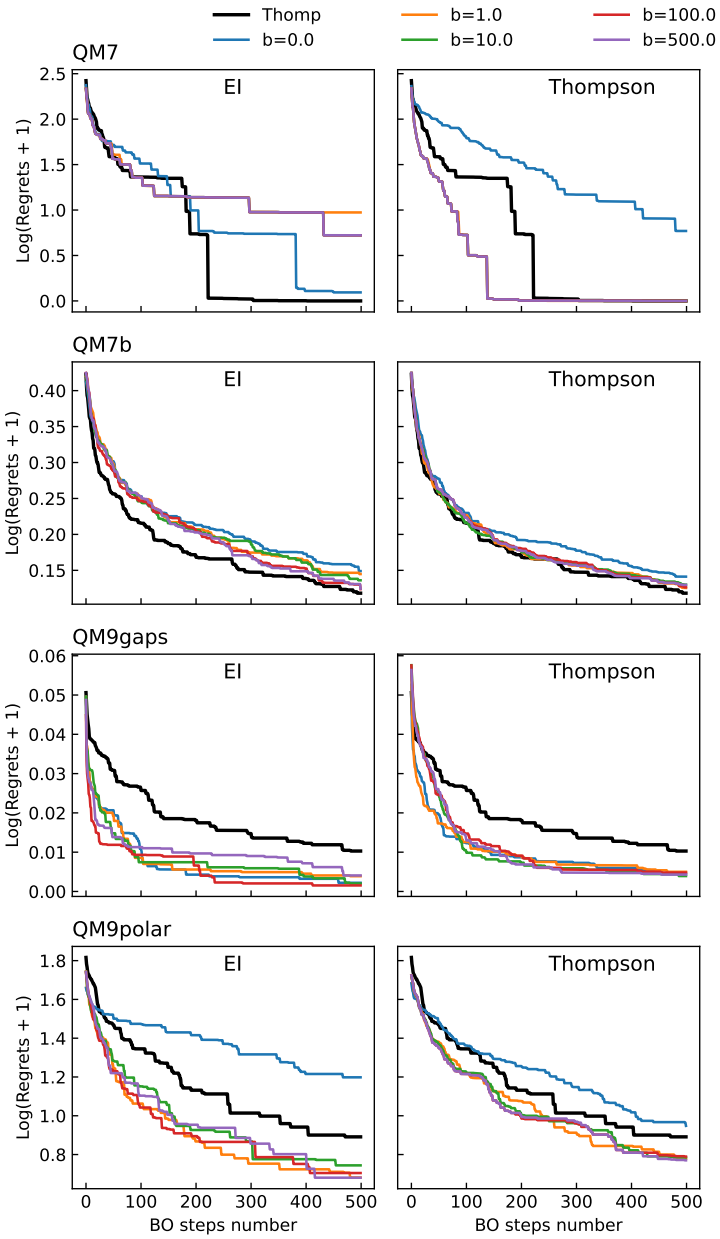

Figure 5.5: Logarithm of average final regrets for all the optimisation tasks, for Coulomb descriptors. The curves plotted in the left column and in the right refer to results obtained with EI-LAW2ORDER and Thomp-LAW2ORDER BBO methods respectively. All regrets have been shifted by +1, so the values 0 on the y axis correspond to a null regret.

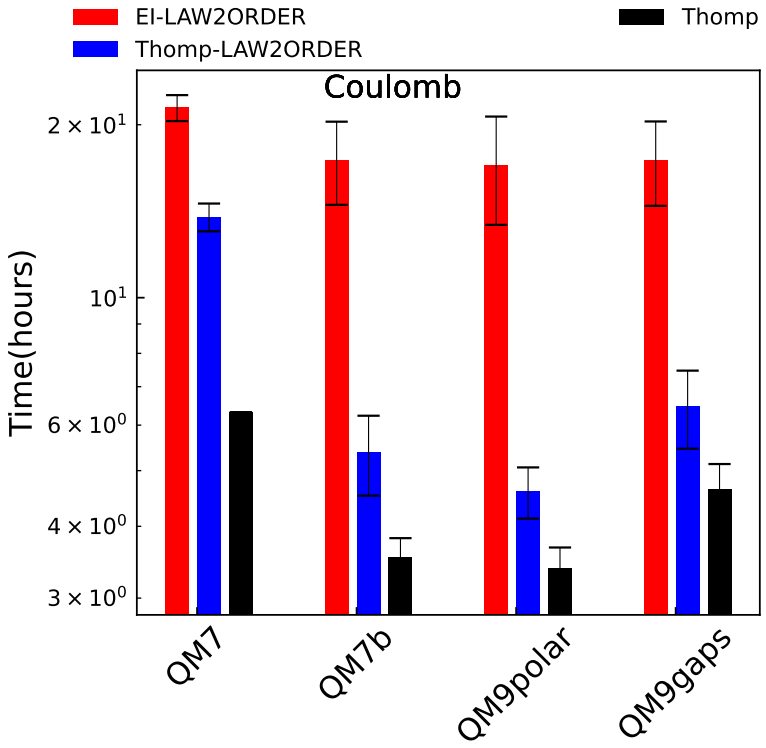

Figure 5.6: Computational time for the EI-LAW2ORDER, the Thomp-LAW2ORDER and the Thompson methods, using Coulomb descriptors. Time is expressed in hours.

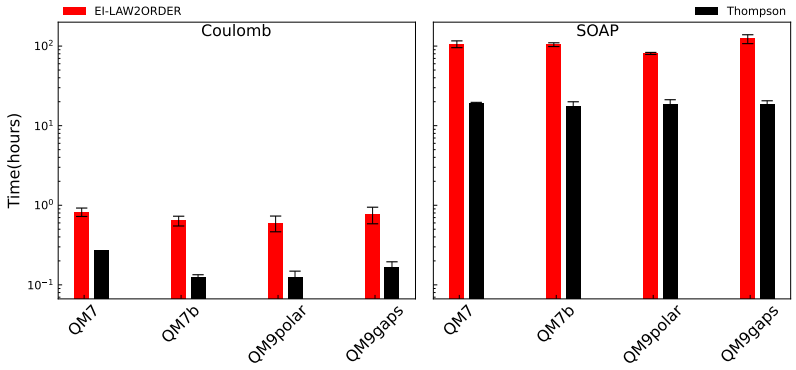

Figure 5.7: Computational time for the LAW2ORDER and the Thompson methods, using Coulomb descriptors (left graph) and SOAP descriptors (right graph). Time is expressed in hours.

5.6 Conclusions

The results presented in this chapter show that the LAW2ORDER BBO method outperforms the Thompson based BBO method, for any value of the parameter b, except for $b = 0$, if Coulomb descriptors are used. Also, the poor results obtained for some of the optimisation tasks when $b = 0$, confirms the hypothesis that taking into account the acquisition function in the batching policy improves the performance of BBO.

However, the impact of the acquisition function is not clear, as the relationship between the parameter b and the average regrets is quite variable among the different optimisation tasks, and no discernible pattern can be observed. A possible reason for this is that the parameter b needs to be tuned in a range of values different from the one we chose.

The results obtained using the LAW2ORDER method with SOAP descriptors and REMatch kernel are comparable or worse than those achieved with the Thompson BBO method in the same settings. Also, this choice of descriptors and kernel makes the LAW2ORDER method computationally expensive, as 150 iterations required on average 103 hours compared to the 17 hours required to perform 500 iterations with Coulomb descriptors.

Unfortunately, as described in [23], alternative kernels suitable to use with SOAP descriptors have lower capacity to measure similarity between molecules, and are thus unlikely to result in improved convergence of LAW2ORDER BBO. In view of all these considerations, we conclude that, while the LAW2ORDER method has shown encouraging results with Coulomb descriptors, and it is worthwhile to be explored in future work in this configuration, its application to SOAP descriptors presents no advantages.

Chapter 6

Conclusions

6.1 Overview

We have presented research focused on the development of Bayesian optimisation methods, intended to accelerate the design of experiments, particularly in the field of Materials Discovery. We investigated both sequential and batched BO approaches, which were designed to deal with different scenarios.

In Chapters 3 and 4 we introduced two main sequential BO methods, i.e. the exclusion radius and the discrepancy prediction methods and their variants. These methods have been developed to deal with the scenario where prior knowledge about the current optimisation task is available, in the form of an approximate model of the objective function to optimise. Such a model, which we called the predictor, is exploited to generate a number of starting points to warm start the BO process, so as to accelerate convergence. All our sequential methods with initial predicted data have been formulated under the assumption that the predictor has close to no evaluation cost, but it might present non negligible disparity with respect to the actual objective, at least locally. The main contribution of this part of our work is the development of strategies to deal with the inaccuracy of the predictor.

A second approach we explored is the implementation of a batched Bayesian optimisation method (BBO) applicable to optimisation tasks defined over molecular spaces, and suitable for discrete input variables. The key advantage of BBO compared to sequential BO is the acquisition of multiple points at each step of the optimisation process. This can considerably speed up the convergence, provided that the objective function can be evaluated at all the batch points in parallel. In the case where this evaluation is given by the results of physical experiments, the experiments need to be executed in parallel batches in order to benefit from BBO. Optimisation in discrete spaces is challenging specifically for BBO, due to the fact that it cannot rely

on the acquisition functions designed for sequential BO, which are perfectly suitable to handle discrete as well as categorical variables. Acquisition functions designed to suggest multiple points at once need to be used instead, and the definition of such acquisition functions in discrete space is presently still not well established.

To address this issue, we studied the transferability of LAW2ORDER, a state-of-the-art BBO method originally proposed to solve optimisation tasks over permutation problems, to the optimisation of chemical and physical properties of small organic molecules. The goal of this investigation is to find a BBO method for discrete spaces that outperforms the Thompson-BBO and the DPPs-BBO methods, described in Chapter 5. These methods have already been successfully applied to accomplish BBO in discrete space, but they lack an appropriate balance between variety of the batch points and their informativeness. As mentioned in Chapter 5 the Thompson-BBO method is mainly exploitative, while the DPPs-BO method enforces only diversity. The LAW2ORDER method balances diversity and informativeness by combining determinantal point processes with a sequential acquisition function.

However its transferability to the materials design optimisation tasks poses at least two challenges:

- The appropriate choice of molecular descriptors.
- The choice of a similarity kernel between molecules, compatible with the selected type of descriptor.

The correct choice of molecular descriptors is essential to ensure that we capture the features of the molecule that are most relevant to the optimisation task at hand. The choice of similarity kernel is also crucial as it determines the posterior variance of the GP surrogate model, K_{post}, and thus the batching policy, as shown in Eq. (5.6).

6.2 Summary of experimental results

The outcome of the of all the BO approaches we implemented, both sequential and batches are summarised below.

6.2.1 Methods for the exploitation of predicted data

Among all the sequential methods we implemented, the most performant is the exclusion radius method, which outperforms the multi-fidelity method in majority of cases, especially in terms of early convergence: this means that if we decide to end the optimisation when we reach a satisfactory, but not-optimal, value for our objective, the exclusion radius method could reach this target earlier than the multi-fidelity method in all cases. Experints on benchmark functions, where the value of

the absolute optimum was known, show that this sub-optimal target can be as close to the actual optimum as by 5%, as discussed in Section 3.6. Generally, also the discrepancy prediction method reaches early convergence before multi-fidelity BO, with few exceptions.

To assess that the advantage of our methods was not due merely to the warm starting with predicted points, we also performed experiments with the exclusion radius method, setting the radius $r = 0$, which implies that no predicted points were deleted. We observed that, despite the results obtained with this setting are competitive, the results observed either with the exclusion radius method for higher values of r or with the discrepancy prediction method were generally better, particularly in high error regime. This proves that applying a strategy to deal with the predictor's inaccuracy contributes to accelerate the convergence of the optimisation process. From the analysis of the regrets however, we could conclude that the advantage of both the exclusion radius and of the discrepancy prediction methods over the multi-fidelity method was maximum at the early optimisation stages, and it extinguishes later on.

For this reason we developed several variants of both methods, with the intent to find more efficient criteria to eliminate or to correct the predicted points.

Despite our efforts, the original methods described in Chapter 3 are better or comparable to the any of the variants illustrated in Chapter 4.

Below we provide a summary of these attempts, and the interpretation of the results they gave.

Exclusion radius methods

To improve the exclusion radius method we developed two strategies to remove all predicted points before the end of BO. The first one, called the early switch method, is a rather naive one, which at a given iteration eliminates all the remaining predicted points at once. This method was intended as a preliminary study to explore if there was any benefit in continuing with only real points in the later stages of BO. Unsurprisingly the early switch method did not outperform the original exclusion radius and it will not be further discussed here.

The second strategy, i.e. the adaptive radius exclusion method, consists in progressively increasing the radius r at each optimisation step, so to accelerate the removal of the predicted points as the BO proceeds and thus their presence is potentially more and more detrimental. Also this method is designed to delete all predicted points by the end of the optimisation process, but in a gradual way. We explored a number of different rates for the parameter r, but none of these values achieved a consistent improvement over the original exclusion radius method. Retrospectively, we discovered that the predictive accuracy of the predictive accuracy of

the GP surrogate model dropped just after the complete elimination of the predicted points, as shown in Figures 4.10.

A possible explanation is that after the deletion of all predicted data, some portions of the search space remain with very little data or with no data at all, so that the GP surrogate model has no longer good information in those regions. This could happen because while the predicted points are uniformly sampled across all the space, the real points are selected according to the acquisition function, which mainly focuses the search in regions that have the highest probability of containing the optimum.

Discrepancy prediction methods

In order to improve the discrepancy prediction method we focused on improving the predictive accuracy of the surrogate model for the discrepancy. For this purpose we developed two strategies: one that relies on Gradient Boosted Regressor (GBR) to model the discrepancy, while the other one uses a stochastic sampling policy, called the FarNeighbour, to augment the data set of the surrogate model for the discrepancy.

The motivation behind the first strategy is the large flexibility GBR models are known for, which results from the fact that GBR, being an ensemble learning model, combines multiple learners. Such a flexible model could potentially be very good in capturing the real structure of the discrepancy, which, in principle, might display large variability across the search space.

The discrepancy prediction method combined with the FarNeighbour policy instead is designed to selectively increase the accuracy of the surrogate model for the discrepancy in regions of the space which are useful to solve the current optimisation task, by randomly sampling extra real points in proximity of the last point suggested by the acquisition function. FarNeighbour policy has been used in combination with both GP or BGR models as a surrogate for the discrepancy.

Unfortunately neither of these two approaches has proved to be better than the original discrepancy prediction method consistently across all the experimental configurations.

A possible pitfall of the FarNeighbour policy is the overall reduction in the exploration of the space, as discussed in Section 4.5.2.

We point out here that even if the discrepancy prediction method with GBR shows lower predicted regrets when it is applied in combination with the FarNeighbour policy, we do not consider this a relevant success, as predicted regrets as a measure of convergence is typically used only for multi-fidelity BO approaches.

In the case of the discrepancy prediction method with GBR, we ascribed the lack of improvement with respect to the original method to possible overfitting of

the model due to the low dimensionality of the benchmark functions to which we applied it. However we also suggest that this approach could still be successfully applied to materials design optimisation tasks, which are defined over significantly higher dimensions: for instance the tasks reported in Chapter 5 had dimensionality 29 when we used Coulomb descriptors.

Final Conclusions

The verdict on all the methods with predicted data we implemented can be summarised as follows:

1. The best of all methods with predicted data is the original version of the exclusion radius method.

2. The experimental setting we suggest to use for the enumerate is $r/l = 0.1$, where r is the radius and l is the length of the dimension of the search space (assuming that the value of l is the same for all the dimensions): this has proven to be work for all the benchmark functions we studied, in all the error regimes.

3. Both the exclusion radius and the discrepancy prediction methods generally outperform the multi-fidelity BO method in terms of early convergence, but the first one is more performant, especially if the final regrets after all the 80 real observations are considered.

4. The best discrepancy prediction method is the original version, presented in Chapter 3, at least for the optimisation problems we investigated. However we are still considering the application of this method with GBR a possible approach to explore for problems in the field of materials discovery.

5. An important advantage of our methods with predicted data over the multi-fidelity approach is the much shorter computational time, which is in the order of minutes for both exclusion radius and discrepancy prediction methods, while it varies from about 1 hour to 5 hours for the multi-fidelity method, depending on the cost assigned to the actual objective function[1].

[1]This comparison holds if all the methods are stopped after the same number of real observations, which typically implies a significantly higher number of BO iteration for the multi-fidelity method.

6.2.2 LAW2ORDER BBO method for molecular properties optimisation

The experimental results we observed for the LAW2ORDER BBO method applied to molecular properties optimisation led to the following Conclusions:

1. The LAW2ORDER method generally outperforms the Thompson-BBO method when Coulomb descriptors are used to represent molecules.

2. This same method does not present any advantage over the Thompson-BBO approach if SOAP descriptors are used. On the contrary, the LAW2ORDER method would result in much higher computational cost, without reaching any faster convergence. The LAW2ORDER method with SOAP descriptor is also much slower than the same method with Coulomb descriptors.

3. Taking into account the acquisition function for the batching policy generally leads to better performance for the LAW2ORDER method, with respect to considering the posterior variance only.

4. However we could not establish a straightforward relationship between the weight given to the acquisition function and the convergence rate of the optimisation process, and further investigation is needed in this respect.

The fact that we managed to outperform the Thompson-BBO method at least with Coulomb descriptors is encouraging, although we need also to notice that the Thompson based method requires shorter computational time: 500 iterations with Coulomb descriptors took on average 17 hours with the LAW2ORDER method and around 4.5 hours with the Thompson-BBO method.

Point 2 proves that, as expected, molecular descriptors and similarity kernels have a large influence on the performance of the LAW2ORDER method: Coulomb descriptors could be a better representation for the molecules that we studied. However the high dimensionality of the SOAP descriptors is also an obstacle, regardless of which molecules are considered. This issue can be addressed either choosing molecular descriptors with lower dimensionality or combining the LAW2ORDER method with BBO approaches suitable for high dimension spaces, as shown in Section 6.3.

Present perspective of LAW2ORDER real world applications

To prove that the LAW2ORDER method can be applied to real world problems in the field materials design, we tested its application to a benchmark chemical reaction, with potential environemental application.

This part of the project, which currently is still ongoing, is in collaboration with the research group directed by Prof A. Cooper at the Department of Chemistry in the University of Liverpool. The chemical process under study is an aqueous photolysis reaction between of a pharmaceutical compound dissolved in water and a second organic molecule, in presence of light. As a result of this reaction the pharmaceutical compound, in our case cimetidine, is decomposed in smaller molecules, or degraded, as described in [17].

As the degradation process is typically not complete, and a certain amount of cimetidine is left after the end of the reaction, we aim to find the molecule which reacts with the highest amount of cimetidine, leaving us with the smallest amount of residual cimetidine. Specifically, our task is the selection among a given data set of candidates, all of which are organic dyes.

To solve this optimisation problem we used the LAW2ORDER method with Coulomb descriptors and RBF kernel, since this configuration gave the best results on the tasks studied in Chapter 5. Figure 6.1 represents a scheme of the experiment: the top of the Figure illustrates the basic steps of the experiment. In particular the last steps shows how the remaining concentration of cimetidine is quantified, using an experimental technique called high-pressure liquid chromatography (HPLC). The bottom of Figure 6.1 represents instead the Bayesian optimisation loop we used in this context: the input variables are the Coulomb descriptors of the batch of dye molecules that, at each experiment, have been used to react with cimetidine. The results of the HPLC measurements are the observations of our objective function, which is the final concentration of cimetidine. At each BBO step the LAW2ORDER batching policy generates a batch of 16 new combinations of a particular dye and its concentration in water, for the next set of experiments. The batch size we choose corresponds to the maximum number of experiments that can currently be performed in parallel.

While the software to run LAW2ORDER is currently integrated in the whole loop for the experimental set up, the experimental results are not yet available at the time of writing.

However we have achieved a proof of concept application of this BBO method for discrete molecular space.

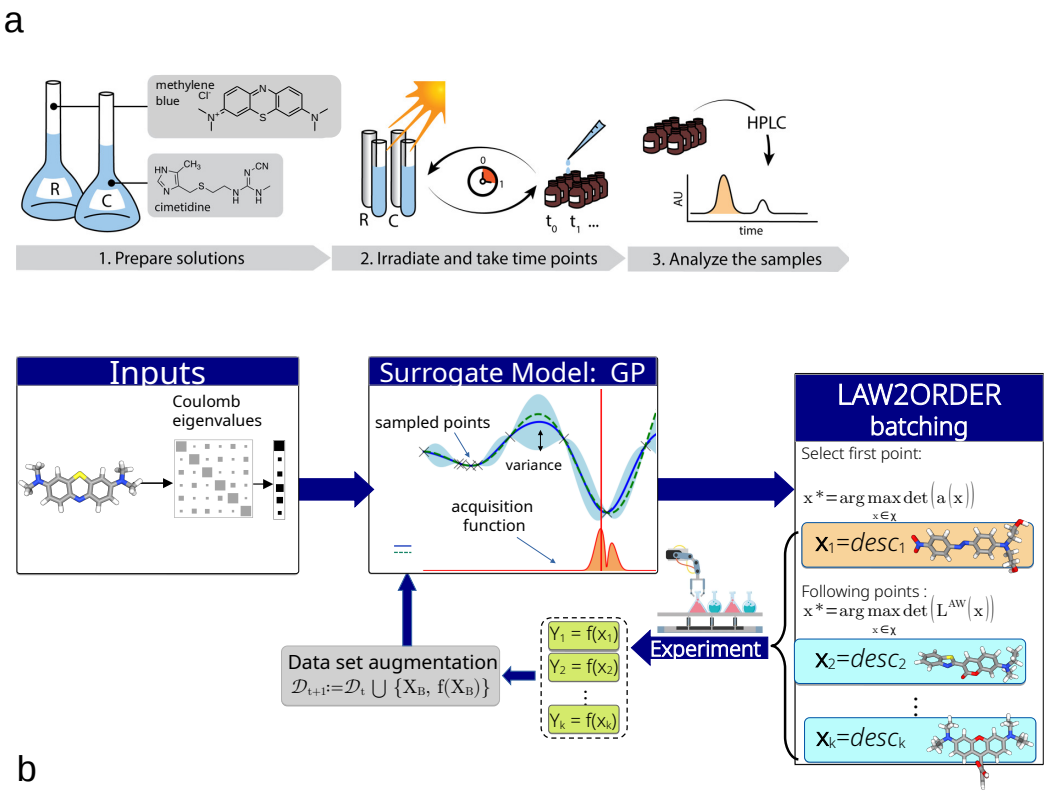
a
methylene blue
cimetidine
R
C
1. Prepare solutions
2. Irradiate and take time points
HPLC
AU
time
3. Analyze the samples
Inputs
Coulomb eigenvalues
Surrogate Model: GP
sampled points
variance
acquisition function
LAW2ORDER batching
Select first point:
Following points :
Experiment
Data set augmentation
b

Figure 6.1

6.3 Future work

At the end of this thesis we propose possible extensions of the research. We identified possible ways to further develop the work we presented here.

6.3.1 Exclusion radius and discrepancy prediction method for materials design

First of all we propose to apply the exclusion radius and the discrepancy prediction methods introduced in Chapter 3 to real world optimisation problems related to materials design. These methods are developed for sequential Bayesian optimisation, and thus are applicable to both continuous and discrete search spaces. As previously mentioned in this chapter, we also consider the discrepancy prediction method with GBR a potentially interesting approach to test in the molecular space.

Here we suggest to use machine learning models as predictors to approximate the actual objective function and generate predicted data, in the hypotheses that the evaluation of such models is cheaper than performing real experiments.

Traditional machine learning methods like kernel ridge regression [24], extreme boost gradient regression [28, 36, 110], k-means [38] and random forests [65, 106] have already been employed to predict materials properties, like shear stress, thermal conductivity and dielectric properties. More recently used models however include graph neural networks [83, 90], recurrent neural networks [93], variational autoencoders [63, 81]. These last approaches are more suitable for large data sets compared with the ones mentioned before and, generally, are better at capturing complex patterns and relationships. On the other hand, usually they are also computationally more expensive and tend to require a larger number of training points. The advantage in using rather than acquiring real observations depends on the computational resources available, on the duration and material cost of running a real experiment as well as the availability of pre-trained models.

6.3.2 Improving the LAW2ORDER method

Despite giving encouraging results when used with the Coulomb descriptors and an RBF kernel, the LAW2ORDER method still presents slow convergence, and the impact of the sequential acquisition function on the performance of this method needs further investigation. In Chapter 5 we ascribed slow convergence to the high dimensionality of the molecular descriptors we used, especially for SOAP descriptors, and for this reason we suggested the combination of the LAW2ORDER batching policy with the TURBO BBO method [26] which is designed to handle high dimensional inputs and it is also applicable to discrete spaces.

The TURBO method deals with high dimensionality by partitioning the search space in smaller regions, called the *trust regions*, and a local optimisation in each trust region is performed using an independent GP. At each BBO step, an acquisition function selects a batch of candidates. The best point at this iteration is found among the union of all the batch points across all the thrust regions. We could apply the TURBO method using a LAW2ORDER to select batches locally in each trust region.

Once we manage to mitigate the problem of high dimensionality, we could start to explore the use of different sequential acquisition function for the LAW2ORDER batching policy, according to Eq. (5.6). For example we could test how this method performs the EST acquisition function [111], which has been used in the original paper by Oh et al. [79], in lower dimensional space compared to the one we explored in Chapter 5.

6.3.3 Towards application to biomolecules

The real world applications of our methods so far were restricted to small organic molecules, but in the last few years the interest in the field of Materials Science and Drug Discovery has focused on biomolecules, largely on aminoacids and proteins. Consequently, a number of BO approaches have been developed for optimisation problems related to this type of molecules [15, 46, 113], and we can also think of extending the methods described in this thesis to this topic.

However we can already foresee a challenge: indeed the size of biomolecules is such to make their numerical encoding extremely high dimensional, if we use molecular descriptors that encode their atomic composition, like the Coulomb or the SOAP descriptors we used for small molecules. For instance descriptors often used in literature to represent proteins encode their geometric properties, like tertiary and secondary structures[2], rather then atomic composition. Alternatively, the numeric value of chemical and physiochemical properties, such as hydrophobicity and charge of a protein, can be used as input variables for Bayesian optimisation.

Thus, if for example we want to apply the LAW2ORDER method to proteins we suggest to explore the use of the aforementioned descriptors, which in many cases can be combined with well known kernels like RBF.

6.3.4 Software considerations

All the experiments reported in this thesis have been implemented using GPyOpt package, with the exception of the multi-fidelity experiments in Chapter 3, for which

[2]Proteins have usually a quite complex 3D structure, hierarchically organized at multiple levels, in primary, secondary and tertiary structures. When multiple proteins aggregate with each other the form what is called a quaternary structure. More details can be found in [15]

we used Emukit. Currently GPyOpt is no longer maintained, and can thus be considered obsolete. For the future we other libraries might be considered, taking into account their computational speed, their versatility, the support for parallelization and the potential use of GPUs, and the amount of state of the art methods implemented.

Let us consider here the following libraries: scikit-optimize[3], Emukit [84], GPFlowOpt [71], BOTorch [8]. The scikit-optimize library is a popular library, which is now comparable to GPyOpt in terms of ease of use and amount of methods implemented. However it does not have built-in support for parallel optimization.

Emukit is more flexible and modular, requiring less effort to include extra computational steps in the BO loop. Furthermore built-in support for parallel optimization.

GPFlowOpt is built on top of TensorFlow and thus it can exploit parallel computation resources and it can use GPUs when available. For these reasons it can achieve higher computational efficiency compared to the Emukit and scikit-optimize. This library is conceptually structured as GPyOPt, and it has almost all the same methods implemented, but with the benefit of computational acceleration. Thus it could be a good choice for the future, but an even more convenient choice could be BOTorch.

BOTorch is built on top of PyTorch and, like GPFlowOpt it can efficiently perform parallel computation on GPUs, making it suitable for parallel evaluations. Compared to GPFlowOpt however, it provides the implementation of several state of the art Bayesian optimisation approaches, including the TURBO BBO method aforementioned.

At the end of these considerations, for the purpose of the future work we outlined in this Chapter, we would suggest to opt for BOTorch.

[3]https://github.com/scikit-optimize/scikit-optimize

Appendix A

Results Tables for Methods to integrate Predicted Data into Bayesian optimisation

A.1 The Early Switch Method

A.1.1 Simple regrets at low error regime, for the early switch method

LOW ERROR				
Function	**r/l**	$\mathbf{R_{exclusion}}$	$\mathbf{R_{Switch,best}}$	$\mathbf{t_{sw}^{best}}$
Ackley	0.05	**3.282E-02** ± **4.084E-03**	3.439E-02 ± 2.556E-02	70
	0.1	**3.352E-02** ± **3.663E-03**	3.363E-02 ± 2.246E-02	70
	0.15	**4.305E-02** ± **4.743E-03**	4.353E-02 ± 2.965E-02	70
	0.2	**4.606E-02** ± **5.070E-03**	4.948E-02 ± 3.411E-02	30
	0.3	5.539E-02 ± 6.053E-03	**5.274E-02** ± **3.742E-02**	30
Griewank	0.05	**3.986E-03** ± **5.059E-04**	4.629E-03 ± 3.603E-03	70
	0.1	**5.045E-03** ± **5.035E-04**	5.685E-03 ± 3.615E-03	70
	0.15	6.127E-03 ± 6.181E-04	**5.716E-03** ± **4.313E-03**	20
	0.2	6.271E-03 ± 5.354E-04	**6.258E-03** ± **3.463E-03**	60
	0.3	6.757E-03 ± 5.284E-04	**6.542E-03** ± **3.626E-03**	30
michalewicz	0.05	**8.460E-04** ± **1.006E-04**	8.504E-04 ± 6.941E-04	70
	0.1	7.162E-04 ± 1.023E-04	**7.127E-04** ± **7.101E-04**	60
	0.15	9.485E-04 ± 1.244E-04	**9.193E-04** ± **7.852E-04**	60
	0.2	1.190E-03 ± 1.427E-04	**1.156E-03** ± **9.941E-04**	50
	0.3	1.420E-03 ± 1.542E-04	**1.332E-03** ± **1.097E-03**	40
Rastrigin	0.05	**5.209E-01** ± **1.157E-01**	6.528E-01 ± 7.985E-01	70
	0.1	**5.491E-01** ± **1.434E-01**	6.916E-01 ± 1.050E+00	70
	0.15	**6.479E-01** ± **9.089E-02**	8.573E-01 ± 8.986E-01	60
	0.2	**6.770E-01** ± **1.031E-01**	7.293E-01 ± 7.159E-01	70
	0.3	1.004E+00 ± 1.304E-01	**9.386E-01** ± **9.279E-01**	40
Styblinsky	0.05	1.555E-01 ± 5.335E-02	**1.019E-01** ± **9.541E-02**	70
	0.1	**7.768E-02** ± **1.540E-02**	1.034E-01 ± 5.522E-02	50
	0.15	**6.892E-02** ± **1.523E-02**	7.219E-02 ± 7.890E-02	40
	0.2	8.250E-02 ± 1.866E-02	**7.382E-02** ± **6.636E-02**	40
	0.3	1.050E-01 ± 1.724E-02	**9.681E-02** ± **6.721E-02**	10

Table A.1: Final **simple** average regrets in the **low** error regime. Results for the exclusion radius method radius are compared to the best results for the early switch method, obtained for tbest t_{best}. Values in bold indicate the lowest regret at each value of r/l. The absolute best result for each benchmark is underlined.

A.1.2 Simple regrets at medium error regime, for the early switch method

MEDIUM ERROR				
Function	**r/l**	$\mathbf{R_{exclusion}}$	$\mathbf{R_{Switch,best}}$	$\mathbf{t_{sw}^{best}}$
Ackley	0.05	**3.697E-02 ± 3.155E-03**	4.084E-02 ± 3.078E-02	70
	0.1	**3.757E-02 ± 3.145E-03**	3.936E-02 ± 2.782E-02	70
	0.15	**4.189E-02 ± 3.100E-03**	4.290E-02 ± 2.882E-02	60
	0.2	**4.534E-02 ± 3.477E-03**	4.590E-02 ± 3.050E-02	60
	0.3	**4.988E-02 ± 4.095E-03**	5.176E-02 ± 3.656E-02	60
Griewank	0.05	**3.892E-03 ± 4.972E-04**	5.314E-03 ± 3.900E-03	70
	0.1	**5.357E-03 ± 5.126E-04**	6.049E-03 ± 3.728E-03	70
	0.15	**5.823E-03 ± 4.911E-04**	5.883E-03 ± 3.827E-03	60
	0.2	7.356E-03 ± 7.075E-04	**6.397E-03 ± 4.019E-03**	20
	0.3	**6.075E-03 ± 5.309E-04**	6.222E-03 ± 3.812E-03	60
michalewicz	0.05	**6.505E-04 ± 7.988E-05**	7.198E-04 ± 6.687E-04	60
	0.1	**7.247E-04 ± 8.407E-05**	7.708E-04 ± 7.061E-04	30
	0.15	9.158E-04 ± 1.178E-04	**8.968E-04 ± 9.685E-04**	30
	0.2	**7.819E-04 ± 8.558E-05**	7.949E-04 ± 7.138E-04	50
	0.3	1.097E-03 ± 1.109E-04	**1.042E-03 ± 9.294E-04**	20
Rastrigin	0.05	**6.299E-01 ± 1.133E-01**	6.667E-01 ± 7.366E-01	60
	0.1	**6.929E-01 ± 1.093E-01**	9.196E-01 ± 1.105E+00	70
	0.15	**5.244E-01 ± 9.302E-02**	6.539E-01 ± 8.485E-01	50
	0.2	**6.474E-01 ± 7.790E-02**	6.841E-01 ± 6.546E-01	60
	0.3	1.062E+00 ± 1.670E-01	**1.054E+00 ± 1.084E+00**	10
Styblinsky	0.05	2.042E-01 ± 9.778E-02	**6.806E-02 ± 5.470E-02**	70
	0.1	**6.940E-02 ± 1.349E-02**	9.517E-02 ± 8.542E-02	60
	0.15	1.139E-01 ± 1.661E-02	**1.006E-01 ± 6.881E-02**	50
	0.2	**1.255E-01 ± 1.849E-02**	1.288E-01 ± 1.093E-01	10
	0.3	1.057E-01 ± 1.561E-02	**8.324E-02 ± 6.845E-02**	20

Table A.2: Final **simple** average regrets in the **medium** error regime. Results for the exclusion radius method radius are compared to the best results for the early switch method, obtained for tbest t_{best}. Values in bold indicate the lowest regret at each value of r/l. The absolute best result for each benchmark is underlined.

A.1.3 Simple regrets at high error regime, for the early switch method

HIGH ERROR				
Function	**r/l**	$\mathbf{R_{exclusion}}$	$\mathbf{R_{Switch,best}}$	$\mathbf{t_{sw}^{best}}$
Ackley	0.05	**3.273E-02 ± 2.864E-03**	3.407E-02 ± 2.496E-02	70
	0.1	**3.518E-02 ± 2.460E-03**	3.644E-02 ± 2.186E-02	70
	0.15	4.721E-02 ± 3.418E-03	**4.710E-02 ± 3.014E-02**	70
	0.2	**4.347E-02 ± 3.631E-03**	4.428E-02 ± 3.220E-02	60
	0.3	4.943E-02 ± 4.383E-03	**4.346E-02 ± 3.374E-02**	30
Griewank	0.05	**5.070E-03 ± 4.782E-04**	5.654E-03 ± 3.830E-03	70
	0.1	**5.500E-03 ± 4.590E-04**	5.711E-03 ± 3.440E-03	70
	0.15	**5.127E-03 ± 5.156E-04**	5.379E-03 ± 3.898E-03	70
	0.2	6.603E-03 ± 4.571E-04	**5.991E-03 ± 4.021E-03**	30
	0.3	6.582E-03 ± 4.843E-04	**6.303E-03 ± 3.666E-03**	20
michalewicz	0.05	1.254E-03 ± 1.368E-04	**1.185E-03 ± 8.698E-04**	70
	0.1	8.532E-04 ± 1.044E-04	**8.484E-04 ± 8.051E-04**	60
	0.15	8.798E-04 ± 1.121E-04	**8.757E-04 ± 8.629E-04**	70
	0.2	9.606E-04 ± 1.035E-04	**9.529E-04 ± 7.588E-04**	40
	0.3	1.121E-02 ± 9.908E-03	**1.092E-03 ± 9.843E-04**	10
Rastrigin	0.05	**8.379E-01 ± 1.103E-01**	9.153E-01 ± 1.034E+00	70
	0.1	**6.584E-01 ± 7.857E-02**	7.521E-01 ± 8.274E-01	70
	0.15	**9.065E-01 ± 8.898E-02**	1.004E+00 ± 8.161E-01	50
	0.2	**1.018E+00 ± 1.068E-01**	1.114E+00 ± 1.045E+00	60
	0.3	1.192E+00 ± 1.260E-01	**1.152E+00 ± 1.083E+00**	40
Styblinsky	0.05	2.143E-01 ± 5.167E-02	**6.823E-02 ± 7.255E-02**	10
	0.1	2.180E-01 ± 1.231E-01	**8.769E-02 ± 6.762E-02**	70
	0.15	**7.358E-02 ± 1.210E-02**	9.361E-02 ± 8.053E-02	10
	0.2	**9.177E-02 ± 2.575E-02**	1.106E-01 ± 9.397E-02	50
	0.3	1.273E-01 ± 3.171E-02	**1.226E-01 ± 8.488E-02**	10

Table A.3: Final **simple** average regrets in the **high** error regime. Results for the exclusion radius method radius are compared to the best results for the early switch method, obtained for tbest t_{best}. Values in bold indicate the lowest regret at each value of r/l. The absolute best result for each benchmark is underlined.

A.1.4 Predicted regrets at low error regime, for the early switch method

LOW ERROR				
Function	**r/l**	$\mathbf{R_{exclusion}}$	$\mathbf{R_{Switch,best}}$	$\mathbf{t_{sw}^{best}}$
Ackley	0.05	1.529E-02 ± 1.180E-03	**1.529E-02 ± 1.180E-03**	70
	0.1	1.761E-02 ± 1.308E-03	**1.601E-02 ± 1.189E-03**	10
	0.15	1.674E-02 ± 1.277E-03	**1.597E-02 ± 1.271E-03**	20
	0.2	1.653E-02 ± 1.248E-03	**1.601E-02 ± 1.131E-03**	30
	0.3	**1.690E-02 ± 1.289E-03**	1.733E-02 ± 1.421E-03	40
Griewank	0.05	**2.600E-03 ± 5.145E-04**	2.741E-03 ± 5.237E-04	70
	0.1	**3.583E-03 ± 5.468E-04**	3.745E-03 ± 5.650E-04	30
	0.15	4.433E-03 ± 5.868E-04	**4.104E-03 ± 5.795E-04**	20
	0.2	4.751E-03 ± 5.421E-04	**4.252E-03 ± 5.755E-04**	20
	0.3	5.074E-03 ± 5.080E-04	**4.906E-03 ± 5.257E-04**	30
michalewicz	0.05	**1.503E-04 ± 2.567E-05**	1.629E-04 ± 2.709E-05	30
	0.1	**1.644E-04 ± 2.760E-05**	1.701E-04 ± 2.745E-05	70
	0.15	**2.062E-04 ± 4.450E-05**	2.118E-04 ± 4.430E-05	30
	0.2	**1.673E-04 ± 2.845E-05**	1.730E-04 ± 2.829E-05	50
	0.3	**2.260E-04 ± 5.153E-05**	2.279E-04 ± 4.909E-05	30
Rastrigin	0.05	**4.535E-01 ± 1.143E-01**	5.504E-01 ± 1.186E-01	70
	0.1	**5.288E-01 ± 1.309E-01**	5.994E-01 ± 1.095E-01	10
	0.15	**5.853E-01 ± 9.150E-02**	6.753E-01 ± 1.312E-01	70
	0.2	**5.204E-01 ± 9.139E-02**	5.229E-01 ± 9.194E-02	70
	0.3	7.569E-01 ± 1.144E-01	**6.774E-01 ± 1.054E-01**	60
Styblinsky	0.05	7.923E-02 ± 5.193E-02	**9.074E-04 ± 2.106E-04**	20
	0.1	9.200E-04 ± 1.523E-04	**8.452E-04 ± 1.713E-04**	10
	0.15	1.059E-03 ± 2.186E-04	**8.097E-04 ± 1.644E-04**	30
	0.2	1.485E-03 ± 3.336E-04	**9.336E-04 ± 1.995E-04**	30
	0.3	1.644E-03 ± 4.116E-04	**9.141E-04 ± 1.627E-04**	20

Table A.4: Final **predicted** average regrets in the **low** error regime. Results for the exclusion radius method radius are compared to the best results for the early switch method, obtained for tbest t_{best}. Values in bold indicate the lowest regret at each value of r/l. The absolute best result for each benchmark is underlined.

A.1.5 Predicted regrets at medium error regime, for the early switch method

MEDIUM ERROR				
Function	**r/l**	$\mathbf{R_{exclusion}}$	$\mathbf{R_{Switch,best}}$	$\mathbf{t_{sw}^{best}}$
Ackley	0.05	**1.637E-02 ± 1.077E-03**	1.662E-02 ± 1.081E-03	70
	0.1	1.733E-02 ± 1.140E-03	**1.653E-02 ± 1.093E-03**	20
	0.15	1.727E-02 ± 1.161E-03	**1.642E-02 ± 1.149E-03**	20
	0.2	1.656E-02 ± 1.130E-03	**1.595E-02 ± 1.046E-03**	60
	0.3	1.647E-02 ± 1.088E-03	**1.646E-02 ± 1.088E-03**	60
Griewank	0.05	**2.872E-03 ± 4.911E-04**	3.346E-03 ± 5.065E-04	70
	0.1	**4.265E-03 ± 5.181E-04**	4.756E-03 ± 5.188E-04	70
	0.15	**4.313E-03 ± 5.056E-04**	4.816E-03 ± 5.010E-04	70
	0.2	6.028E-03 ± 7.072E-04	**4.757E-03 ± 5.330E-04**	10
	0.3	**4.685E-03 ± 5.146E-04**	4.783E-03 ± 5.268E-04	40
michalewicz	0.05	**1.494E-04 ± 2.039E-05**	1.545E-04 ± 2.032E-05	40
	0.1	**1.518E-04 ± 1.854E-05**	1.569E-04 ± 1.846E-05	60
	0.15	**1.563E-04 ± 2.027E-05**	1.614E-04 ± 2.018E-05	50
	0.2	**1.631E-04 ± 2.272E-05**	1.682E-04 ± 2.261E-05	50
	0.3	1.819E-04 ± 2.933E-05	**1.703E-04 ± 2.132E-05**	40
Rastrigin	0.05	5.732E-01 ± 1.111E-01	**5.183E-01 ± 7.934E-02**	20
	0.1	**6.350E-01 ± 1.082E-01**	7.790E-01 ± 1.249E-01	70
	0.15	**3.728E-01 ± 9.025E-02**	4.523E-01 ± 1.011E-01	70
	0.2	**4.897E-01 ± 8.226E-02**	5.033E-01 ± 8.677E-02	60
	0.3	**6.950E-01 ± 1.417E-01**	7.317E-01 ± 1.283E-01	10
Styblinsky	0.05	1.211E-01 ± 8.281E-02	**1.209E-03 ± 2.502E-04**	10
	0.1	1.633E-03 ± 4.786E-04	**1.019E-03 ± 2.457E-04**	20
	0.15	**1.073E-03 ± 1.558E-04**	1.212E-03 ± 2.586E-04	40
	0.2	1.355E-03 ± 2.148E-04	**1.178E-03 ± 2.882E-04**	40
	0.3	**1.422E-03 ± 4.017E-04**	1.602E-03 ± 4.221E-04	10

Table A.5: Final **predicted** average regrets in the **medium** error regime. Results for the exclusion radius method radius are compared to the best results for the early switch method, obtained for tbest t_{best}. Values in bold indicate the lowest regret at each value of r/l. The absolute best result for each benchmark is underlined.

A.1.6 Predicted regrets at high error regime, for the early switch method

HIGH ERROR				
Function	**r/l**	$\mathbf{R_{exclusion}}$	$\mathbf{R_{Switch,best}}$	$\mathbf{t_{sw}^{best}}$
Ackley	0.05	**1.594E-02** ± **9.688E-04**	1.604E-02 ± 9.780E-04	70
	0.1	**1.620E-02** ± **9.993E-04**	1.635E-02 ± 1.044E-03	50
	0.15	1.682E-02 ± 1.054E-03	**1.617E-02** ± **1.020E-03**	40
	0.2	**1.681E-02** ± **1.114E-03**	1.687E-02 ± 1.103E-03	40
	0.3	1.695E-02 ± 1.077E-03	**1.687E-02** ± **1.049E-03**	10
Griewank	0.05	**4.059E-03** ± **4.977E-04**	4.141E-03 ± 4.941E-04	70
	0.1	**4.024E-03** ± **4.887E-04**	4.166E-03 ± 4.887E-04	70
	0.15	**3.874E-03** ± **4.856E-04**	4.128E-03 ± 4.834E-04	70
	0.2	**4.472E-03** ± **4.869E-04**	4.508E-03 ± 5.611E-04	30
	0.3	**4.927E-03** ± **4.918E-04**	5.117E-03 ± 4.882E-04	20
michalewicz	0.05	2.499E-04 ± 6.124E-05	**1.775E-04** ± **2.661E-05**	30
	0.1	**1.726E-04** ± **2.858E-05**	1.739E-04 ± 2.855E-05	30
	0.15	**1.718E-04** ± **2.895E-05**	1.731E-04 ± 2.892E-05	40
	0.2	**1.722E-04** ± **2.614E-05**	1.735E-04 ± 2.611E-05	30
	0.3	1.015E-02 ± 9.901E-03	**1.956E-04** ± **3.123E-05**	20
Rastrigin	0.05	7.678E-01 ± 1.072E-01	**7.360E-01** ± **9.250E-02**	40
	0.1	**5.782E-01** ± **7.891E-02**	6.543E-01 ± 9.224E-02	70
	0.15	7.804E-01 ± 8.544E-02	**7.146E-01** ± **7.239E-02**	10
	0.2	8.069E-01 ± 9.987E-02	**7.892E-01** ± **8.556E-02**	40
	0.3	8.857E-01 ± 1.126E-01	**7.864E-01** ± **1.014E-01**	40
Styblinsky	0.05	9.232E-02 ± 4.819E-02	**1.124E-03** ± **2.622E-04**	10
	0.1	8.896E-02 ± 8.333E-02	**1.270E-03** ± **2.567E-04**	20
	0.15	1.060E-03 ± 1.850E-04	**9.303E-04** ± **1.941E-04**	10
	0.2	1.089E-03 ± 1.875E-04	**1.082E-03** ± **2.547E-04**	40
	0.3	**7.434E-04** ± **1.803E-04**	9.305E-04 ± 1.880E-04	10

Table A.6: Final **predicted** average regrets in the **high** error regime. Results for the exclusion radius method radius are compared to the best results for the early switch method, obtained for tbest t_{best}. Values in bold indicate the lowest regret at each value of r/l. The absolute best result for each benchmark is underlined.

A.2 The Adaptive Radius Exclusion Method

A.2.1 Simple regrets at low error regime, for adaptive radius exclusion method

LOW ERROR				
Function	**r/l**	$\mathbf{R_{exclusion}}$	$\mathbf{R_{Adaptive,best}}$	$\mathbf{t^*_{best}}$
Ackley	0.05	**3.282E-02 ± 4.084E-03**	4.545E-02 ± 5.884E-03	80
	0.1	3.352E-02 ± 3.663E-03	-	-
	0.15	4.305E-02 ± 4.743E-03	-	-
	0.2	4.606E-02 ± 5.070E-03	-	-
	0.3	5.539E-02 ± 6.053E-03	-	-
Griewank	0.05	**3.986E-03 ± 5.059E-04**	5.599E-03 ± 5.954E-04	65
	0.1	5.045E-03 ± 5.035E-04	-	-
	0.15	6.127E-03 ± 6.181E-04	-	-
	0.2	6.271E-03 ± 5.354E-04	-	-
	0.3	6.757E-03 ± 5.284E-04	-	-
Michalewicz	0.05	8.460E-04 ± 1.006E-04	8.645E-04 ± 1.225E-04	65
	0.1	**7.162E-04 ± 1.023E-04**	-	-
	0.15	9.485E-04 ± 1.244E-04	-	-
	0.2	1.190E-03 ± 1.427E-04	-	-
	0.3	1.420E-03 ± 1.542E-04	-	-
Rastrigin	0.05	**5.209E-01 ± 1.157E-01**	6.644E-01 ± 9.768E-02	75
	0.1	5.491E-01 ± 1.434E-01	-	-
	0.15	6.479E-01 ± 9.089E-02	-	-
	0.2	6.770E-01 ± 1.031E-01	-	-
	0.3	1.004E+00 ± 1.304E-01	-	-
Styblinski-Tang	0.05	1.555E-01 ± 5.335E-02	8.212E-02 ± 1.616E-02	80
	0.1	7.768E-02 ± 1.540E-02	-	-
	0.15	**6.892E-02 ± 1.523E-02**	-	-
	0.2	8.250E-02 ± 1.866E-02	-	-
	0.3	1.050E-01 ± 1.724E-02	-	-

Table A.7: Final average **simple** regrets comparison between exclusion radius method and the adaptive radius exclusion method in **low** noise regime. For the adaptive radius exclusion method only the best results are reported, together with the value of t^* at which they have been achieved. The best result for each benchmark function is in bold.

A.2.2 Simple regrets at medium error regime, for adaptive radius exclusion method

MEDIUM ERROR				
Function	**r/l**	$\mathbf{R_{exclusion}}$	$\mathbf{R_{Adaptive,best}}$	$\mathbf{t^*_{best}}$
Ackley	0.05	**3.697E-02 ± 3.155E-03**	4.838E-02 ± 3.861E-03	80
	0.1	3.757E-02 ± 3.145E-03	-	-
	0.15	4.189E-02 ± 3.100E-03	-	-
	0.2	4.534E-02 ± 3.477E-03	-	-
	0.3	4.988E-02 ± 4.095E-03	-	-
Griewank	0.05	**3.892E-03 ± 4.972E-04**	6.486E-03 ± 5.942E-04	25
	0.1	5.357E-03 ± 5.126E-04	-	-
	0.15	5.823E-03 ± 4.911E-04	-	-
	0.2	7.356E-03 ± 7.075E-04	-	-
	0.3	6.075E-03 ± 5.309E-04	-	-
Michalewicz	0.05	**6.505E-04 ± 7.988E-05**	7.467E-04 ± 8.124E-05	35
	0.1	7.247E-04 ± 8.407E-05	-	-
	0.15	9.158E-04 ± 1.178E-04	-	-
	0.2	7.819E-04 ± 8.558E-05	-	-
	0.3	1.097E-03 ± 1.109E-04	-	-
Rastrigin	0.05	6.299E-01 ± 1.133E-01	7.914E-01 ± 9.453E-02	80
	0.1	6.929E-01 ± 1.093E-01	-	-
	0.15	**5.244E-01 ± 9.302E-02**	-	-
	0.2	6.474E-01 ± 7.790E-02	-	-
	0.3	1.062E+00 ± 1.670E-01	-	-
Styblinski-Tang	0.05	2.042E-01 ± 9.778E-02	**6.710E-02 ± 1.733E-02**	65
	0.1	6.940E-02 ± 1.349E-02	-	-
	0.15	1.139E-01 ± 1.661E-02	-	-
	0.2	1.255E-01 ± 1.849E-02	-	-
	0.3	1.057E-01 ± 1.561E-02	-	-

Table A.8: Final average **simple** regrets comparison between exclusion radius method and the adaptive radius exclusion method in **medium** noise regime. For the adaptive radius exclusion method only the best results are reported, together with the value of t^* at which they have been achieved. The best result for each benchmark function is in bold.

A.2.3 Simple regrets at high error regime, for adaptive radius exclusion method

HIGH ERROR				
Function	**r/l**	$\mathbf{R_{exclusion}}$	$\mathbf{R_{Adaptive,best}}$	$\mathbf{t^*_{best}}$
Ackley	0.05	**3.273E-02 ± 2.864E-03**	4.650E-02 ± 4.011E-03	75
	0.1	3.518E-02 ± 2.460E-03	-	-
	0.15	4.721E-02 ± 3.418E-03	-	-
	0.2	4.347E-02 ± 3.631E-03	-	-
	0.3	4.943E-02 ± 4.383E-03	-	-
Griewank	0.05	**5.070E-03 ± 4.782E-04**	5.825E-03 ± 5.581E-04	75
	0.1	5.500E-03 ± 4.590E-04	-	-
	0.15	5.127E-03 ± 5.156E-04	-	-
	0.2	6.603E-03 ± 4.571E-04	-	-
	0.3	6.582E-03 ± 4.843E-04	-	-
Michalewicz	0.05	1.254E-03 ± 1.368E-04	1.274E-03 ± 1.292E-04	55
	0.1	**8.532E-04 ± 1.044E-04**	-	-
	0.15	8.798E-04 ± 1.121E-04	-	-
	0.2	9.606E-04 ± 1.035E-04	-	-
	0.3	1.121E-02 ± 9.908E-03	-	-
Rastrigin	0.05	8.379E-01 ± 1.103E-01	9.599E-01 ± 1.076E-01	80
	0.1	**6.584E-01 ± 7.857E-02**	-	-
	0.15	9.065E-01 ± 8.898E-02	-	-
	0.2	1.018E+00 ± 1.068E-01	-	-
	0.3	1.192E+00 ± 1.260E-01	-	-
Styblinski-Tang	0.05	2.143E-01 ± 5.167E-02	9.137E-02 ± 1.664E-02	75
	0.1	2.180E-01 ± 1.231E-01	-	-
	0.15	**7.358E-02 ± 1.210E-02**	-	-
	0.2	9.177E-02 ± 2.575E-02	-	-
	0.3	1.273E-01 ± 3.171E-02	-	-

Table A.9: Final average **simple** regrets comparison between exclusion radius method and the adaptive radius exclusion method in **high** noise regime. For the adaptive radius exclusion method only the best results are reported, together with the value of t^* at which they have been achieved. The best result for each benchmark function is in bold.

A.2.4 Predicted regrets at low error regime, for adaptive radius exclusion method

LOW ERROR				
Function	**r/l**	$\mathbf{R_{exclusion}}$	$\mathbf{R_{Adaptive,best}}$	$\mathbf{t^*_{best}}$
Ackley	0.05	**1.529E-02 ± 1.180E-03**	1.529E-02 ± 1.144E-03	25
	0.1	1.761E-02 ± 1.308E-03	-	-
	0.15	1.674E-02 ± 1.277E-03	-	-
	0.2	1.653E-02 ± 1.248E-03	-	-
	0.3	1.690E-02 ± 1.289E-03	-	-
Griewank	0.05	**2.600E-03 ± 5.145E-04**	3.958E-03 ± 5.642E-04	65
	0.1	3.583E-03 ± 5.468E-04	-	-
	0.15	4.433E-03 ± 5.868E-04	-	-
	0.2	4.751E-03 ± 5.421E-04	-	-
	0.3	5.074E-03 ± 5.080E-04	-	-
Michalewicz	0.05	**1.503E-04 ± 2.567E-05**	1.847E-04 ± 2.943E-05	65
	0.1	1.644E-04 ± 2.760E-05	-	-
	0.15	2.062E-04 ± 4.450E-05	-	-
	0.2	1.673E-04 ± 2.845E-05	-	-
	0.3	2.260E-04 ± 5.153E-05	-	-
Rastrigin	0.05	4.535E-01 ± 1.143E-01	**4.418E-01 ± 8.848E-02**	75
	0.1	5.288E-01 ± 1.309E-01	-	-
	0.15	5.853E-01 ± 9.150E-02	-	-
	0.2	5.204E-01 ± 9.139E-02	-	-
	0.3	7.569E-01 ± 1.144E-01	-	-
Styblinski-Tang	0.05	7.923E-02 ± 5.193E-02	**8.480E-04 ± 1.817E-04**	80
	0.1	9.200E-04 ± 1.523E-04	-	-
	0.15	1.059E-03 ± 2.186E-04	-	-
	0.2	1.485E-03 ± 3.336E-04	-	-
	0.3	1.644E-03 ± 4.116E-04	-	-

Table A.10: Final average **predicted** regrets comparison between exclusion radius method and the adaptive radius exclusion method in **high** noise regime. For the adaptive radius exclusion method only the best results are reported, together with the value of t^* at which they have been achieved. The best result for each benchmark function is in bold.

A.2.5 Predicted regrets at medium error regime, for adaptive radius exclusion method

MEDIUM ERROR				
Function	**r/l**	$\mathbf{R_{exclusion}}$	$\mathbf{R_{Adaptive,best}}$	$\mathbf{t^*_{best}}$
Ackley	0.05	**1.637E-02 ± 1.077E-03**	1.671E-02 ± 1.144E-03	75
	0.1	1.733E-02 ± 1.140E-03	-	-
	0.15	1.727E-02 ± 1.161E-03	-	-
	0.2	1.656E-02 ± 1.130E-03	-	-
	0.3	1.647E-02 ± 1.088E-03	-	-
Griewank	0.05	**2.872E-03 ± 4.911E-04**	4.934E-03 ± 6.005E-04	80
	0.1	4.265E-03 ± 5.181E-04	-	-
	0.15	4.313E-03 ± 5.056E-04	-	-
	0.2	6.028E-03 ± 7.072E-04	-	-
	0.3	4.685E-03 ± 5.146E-04	-	-
Michalewicz	0.05	**1.494E-04 ± 2.039E-05**	1.499E-04 ± 1.771E-05	25
	0.1	1.518E-04 ± 1.854E-05	-	-
	0.15	1.563E-04 ± 2.027E-05	-	-
	0.2	1.631E-04 ± 2.272E-05	-	-
	0.3	1.819E-04 ± 2.933E-05	-	-
Rastrigin	0.05	5.732E-01 ± 1.111E-01	4.973E-01 ± 8.281E-02	25
	0.1	6.350E-01 ± 1.082E-01	-	-
	0.15	**3.728E-01 ± 9.025E-02**	-	-
	0.2	4.897E-01 ± 8.226E-02	-	-
	0.3	6.950E-01 ± 1.417E-01	-	-
Styblinski-Tang	0.05	1.211E-01 ± 8.281E-02	**1.061E-03 ± 2.826E-04**	35
	0.1	1.633E-03 ± 4.786E-04	-	-
	0.15	1.073E-03 ± 1.558E-04	-	-
	0.2	1.355E-03 ± 2.148E-04	-	-
	0.3	1.422E-03 ± 4.017E-04	-	-

Table A.11: Final average **predicted** regrets comparison between exclusion radius method and the adaptive radius exclusion method in **medium** noise regime. For the adaptive radius exclusion method only the best results are reported, together with the value of t^* at which they have been achieved. The best result for each benchmark function is in bold.

A.2.6 Predicted regrets at high error regime, for adaptive radius exclusion method

HIGH ERROR				
Function	**r/l**	$\mathbf{R_{exclusion}}$	$\mathbf{R_{Adaptive,best}}$	$\mathbf{t^*_{best}}$
Ackley	0.05	**1.594E-02 ± 9.688E-04**	1.672E-02 ± 1.025E-03	65
	0.1	1.620E-02 ± 9.993E-04	-	-
	0.15	1.682E-02 ± 1.054E-03	-	-
	0.2	1.681E-02 ± 1.114E-03	-	-
	0.3	1.695E-02 ± 1.077E-03	-	-
Griewank	0.05	4.059E-03 ± 4.977E-04	4.729E-03 ± 5.270E-04	75
	0.1	4.024E-03 ± 4.887E-04	-	-
	0.15	**3.874E-03 ± 4.856E-04**	-	-
	0.2	4.472E-03 ± 4.869E-04	-	-
	0.3	4.927E-03 ± 4.918E-04	-	-
Michalewicz	0.05	2.499E-04 ± 6.124E-05	**1.703E-04 ± 2.374E-05**	55
	0.1	1.726E-04 ± 2.858E-05	-	-
	0.15	1.718E-04 ± 2.895E-05	-	-
	0.2	1.722E-04 ± 2.614E-05	-	-
	0.3	1.015E-02 ± 9.901E-03	-	-
Rastrigin	0.05	7.678E-01 ± 1.072E-01	6.948E-01 ± 1.158E-01	55
	0.1	**5.782E-01 ± 7.891E-02**	-	-
	0.15	7.804E-01 ± 8.544E-02	-	-
	0.2	8.069E-01 ± 9.987E-02	-	-
	0.3	8.857E-01 ± 1.126E-01	-	-
Styblinski-Tang	0.05	9.232E-02 ± 4.819E-02	1.112E-03 ± 2.280E-04	35
	0.1	8.896E-02 ± 8.333E-02	-	-
	0.15	1.060E-03 ± 1.850E-04	-	-
	0.2	1.089E-03 ± 1.875E-04	-	-
	0.3	**7.434E-04 ± 1.803E-04**	-	-

Table A.12: Final average **predicted** regrets comparison between exclusion radius method and the adaptive radius exclusion method in **high** noise regime. For the adaptive radius exclusion method only the best results are reported, together with the value of t^* at which they have been achieved. The best result for each benchmark function is in bold.

Appendix B

Regrets for the discrepancy prediction method with augmented real data

B.1 Average regrets for discrepancy prediction method with augmented real data

B.1.1 Simple average regrets using a GP to predict the discrepancy

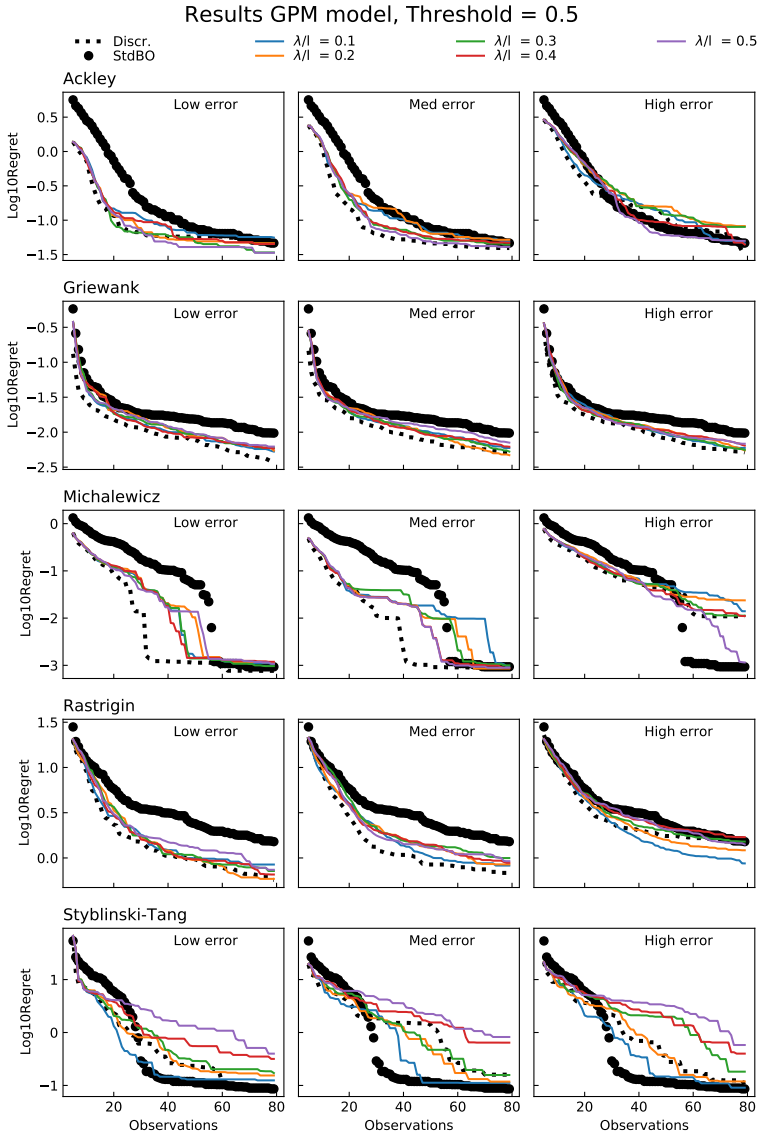

Figure B.1: Average **simple** regrets for all the benchmark functions, using a **GP** surrogate model and a sampling threshold **T=0.5**. Each curve corresponds to a different value of the ratio λ/l. The black dotted curve refers to the discrepancy prediction method without any additional real data.

B.1.2 Simple average regrets using a GBR to predict the discrepancy

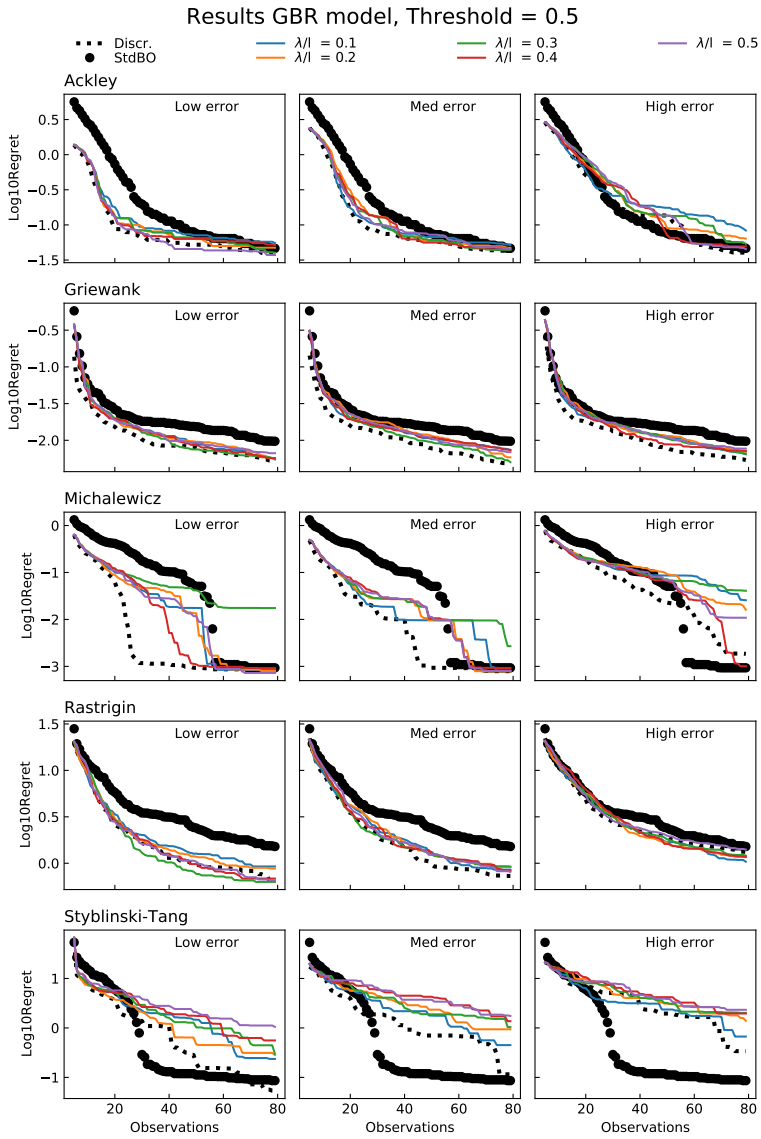

Figure B.2: Average **simple** regrets for all the benchmark functions, using a **GBR** surrogate model and a sampling threshold **T=0.5**. Each curve corresponds to a different value of the ratio λ/l. The black dotted curve refers to the discrepancy prediction method without any additional real data.

B.1.3 Predicted average regrets using a GP to predict the discrepancy

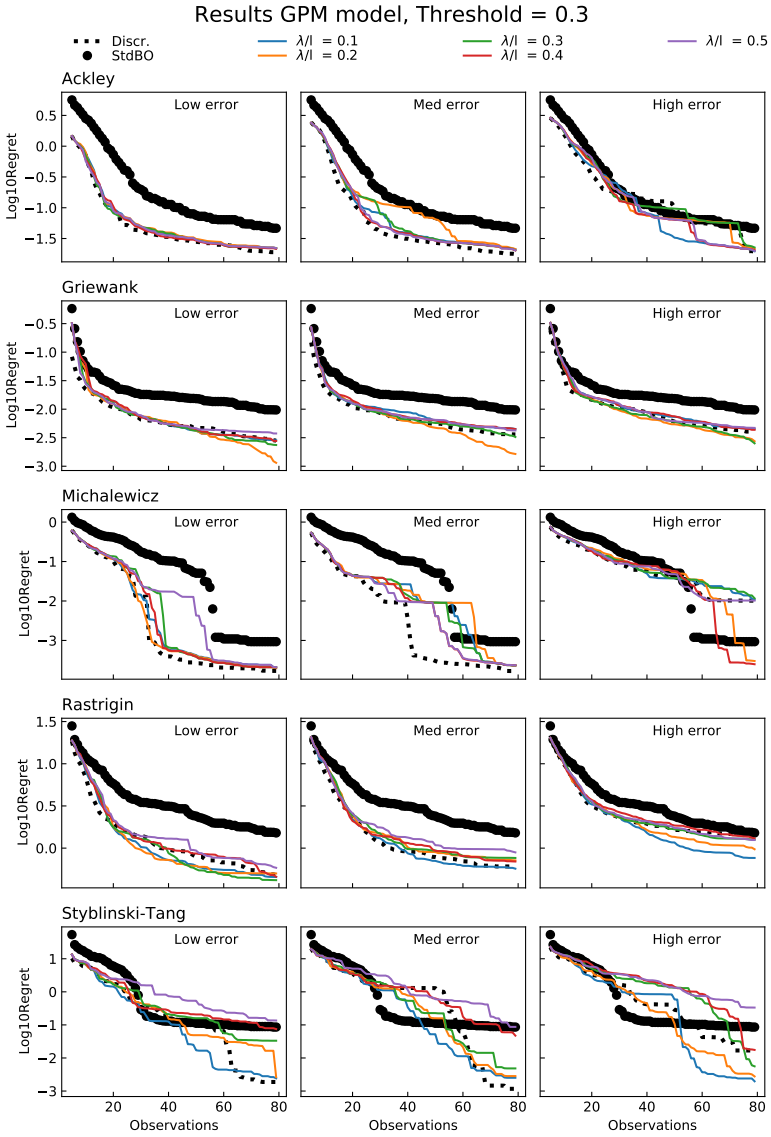

Figure B.3: Average **predicted** regrets for all the benchmark functions, using a **GP** surrogate model and a sampling threshold **T=0.3**. Each curve corresponds to a different value of the ratio λ/l. The black dotted curve refers to the discrepancy prediction method without any additional real data.

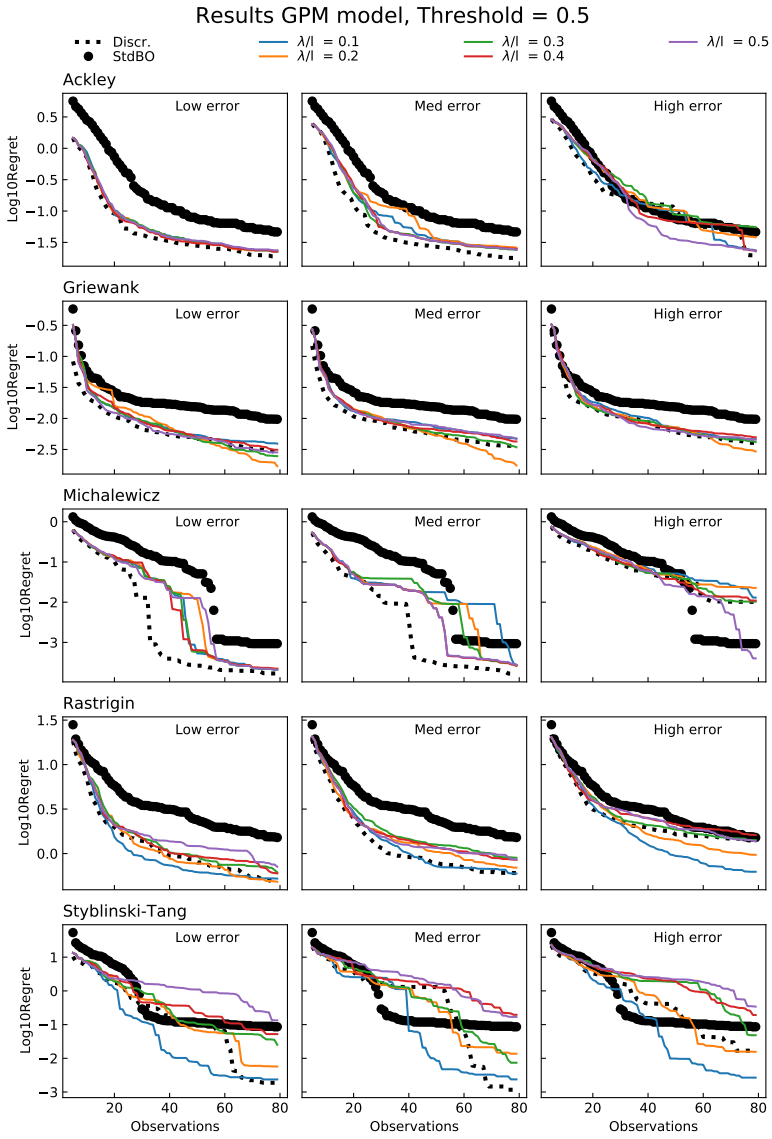

Figure B.4: Average **predicted** regrets for all the benchmark functions, using a **GP** surrogate model and a sampling threshold **T=0.5**. Each curve corresponds to a different value of the ratio λ/l. The black dotted curve refers to the discrepancy prediction method without any additional real data.

B.1.4 Predicted average regrets using a GBR to predict the discrepancy

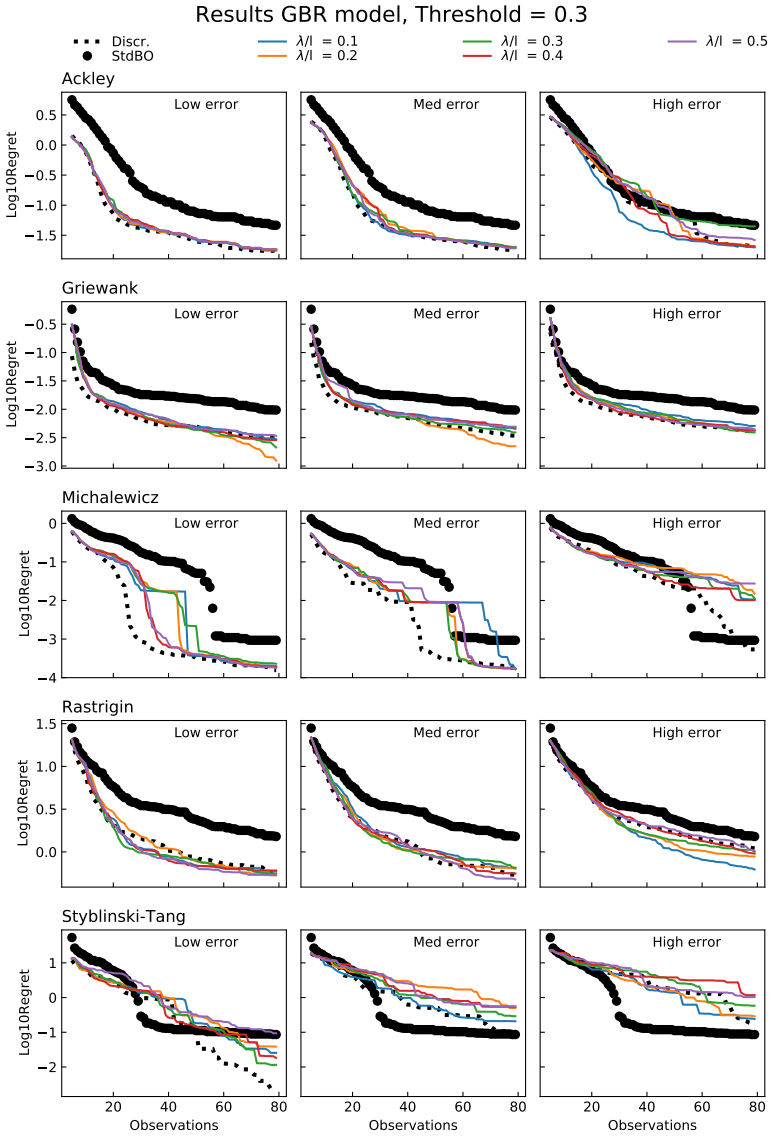

Figure B.5: Average **predicted** regrets for all the benchmark functions, using a **GBR** surrogate model and a sampling threshold **T=0.3**. Each curve corresponds to a different value of the ratio λ/l. The black dotted curve refers to the discrepancy prediction method without any additional real data.

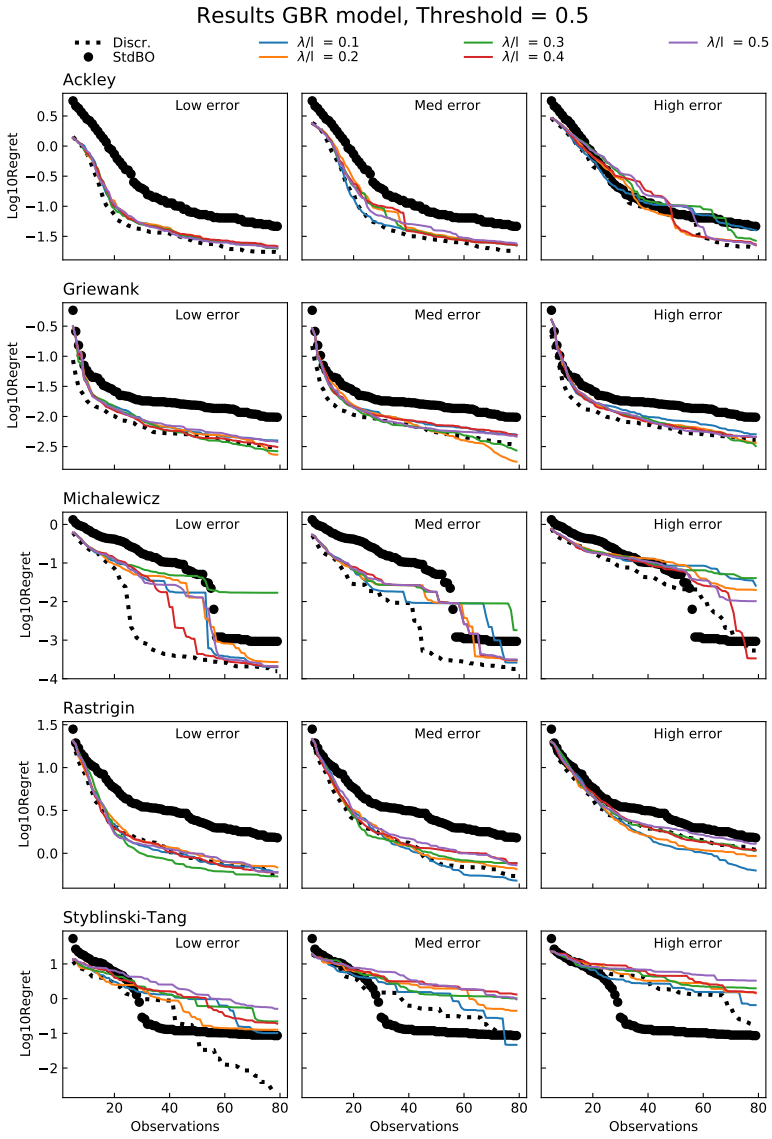

Figure B.6: Average **predicted** regrets for all the benchmark functions, using a **GBR** surrogate model and a sampling threshold **T=0.5**. Each curve corresponds to a different value of the ratio λ/l. The black dotted curve refers to the discrepancy prediction method without any additional real data.

B.2 Final regrets for discrepancy prediction method with augmented real data

B.2.1 Simple final regrets using a GP to predict the discrepancy

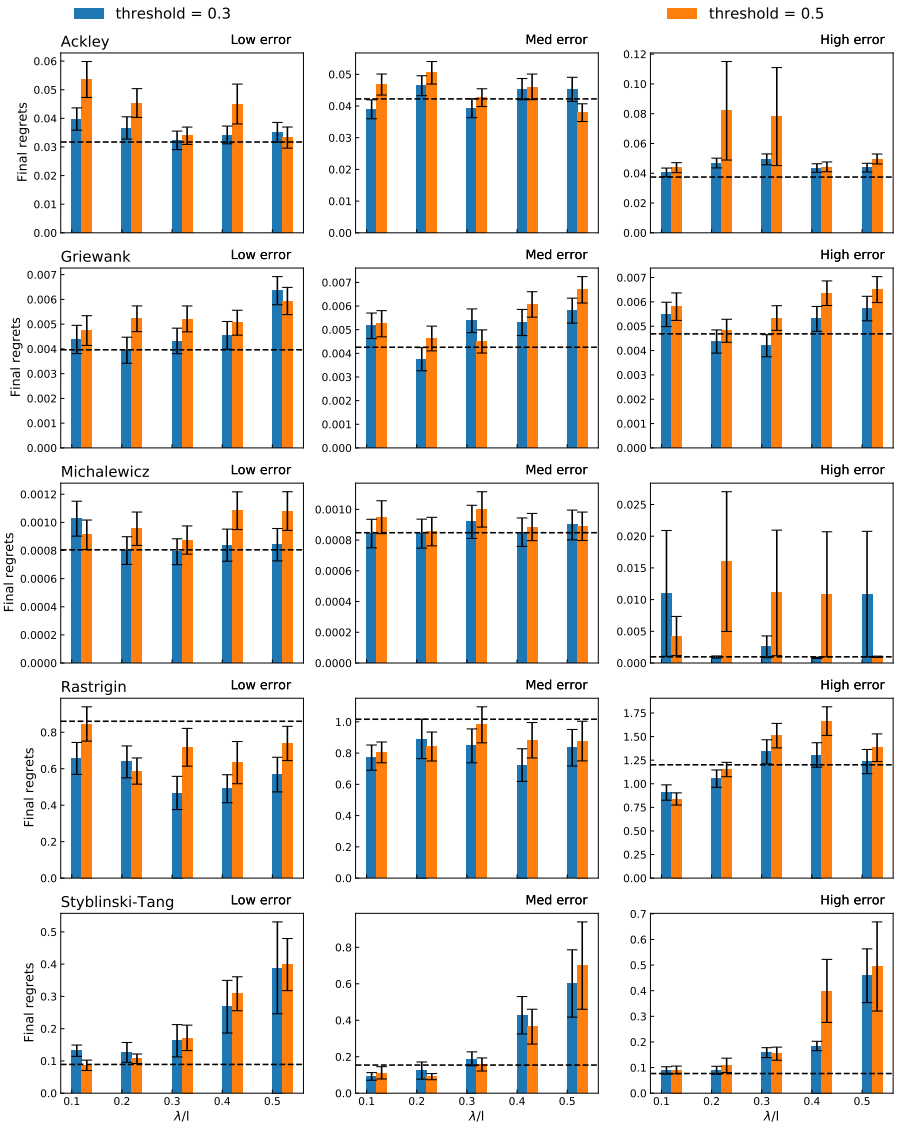

Figure B.7: Comparison between averaged final **simple** regrets $\hat{R}_s$ achieved with $T = 0.3$ or $T = 0.5$, using a **GP** surrogate model for the discrepancy. The values of $\hat{R}_p$ are plotted versus the value of λ/l. The black dashed line shows the averaged final **predicted** regret observed for the discrepancy prediction method.

B.2.2 Simple final regrets using a GBR to predict the discrepancy

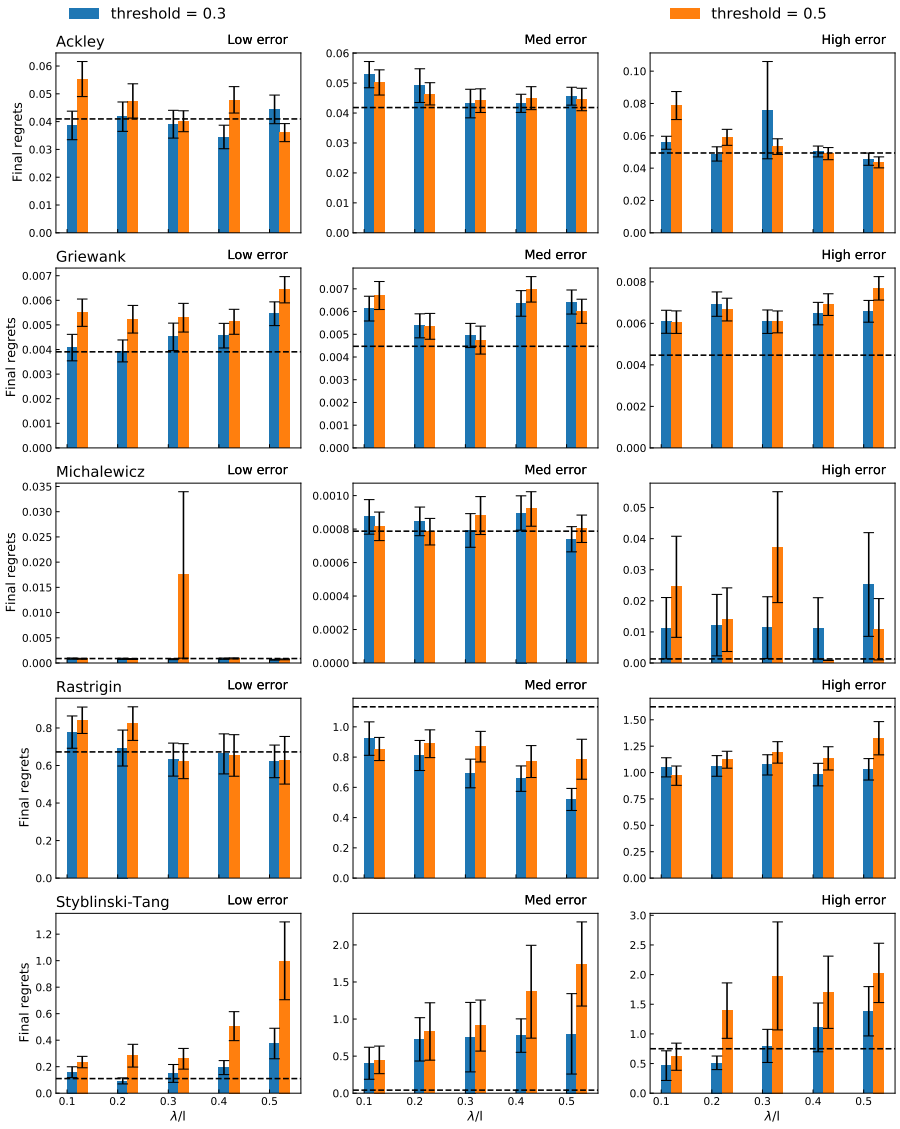

Figure B.8: Comparison between averaged final **simple** regrets $\hat{R}_s$ achieved with $T = 0.3$ or $T = 0.5$, using a **GBR** surrogate model for the discrepancy. The values of $\hat{R}_p$ are plotted versus the value of λ/l. The black dashed line shows the averaged final **predicted** regret observed for the discrepancy prediction method.

B.2.3 Predicted final regrets using a GP to predict the discrepancy

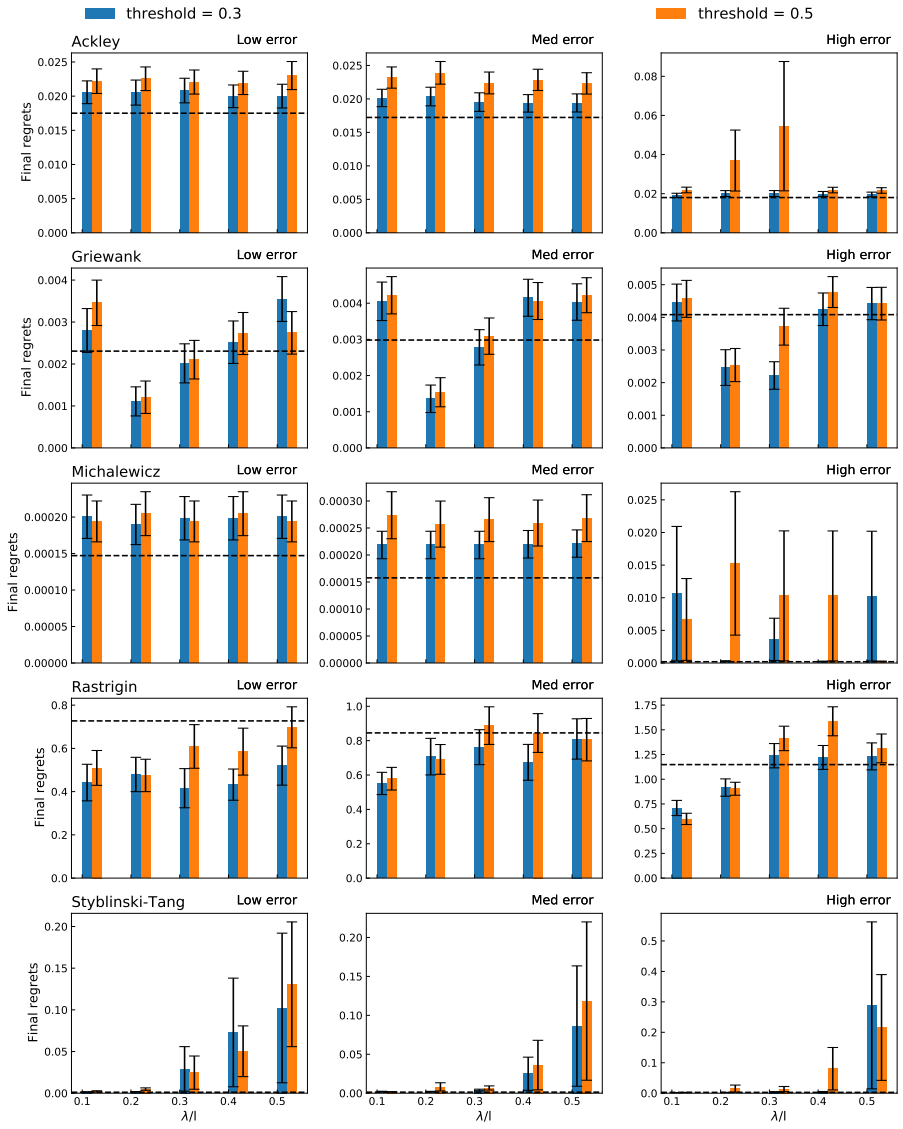

Figure B.9: Comparison between averaged final **predicted** regrets $\hat{R}_p$ achieved with $T = 0.3$ or $T = 0.5$, using a **GP** surrogate model for the discrepancy. The values of $\hat{R}_p$ are plotted versus the value of λ/l. The black dashed line shows the averaged final **predicted** regret observed for the discrepancy prediction method.

B.2.4 Predicted final regrets using a GBR to predict the discrepancy

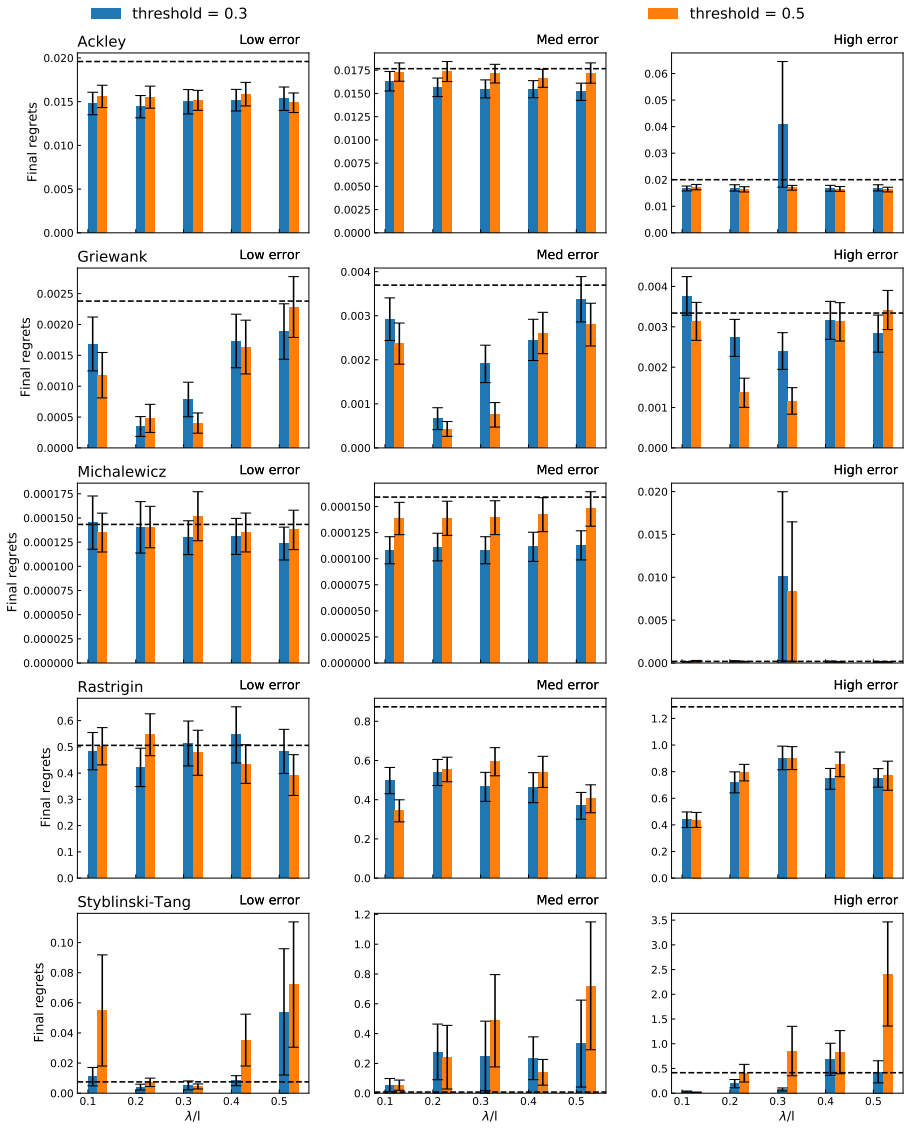

Figure B.10: Comparison between averaged final **predicted** regrets $\hat{R}_p$ achieved with $T = 0.3$ or $T = 0.5$, using a **GBR** surrogate model for the discrepancy. The values of $\hat{R}_p$ are plotted versus the value of λ/l. The black dashed line shows the averaged final **predicted** regret observed for the discrepancy prediction method.

Bibliography

[1] https://bnaic2021.uni.lu/.

[2] Apoorv Agnihotri and Nipun Batra. "Exploring Bayesian Optimization". In: *Distill* (2020). https://distill.pub/2020/bayesian-optimization. DOI: 10.23915/distill.00026.

[3] Shipra Agrawal and Navin Goyal. "Analysis of thompson sampling for the multi-armed bandit problem". In: *Conference on learning theory*. JMLR Workshop and Conference Proceedings. 2012, pp. 39–1.

[4] Jean-Yves Audibert and Sébastien Bubeck. "Regret bounds and minimax policies under partial monitoring". In: *The Journal of Machine Learning Research* 11 (2010), pp. 2785–2836.

[5] Jean-Yves Audibert, Rémi Munos, and Csaba Szepesvári. "Tuning bandit algorithms in stochastic environments". In: *International conference on algorithmic learning theory*. Springer. 2007, pp. 150–165.

[6] Peter Auer. "Using confidence bounds for exploitation-exploration trade-offs". In: *Journal of Machine Learning Research* 3.Nov (2002), pp. 397–422.

[7] The GPyOpt authors. *GPyOpt: A Bayesian Optimization framework in Python*. http://github.com/SheffieldML/GPyOpt. 2016.

[8] Maximilian Balandat et al. "BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization". In: *Advances in Neural Information Processing Systems 33*. 2020. URL: http://arxiv.org/abs/1910.06403.

[9] Albert P Bartók, Risi Kondor, and Gábor Csányi. "On representing chemical environments". In: *Physical Review B* 87.18 (2013), p. 184115.

[10] Daniel Beck and Trevor Cohn. "Learning kernels over strings using Gaussian processes". In: *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. 2017, pp. 67–73.

[11] Jörg Behler. "Four generations of high-dimensional neural network potentials". In: *Chemical Reviews* 121.16 (2021), pp. 10037–10072.

[12] Jörg Behler. "Perspective: Machine learning potentials for atomistic simulations". In: *The Journal of chemical physics* 145.17 (2016), p. 170901.

[13] James O Berger. *Statistical decision theory and Bayesian analysis*. Springer Science & Business Media, 2013.

[14] Christopher M Bishop and Nasser M Nasrabadi. "Pattern recognition and machine learning". In: vol. 4. 4. Springer, 2006.

[15] Carl Ivar Branden and John Tooze. *Introduction to protein structure*. Garland Science, 2012.

[16] Benjamin Burger et al. "A mobile robotic chemist". In: *Nature* 583.7815 (2020), pp. 237–241.

[17] Jeffrey M Buth et al. "Kinetics and Pathways of the Aqueous Photolysis of Pharmaceutical Pollutants: A Versatile Laboratory or Remote Learning Investigation". In: *Journal of Chemical Education* 98.7 (2021), pp. 2411–2418.

[18] Dario Caramelli et al. "Discovering new chemistry with an autonomous robotic platform driven by a reactivity-seeking neural network". In: *ACS Central Science* 7.11 (2021), pp. 1821–1830.

[19] Miguel A Caro. "Optimizing many-body atomic descriptors for enhanced computational performance of machine learning based interatomic potentials". In: *Physical Review B* 100.2 (2019), p. 024112.

[20] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794.

[21] Benoit Choffin and Naonori Ueda. "Scaling Bayesian optimization up to higher dimensions: a review and comparison of recent algorithms". In: *2018 IEEE 28th international workshop on machine learning for signal processing (MLSP)*. IEEE. 2018, pp. 1–6.

[22] Haskell B Curry. "The method of steepest descent for non-linear minimization problems". In: *Quarterly of Applied Mathematics* 2.3 (1944), pp. 258–261.

[23] Sandip De et al. "Comparing molecules and solids across structural and alchemical space". In: *Physical Chemistry Chemical Physics* 18.20 (2016), pp. 13754–13769.

[24] Daniel C Elton et al. "Applying machine learning techniques to predict the properties of energetic materials". In: *Scientific reports* 8.1 (2018), p. 9059.

[25] David Eriksson and Martin Jankowiak. "High-dimensional Bayesian optimization with sparse axis-aligned subspaces". In: *Uncertainty in Artificial Intelligence*. PMLR. 2021, pp. 493–503.

[26] David Eriksson et al. “Scalable global optimization via local bayesian optimization”. In: *Advances in neural information processing systems* 32 (2019).

[27] Hatem Fakhruldeen et al. “Archemist: Autonomous robotic chemistry system architecture”. In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 6013–6019.

[28] De-Cheng Feng et al. “Interpretable XGBoost-SHAP machine-learning model for shear strength prediction of squat RC walls”. In: *Journal of Structural Engineering* 147.11 (2021), p. 04021173.

[29] Alexander IJ Forrester, András Sóbester, and Andy J Keane. “Multi-fidelity optimization via surrogate modelling”. In: *Proceedings of the royal society a: mathematical, physical and engineering sciences* 463.2088 (2007), pp. 3251–3269.

[30] Jerome H Friedman. “Greedy function approximation: a gradient boosting machine”. In: *Annals of statistics* (2001), pp. 1189–1232.

[31] Jerome H Friedman. “Stochastic gradient boosting”. In: *Computational statistics & data analysis* 38.4 (2002), pp. 367–378.

[32] Lukas P Fröhlich, Melanie N Zeilinger, and Edgar D Klenske. “Cautious bayesian optimization for efficient and scalable policy search”. In: *Learning for Dynamics and Control*. PMLR. 2021, pp. 227–240.

[33] Eduardo C Garrido-Merchán and Daniel Hernández-Lobato. “Dealing with categorical and integer-valued variables in bayesian optimization with gaussian processes”. In: *Neurocomputing* 380 (2020), pp. 20–35.

[34] Seyede Fatemeh Ghoreishi and Douglas Allaire. “Multi-information source constrained Bayesian optimization”. In: *Structural and Multidisciplinary Optimization* 59.3 (2019), pp. 977–991.

[35] Rafael Gómez-Bombarelli et al. “Automatic chemical design using a data-driven continuous representation of molecules”. In: *ACS central science* 4.2 (2018), pp. 268–276.

[36] Jie Gong et al. “XGBoost model for electrocaloric temperature change prediction in ceramics”. In: *npj Computational Materials* 8.1 (2022), p. 140.

[37] Javier González et al. “Batch bayesian optimization via local penalization”. In: *Artificial intelligence and statistics*. 2016, pp. 648–657.

[38] Matthew Groves and Edward O Pyzer-Knapp. “Efficient and scalable batch bayesian optimization using K-Means”. In: *arXiv preprint arXiv:1806.01159* (2018).

[39] Trevor Hastie et al. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer, 2009.

[40] Philipp Hennig and Christian J Schuler. "Entropy Search for Information-Efficient Global Optimization." In: *Journal of Machine Learning Research* 13.6 (2012).

[41] Henry C Herbol, Matthias Poloczek, and Paulette Clancy. "Cost-effective materials discovery: Bayesian optimization across multiple information sources". In: *Materials Horizons* (2020).

[42] José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. "Predictive entropy search for efficient global optimization of black-box functions". In: *arXiv preprint arXiv:1406.2541* (2014).

[43] José Miguel Hernández-Lobato et al. "Parallel and distributed Thompson sampling for large-scale accelerated exploration of chemical space". In: *International conference on machine learning*. PMLR. 2017, pp. 1470–1479.

[44] Lauri Himanen et al. "DScribe: Library of descriptors for machine learning in materials science". In: *Computer Physics Communications* 247 (2020), p. 106949. ISSN: 0010-4655. DOI: 10.1016/j.cpc.2019.106949. URL: https://doi.org/10.1016/j.cpc.2019.106949.

[45] John H Holland. "Genetic algorithms". In: *Scientific american* 267.1 (1992), pp. 66–73.

[46] Ruyun Hu et al. "Protein engineering via Bayesian optimization-guided evolutionary algorithm and robotic experiments". In: *Briefings in Bioinformatics* 24.1 (2023), bbac570.

[47] Deng Huang et al. "Sequential kriging optimization using multiple-fidelity evaluations". In: *Structural and Multidisciplinary Optimization* 32.5 (2006), pp. 369–382.

[48] Raban Iten. *Artificial Intelligence for Scientific Discoveries: Extracting Physical Concepts from Experimental Data Using Deep Learning*. Springer Nature, 2023.

[49] Marc OJ Jäger et al. "Machine learning hydrogen adsorption on nanoclusters through structural descriptors". In: *npj Computational Materials* 4.1 (2018), p. 37.

[50] Momin Jamil and Xin-She Yang. "A literature survey of benchmark functions for global optimisation problems". In: *International Journal of Mathematical Modelling and Numerical Optimisation* 4.2 (2013), pp. 150–194.

[51] Donald R Jones, Matthias Schonlau, and William J Welch. "Efficient global optimization of expensive black-box functions". In: *Journal of Global optimization* 13.4 (1998), pp. 455–492.

[52] Kirthevasan Kandasamy et al. "Gaussian process bandit optimisation with multi-fidelity evaluations". In: *Advances in Neural Information Processing Systems*. 2016, pp. 992–1000.

[53] Kirthevasan Kandasamy et al. "Parallelised Bayesian optimisation via Thompson sampling". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2018, pp. 133–142.

[54] Tarun Kathuria, Amit Deshpande, and Pushmeet Kohli. "Batched gaussian process bandit optimization via determinantal point processes". In: *Advances in Neural Information Processing Systems* 29 (2016).

[55] Guolin Ke et al. "Lightgbm: A highly efficient gradient boosting decision tree". In: *Advances in neural information processing systems* 30 (2017).

[56] James Kennedy and Russell Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95-international conference on neural networks*. Vol. 4. IEEE. 1995, pp. 1942–1948.

[57] Marc C Kennedy and Anthony O'Hagan. "Predicting the output from a complex computer code when fast approximations are available". In: *Biometrika* 87.1 (2000), pp. 1–13.

[58] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. "Optimization by simulated annealing". In: *science* 220.4598 (1983), pp. 671–680.

[59] Risi Imre Kondor and John Lafferty. "Diffusion kernels on graphs and other discrete structures". In: *Proceedings of the 19th international conference on machine learning*. Vol. 2002. 2002, pp. 315–322.

[60] Mario Krenn et al. "On scientific understanding with artificial intelligence". In: *Nature Reviews Physics* 4.12 (2022), pp. 761–769.

[61] Alex Kulesza and Ben Taskar. "Determinantal point processes for machine learning". In: *arXiv preprint arXiv:1207.6083* (2012).

[62] Alex Kulesza and Ben Taskar. "k-DPPs: Fixed-size determinantal point processes". In: *ICML*. 2011.

[63] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. "Grammar variational autoencoder". In: *International conference on machine learning*. PMLR. 2017, pp. 1945–1954.

[64] Rémi Lam, Douglas L Allaire, and Karen E Willcox. "Multifidelity optimization using statistical surrogate modeling for non-hierarchical information sources". In: *56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. 2015, p. 0143.

[65] Min-Hsuan Lee. "Robust random forest based non-fullerene organic solar cells efficiency prediction". In: *Organic Electronics* 76 (2020), p. 105465.

[66] Wei-Liem Loh et al. "On Latin hypercube sampling". In: *Annals of statistics* 24.5 (1996), pp. 2058–2080.

[67] Wenlong Lyu et al. "Batch Bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design". In: *International conference on machine learning*. PMLR. 2018, pp. 3306–3314.

[68] Mohit Malu, Gautam Dasarathy, and Andreas Spanias. "Bayesian optimization in high-dimensional spaces: A brief survey". In: *2021 12th International Conference on Information, Intelligence, Systems & Applications (IISA)*. IEEE. 2021, pp. 1–8.

[69] Randall Manteufel. "Evaluating the convergence of Latin hypercube sampling". In: *41st Structures, Structural Dynamics, and Materials Conference and Exhibit*, p. 1636.

[70] Alonso Marco et al. "Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with Bayesian optimization". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 1557–1563.

[71] Alexander G. de G. Matthews et al. "GPflow: A Gaussian process library using TensorFlow". In: *Journal of Machine Learning Research* 18.40 (Apr. 2017), pp. 1–6. URL: http://jmlr.org/papers/v18/16-537.html.

[72] Andrea Mauri, Viviana Consonni, and Roberto Todeschini. "Molecular Descriptors". In: *Handbook of Computational Chemistry*. Ed. by Jerzy Leszczynski et al. Cham: Springer International Publishing, 2017, pp. 2065–2093. ISBN: 978-3-319-27282-5. DOI: 10.1007/978-3-319-27282-5_51. URL: https://doi.org/10.1007/978-3-319-27282-5_51.

[73] Grégoire Montavon et al. "Learning invariant representations of molecules for atomization energy prediction". In: *Advances in neural information processing systems* 25 (2012).

[74] Henry Moss et al. "Boss: Bayesian optimization over string spaces". In: *Advances in neural information processing systems* 33 (2020), pp. 15476–15486.

[75] Henry B Moss et al. "Gibbon: General-purpose information-based bayesian optimisation". In: *The Journal of Machine Learning Research* 22.1 (2021), pp. 10616–10664.

[76] Alexey Natekin and Alois Knoll. "Gradient boosting machines, a tutorial". In: *Frontiers in neurorobotics* 7 (2013), p. 21.

[77] Elvis Nava, Mojmir Mutny, and Andreas Krause. "Diversified Sampling for Batched Bayesian Optimization with Determinantal Point Processes". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, pp. 7031–7054.

[78] John A Nelder and Roger Mead. "A simplex method for function minimization". In: *The computer journal* 7.4 (1965), pp. 308–313.

[79] Changyong Oh et al. "Batch Bayesian Optimization on Permutations using the Acquisition Weighted Kernel". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 6843–6858.

[80] Changyong Oh et al. "Combo: Combinatorial bayesian optimization using graph representations". In: *ICML Workshop on Learning and Reasoning with Graph-Structured Data*. 2019.

[81] André F Oliveira, Juarez LF Da Silva, and Marcos G Quiles. "Molecular property prediction and molecular design using a supervised grammar variational autoencoder". In: *Journal of Chemical Information and Modeling* 62.4 (2022), pp. 817–828.

[82] Rafael Oliveira, Lionel Ott, and Fabio Ramos. "Bayesian optimisation under uncertain inputs". In: *The 22nd international conference on artificial intelligence and statistics*. PMLR. 2019, pp. 1177–1184.

[83] Sadman Sadeed Omee et al. "Scalable deeper graph neural networks for high-performance materials property prediction". In: *Patterns* (2022).

[84] Andrei Paleyes et al. "Emulation of physical processes with Emukit". In: *Second Workshop on Machine Learning and the Physical Sciences, NeurIPS*. 2019.

[85] Abhirup Patra et al. "A multi-fidelity information-fusion approach to machine learn and predict polymer bandgap". In: *Computational Materials Science* 172 (2020), p. 109286.

[86] Benjamin Peherstorfer, Karen Willcox, and Max Gunzburger. "Survey of multifidelity methods in uncertainty propagation, inference, and optimization". In: *Siam Review* 60.3 (2018), pp. 550–591.

[87] Matthias Poloczek, Jialei Wang, and Peter Frazier. "Multi-information source optimization". In: *Advances in Neural Information Processing Systems*. 2017, pp. 4288–4298.

[88] Matthias Poloczek, Jialei Wang, and Peter I Frazier. "Warm starting Bayesian optimization". In: *2016 Winter Simulation Conference (WSC)*. IEEE. 2016, pp. 770–781.

[89] Raghunathan Ramakrishnan et al. "Quantum chemistry structures and properties of 134 kilo molecules". In: *Scientific data* 1.1 (2014), pp. 1–7.

[90] Patrick Reiser et al. "Graph neural networks for materials science and chemistry". In: *Communications Materials* 3.1 (2022), p. 93.

[91] Matthias Rupp et al. "Fast and accurate modeling of molecular atomization energies with machine learning". In: *Physical review letters* 108.5 (2012), p. 058301.

[92] Omer Sagi and Lior Rokach. "Ensemble learning: A survey". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8.4 (2018), e1249.

[93] Marwin HS Segler et al. "Generating focused molecule libraries for drug discovery with recurrent neural networks". In: *ACS central science* 4.1 (2018), pp. 120–131.

[94] Amar Shah and Zoubin Ghahramani. "Parallel predictive entropy search for batch global optimization of expensive objective functions". In: *Advances in neural information processing systems* 28 (2015).

[95] Benjamin J Shields et al. "Bayesian reaction optimization as a tool for chemical synthesis". In: *Nature* 590.7844 (2021), pp. 89–96.

[96] Saša Singer and John Nelder. "Nelder-mead algorithm". In: *Scholarpedia* 4.7 (2009), p. 2928.

[97] SN Sivanandam et al. *Genetic algorithms*. Springer, 2008.

[98] Jialin Song, Yuxin Chen, and Yisong Yue. "A general framework for multi-fidelity bayesian optimization with gaussian processes". In: *The 22nd International Conference on Artificial Intelligence and Statistics*. 2019, pp. 3158–3167.

[99] Wojciech Stark et al. "Benchmarking machine-learned interatomic potential methods for reactive molecular dynamics at metal surfaces". In: *Bulletin of the American Physical Society* (2023).

[100] MA Styblinski and T-S Tang. "Experiments in nonconvex optimization: stochastic approximation with function smoothing and simulated annealing". In: *Neural Networks* 3.4 (1990), pp. 467–483.

[101] Kevin Swersky, Jasper Snoek, and Ryan P Adams. "Multi-task bayesian optimization". In: *Advances in neural information processing systems*. 2013, pp. 2004–2012.

[102] Kevin Swersky et al. "Amortized bayesian optimization over discrete spaces". In: *Conference on Uncertainty in Artificial Intelligence*. PMLR. 2020, pp. 769–778.

[103] William R Thompson. "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples". In: *Biometrika* 25.3-4 (1933), pp. 285–294.

[104] Robert Tibshirani. "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 58.1 (1996), pp. 267–288.

[105] Roberto Todeschini and Viviana Consonni. *Handbook of molecular descriptors*. John Wiley & Sons, 2008.

[106] Steven B Torrisi et al. "Random forest machine learning models for interpretable X-ray absorption near-edge structure spectrum-property relationships". In: *npj Computational Materials* 6.1 (2020), p. 109.

[107] Tsuyoshi Ueno et al. "COMBO: An efficient Bayesian optimization library for materials science". In: *Materials discovery* 4 (2016), pp. 18–21.

[108] Rashmi Korlakai Vinayak and Ran Gilad-Bachrach. "Dart: Dropouts meet multiple additive regression trees". In: *Artificial Intelligence and Statistics*. PMLR. 2015, pp. 489–497.

[109] *Virtual Library of Simulation Experiments:* `https://www.sfu.ca/~ssurjano/rastr.html`. Accessed: 2021-02-25.

[110] Xinming Wang et al. "Identification of crystalline materials with ultra-low thermal conductivity based on machine learning study". In: *The Journal of Physical Chemistry C* 124.16 (2020), pp. 8488–8495.

[111] Zi Wang, Bolei Zhou, and Stefanie Jegelka. "Optimization as estimation with Gaussian processes in bandit settings". In: *Artificial Intelligence and Statistics*. PMLR. 2016, pp. 1022–1031.

[112] Zi Wang et al. "Batched high-dimensional bayesian optimization via structural kernel learning". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 3656–3664.

[113] Sarah Webb et al. "Deep learning for biology". In: *Nature* 554.7693 (2018), pp. 555–557.

[114] David Weininger. "SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules". In: *Journal of chemical information and computer sciences* 28.1 (1988), pp. 31–36.

[115] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. Vol. 2. MIT press Cambridge, MA, 2006.

[116] Jian Wu and Peter Frazier. "The parallel knowledge gradient method for batch bayesian optimization". In: *Advances in Neural Information Processing Systems*. 2016, pp. 3126–3134.

[117] Jian Wu et al. "Practical multi-fidelity bayesian optimization for hyperparameter tuning". In: *Uncertainty in Artificial Intelligence*. PMLR. 2020, pp. 788–798.