Mayer, Verena:

Alternative Infill-Kriterien für Random Forests in Sequential Model-Based Optimization

Masterarbeit, Wintersemester 2017

Fakultät für Mathematik, Informatik und Statistik

Ludwig-Maximilians-Universität München

https://doi.org/10.5282/ubm/epub.38411

Masterarbeit

Statistik

Alternative Infill-Kriterien für Random Forests in Sequential Model-Based Optimization

---

Vorgelegt von: Verena Mayer

Erstprüfer: Prof. Dr. Bernd Bischl

Betreuer:
Prof. Dr. Bernd Bischl
Giuseppe Casalichio
Janek Thomas

Institut für Statistik
Ludwig-Maximilians-Universität München

20. Januar 2017

Abstract

Sequential Model-Based Optimisation (SMBO) ist eine effiziente Möglichkeit für die Optimierung von kostenintensiven Black-Box-Funktionen. Unter diese Art Problemstellung lässt sich unter anderem die Hyperparameter-Optimierung statistischer Modelle einordnen. Bei SMBO kommen Ersatzmodelle und sogenannte Infill-Kriterien zum Einsatz. Für die Anwendung in kategorialen oder gemischten Parameterräumen finden als Ersatzmodelle meistens Random Forests Anwendung. Hier ergibt sich allerdings der Nachteil, dass etablierte Infill-Kriterien in Kombination mit Random Forests nicht zu optimalen Resultaten führen. Im Rahmen dieser Masterarbeit wurde versucht, ein alternatives Infill-Kriterium zu entwickeln, das sich besser für den Einsatz bei Random Forests eignet.
Benchmark-Experimente auf verschiedenen Problemen ergaben, dass das neue Infill-Kriterium im zweidimensionalen Bereich, verglichen mit anderen Infill-Kriterien, zu besseren Ergebnissen führt. Bei höherer Dimension zeigte sich jedoch, dass die Methode gleich gut, oder sogar schlechter als etablierte Infill-Kriterien abschneidet. Neben dem Random Forest als Ersatzmodell wurde das Kriterium auch mit Extra-Trees getestet. Hier ergaben sich vergleichbare Ergebnisse.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

In vielen Bereichen der Industrie gibt es Problemstellungen, die sich als Optimierung einer Black-Box-Funktion beschreiben lassen. Dabei ist alleinig die Eingabe-Ausgabe Beziehung bekannt, aber nicht der innere Arbeitsablauf des Prozesses. Es lassen sich daher zwar an unterschiedlichen Stellen der Funktion Auswertungen durchführen, eine Annahme über die analytische Form kann jedoch nicht getroffen werden. Aus diesem Grund ist meistens die Bildung von Ableitungen nicht möglich. Ist zudem die Auswertung der Funktion kostenintensiv, besteht bei der Optimierung das Problem, dass man die Funktion nicht beliebig oft evaluieren kann, um das Optimum zu finden. Man kann sich das Szenario wie bei einem Spielautomaten vorstellen, der nach gezahltem Einsatz je nach Tastenkombination einen unterschiedlichen Gewinn ausgibt. Durch eine Tastenkombination (Eingabe) erhält man zwar die Information über den dazugehörigen Gewinn (Ausgabe), man kennt jedoch nicht die darunter liegende Funktion und muss somit durch Ausprobieren den größtmöglichen Gewinn ermitteln. Bei geringem Vermögen hat man allerdings das Problem, dass man nur über eine sehr begrenzte Anzahl an Versuchen verfügt, um den optimalen Gewinn zu finden. Diese Art von Problemstellung findet man in vielen technischen Bereichen wie der Automobil-, Halbleiter- oder auch Luft- und Raumfahrtsindustrie, aber auch bei der Hyperparameter-Optimierung mathematischer Modelle, die unter anderem zur Beschreibung von chemischen, physikalischen oder biologischen Systemen dienen können. Unter Kostenintensität versteht man normalerweise die Rechenzeit, die zur Auswertung der Funktion investiert werden muss. Manchmal versteht man darunter aber auch den praktischen Aufwand oder das notwendige Geld, das für eine Funktionsauswertung aufgebracht werden muss.
Eine effiziente Methode für die Optimierung solcher Art von Problemen ist Sequential-Model-Based-Optimization (SMBO). Hinter dieser Methode steht die Idee, dass mit Hilfe eines nicht so kostenintensiven Ersatzmodells versucht werden soll, das Optimum der Black-Box-Funktion zu schätzen. Um zu erreichen, dass möglichst wenige Auswertungen der wahren Funktion notwendig sind, werden die Auswertungen schrittweise durchgeführt. In jedem Schritt wird dabei das Ersatzmodell an den bereits ausgewerteten Funktionsstellen angepasst. Dadurch kann in jedem Schritt entschieden werden, wo eine neue Auswertung der Funktion am sinnvollsten ist, um das Optimum finden zu können. Für die Entscheidung, welche Funktionsstelle der Black-Box-Funktion für eine Auswertung am geeignetsten erscheint, wird normalerweise ein sogenanntes Infill-Kriterium verwendet. Dieses Kriterium berücksichtigt meistens die Informationen des Ersatzmodells, wie zum Beispiel den geschätzten Mittelwert der Black-Box-Funktion, um eine neue Auswertungsstelle vorzuschlagen. Ein mögliches

Ersatzmodell für die Anpassung an die bereits ausgewerteten Funktionsstellen der Black-Box-Funktion ist der Random Forest. Das Problem bei Random Forests als Ersatzmodell ist aber, dass die gängigen Infill-Kriterien für dieses Modell nicht optimal geeignet sind. Im Rahmen dieser Masterarbeit wurde ein alternatives Infill-Kriterium entwickelt, das versucht, die Eigenheiten des Random Forests zu berücksichtigen. Es wurde sowohl die Theorie des neuen Kriteriums ausgearbeitet, als auch Benchmark-Experimente mit dem neuen und den etablierten Infill-Kriterien durchgeführt.[1]
Im ersten Teil dieser Arbeit werden theoretische Grundlagen zu Sequential Model-Based Optimization vermittelt (Kapitel 2). Dabei wird auch die Theorie der beiden Ersatzmodelle Random Forest und Kriging näher behandelt. Im anschließenden Teil wird das neue Infill-Kriterium vorgestellt und dabei auch auf das Hauptproblem von bisherigen Infill-Kriterien für Random Forests eingegangen (Kapitel 3). Im darauf folgenden Teil werden die durchgeführten Benchmark-Experimente näher beschrieben (Kapitel 4). Zum Schluss werden die gewonnenen Erkenntnisse aus dieser Masterarbeit zusammengefasst und ein Ausblick gegeben (Kapitel 5).

2 Sequential Model-Based Optimization

In diesem Kapitel werden theoretische Grundlagen zu Sequential Model-Based Optimization (SMBO) behandelt. Nachdem zunächst eine Beschreibung über die allgemeine Vorgehensweise von SMBO erfolgt (Kapitel 2.1), wird auf die Theorie der Surrogat-Modelle Gauss-Prozess-Regression, Random Forests und Extra-Trees eingegangen (Kapitel 2.2). Anschließend stehen wichtige Infill-Kriterien und deren Optimierung im Fokus (Kapitel 2.3 und Kapitel 2.4), bevor im letzten Teil der theoretischen Ausführungen wichtige Aspekte von SMBO bei der Hyperparameter-Optimierung erläutert werden (Kapitel 2.5).

2.1 Vorgehensweise von SMBO

Für die Optimierung kostenintensiver Black-Box-Funktionen ist Sequential Model-Based Optimization (SMBO) eine häufig eingesetzte Methode. Bei diesem Problem sollen bei einer unbekannten Funktion f die Eingabewerte x^* gefunden werden, mit denen man das globale Maximum (oder Minimum) von f erhält. Mathematisch lässt sich das durch

$$x^* = \arg\max_{x \in \mathcal{X}} f(x) \tag{1}$$

beschreiben. $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_d$ bezeichnet den d-dimensionalen Eingaberaum. Dieser kann sowohl Dimensionen mit stetig skaliertem Wertebereich, als auch Dimensionen mit kategorial skaliertem Bereich enthalten. Die Dimensionen werden im stetigen Fall durch eine obere und untere Schranke beschränkt und im kategorialen Fall durch eine endliche Anzahl an Kategorien. Durch eine Auswertung der Funktion f mit Eingabewerten einer Stelle des Eingaberaums, lässt sich ein Ausgabewert $y = f(x)$ erzeugen.[2]
Black-Box-Funktionen haben die Eigenschaft, dass es zwar möglich ist, durch Eingabewerte den entsprechenden Ausgabewert der Funktion zu erhalten, es aber kein Wissen über die analytische Form der Funktion gibt. Dadurch lassen sich meist keine Ableitungen bilden. Da es sich zusätzlich um kostenintensive Funktionen handelt, steht außerdem nur eine begrenzte Anzahl an möglichen Auswertungen von f zur Verfügung. Bei der Optimierung möchte man aus diesem Grund schon nach wenigen Auswertungen möglichst nahe an das globale Optimum gelangen.Um das zu erreichen, ist bei SMBO die Idee, die zur Verfügung stehende Anzahl möglicher Auswertungen der Black-Box-Funktion nicht auf einmal, sondern Schritt für Schritt durchzuführen. In jedem Schritt wird dabei für den Vorschlag einer neuen Auswertung im

Eingaberaum $\mathcal{X}$, das Wissen bereits ausgewerteter Stellen miteinbezogen. Durch dieses iterative Vorgehen muss der Eingaberaum nicht vollständig abgetastet werden, sondern es werden nur Bereiche genauer untersucht, in dem das globale Optimum der wahren Funktion vermutet wird.
Der genaue Ablauf sieht so aus, dass vor dem Beginn des iterativen Prozesses die zu optimierende Black-Box-Funktion, häufig auch als Target-Funktion bezeichnet, an ein paar Stellen des Eingaberaums $\mathcal{X}$ ausgewertet wird. Die ausgewählten Funktionsstellen ergeben das sogenannte initiale Design, bzw. die initialen Designpunkte. Es gibt verschiedene Ansätze, wie man die Punkte im Eingaberaum für das initiale Design wählt. Beispiele sind eine zufällige Auswahl, space-filling Latin Hypercube Designs oder auch der Ansatz, die Punkte über Meta-Learning zu setzen [3]. Das initiale Design wird mit der Black-Box-Funktion ausgewertet, wodurch man Wissen über die Eingabe-Ausgabe Beziehung erhält. Nachdem die Ausgabewerte des initialen Designs bekannt sind, kann SMBO mit den Iterationen starten. Eine Iteration sieht dabei so aus, dass auf den bereits ausgewerteten Punkten ein Ersatzmodell, das auch als Surrogat-Modell bezeichnet wird, geschätzt wird. Mit Hilfe dieses Modells und einem sogenannten Infill-Kriterium folgt die Entscheidung, welche Stelle im Eingaberaum als nächstes für eine Auswertung herangezogen wird, um näher an das globale Optimum zu gelangen. Die ausgewählte Stelle im Eingaberaum stellt einen neuen Designpunkt dar. Nach dessen Auswertung dient dieser bei der nächsten Iteration zusammen mit den anderen Designpunkten für eine erneute Anpassung des Surrogat-Modells, womit eine neue Iteration eingeleitet wird. Meistens wird SMBO nach einer bestimmten Iterationsanzahl gestoppt und der bis dahin beste Ausgabewert als Optimum angesehen.[1]
Das folgende Beispiel soll das Vorgehen von SMBO veranschaulichen. Es wird dafür eine eindimensionale Funktion maximiert. Die Beispielfunktion stellt die Black-Box-Funktion dar, deren Form wir in der Realität nicht kennen. Als initiales Design wählen wir zwei zufällige Punkte aus. Abbildung 1 zeigt die Target-Funktion als gestrichelte Linie und bereits ausgewertete Funktionsstellen als schwarze Punkte. Sind die wahren Werte für das initiale Design ermittelt, kann mit dem iterativen Prozess begonnen werden.

Eine Iteration besteht aus folgenden Schritten [3]:

(1) Auf den durch die Target-Funktion bereits ausgewerteten Punkten wird ein Surrogat-Modell angepasst. Durch dieses Modell, das im Englischen auch unter dem Namen response-surface-model bekannt ist, soll das Verhalten der Target-Funktion geschätzt werden. Bewährt haben sich Gauss-Prozesse oder Random

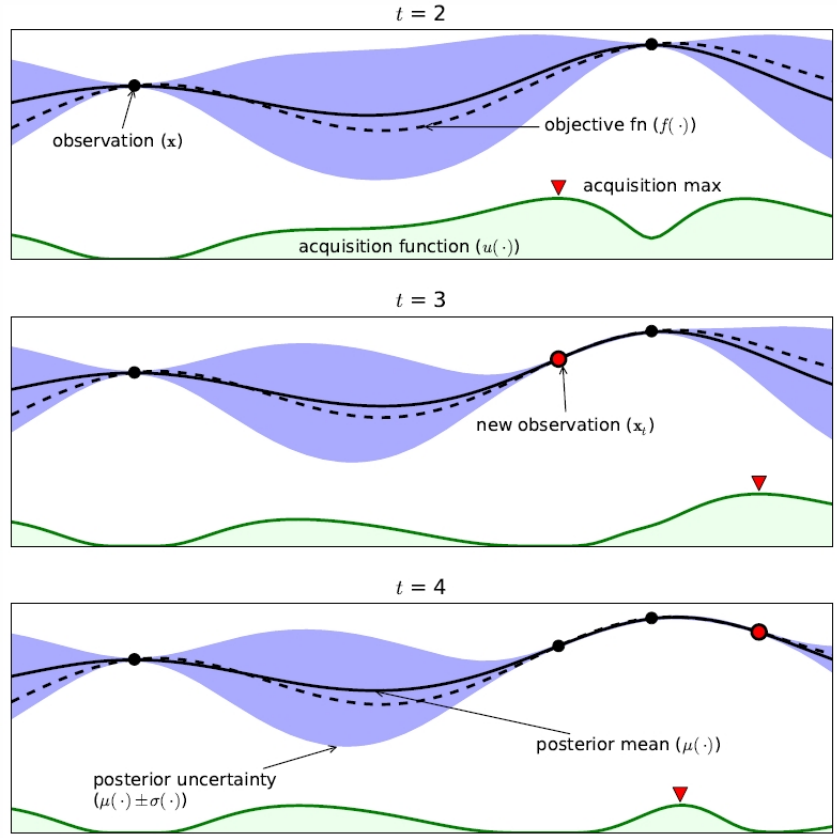

Abbildung 1 Beispiel von SMBO auf einer eindimensionalen Beispielfunktion (gestrichelte Linie). Als Surrogat-Modell wurde eine Gauss-Prozess-Regression (GPR) verwendet. Die durchgezogene schwarze Linie zeigt den geschätzten Mittelwert der GPR und die blaue Schattierung dessen Unsicherheit. Grüne Linie stellt das Infill-Kriterium dar, die Maxima sind dabei durch rote Dreiecke symbolisiert. Quelle: [1]

Forests. In Kapitel 2.2 wird näher auf die Theorie dieser beiden Surrogat-Modelle eingegangen. In Abbildung 1 wurde eine Gauss-Prozesss-Regression als Surrogat-Modell verwendet. Der Schätzer ist als durchgezogene schwarze Linie eingezeichnet. Die blaue Schattierung zeigt dabei die Unsicherheit der Schätzung.

(2) Mit Hilfe des Surrogat-Modells möchte man nun entscheiden, an welcher Stelle eine erneute Auswertung der Target-Funktion am sinnvollsten ist. Dabei möchte man die Stelle auswerten, bei der die Chance für eine Verbesserung der Vorhersage des Optimums, am größten ist. Hier besteht ein Trade-Off zwischen der Erforschung unbekannter Bereiche und der Ausnutzung bereits bekannter Bereiche. Für diese Aufgabe gibt es verschiedene Infill-Kriterien. Im Englischen werden sie auch acquisition-functions genannt. Häufig berücksichtigen diese sowohl den geschätzten Mittelwert, als auch die Varianz des Surrogat-Modells [1].

In Kapitel 2.3 gehen wir auf ein paar mögliche Infill-Kriterien ein.
In der Abbildung ist die Funktion des Infill-Kriteriums als grüne Linie dargestellt. Je höher der Wert, desto eher wird eine Verbesserung vermutet. Man kann erkennen, dass das Infill-Kriterium in Bereichen mit größerer Unsicherheit höhere Werte annimmt, wohingegen bereits ausgewertete Bereiche niedrigere Werte erhalten. Das ist sinnvoll, da man in unsicheren Bereichen eine stärkere Verbesserung zu den bereits ausgewerteten Punkten vermuten kann. Eine neue Auswertung soll möglichst dort vorgeschlagen werden, wo das Infill-Kriterium das Maximum annimmt. In der Praxis finden verschiedene Optimierungstechniken Anwendung (Näheres in Kapitel 2.4).

(3) Anschließend wird die Target-Funktion an der durch das Infill-Kriterium vorgeschlagenen Stelle ausgewertet.

Nun wiederholt man die Schritte (1) bis (3) solange, bis zum Beispiel eine bestimmte Iterationsanzahl erreicht ist, oder das Budget aufgebraucht ist [1]. In Abbildung 1 sieht man bei den folgenden beiden Iterationen, dass man sich nach und nach an das Optimum (in diesem Beispiel das Maximum) herantastet. Am Ende der Iterationen wird zum Beispiel der beste aller bereits ausgewerteten Punkte als Optimum angesehen. Man kann gut erkennen, dass aufgrund der Vorschläge des Infill-Kriteriums nur relativ wenige Auswertungen der Target-Funktion notwendig sind, um in die unmittelbare Nähe des globalen Optimums zu gelangen.
Bei SMBO gibt es verschiedene Möglichkeiten für die Wahl des initialen Designs, der Surrogat-Modelle und des Infill-Kriteriums. Zum initialen Design sei anzumerken, dass nicht nur die Methode, wie die Punkte in den Raum gelegt werden, sondern auch die Größe des initialen Designs eine Rolle spielt. Zu groß sollte man es nicht wählen, da ansonsten hohe Kosten durch die Erstellung des initialen Designs entstehen und dann evtl. nicht mehr so viel Budget für den iterativen Prozess vorhanden ist. Im Gegensatz dazu würde ein sehr kleines Design eine gute Schätzung der wahren Funktion durch das Ersatzmodell verhindern [4]. Deshalb sind viele Iterationen notwendig, bis gute Vorschläge möglich sind, die in Richtung des globalen Optimums führen. Je nach verwendetem Kriterium für den Vorschlag neuer Punkte kann es sogar dazu kommen, dass das globale Optimum nicht gefunden werden kann.
In den nun folgenden Kapiteln werden Surrogat-Modelle und Infill-Kriterien näher behandelt.

2.2 Surrogat-Modelle

Ein Surrogat-Modell sollte nicht zu kostenintensiv in der Schätzung sein, da bei SMBO viele Modellanpassungen und Prognosen der Target-Funktion durchgeführt werden müssen. Für den Einsatz mit Infill-Kriterien ist des Weiteren das Vorhandensein eines guten Varianzschätzers von Vorteil (siehe Kapitel 2.3).
In diesem Kapitel werden wir die beiden am häufigsten verwendeten Surrogat-Modelle, Gauss-Prozess-Regression und Random Forests, näher betrachten. Zusätzlich wird kurz auf die Theorie von Extra-Trees eingegangen.
Bei Surrogat- Modellen bewegen wir uns im Feld der überwachten Lernalgorithmen (Supervised-Learning). Mit der Hilfe eines Trainingsdatensatzes bei dem Eingabe (x) und Ausgabe (y) gegeben sind, sollen so gut wie möglich die Gesetzmäßigkeiten zwischen den abhängigen und unabhängigen Variablen gelernt werden, um später so gut wie möglich die Ausgabe bei unbekannten Eingabe-Werten prognostizieren zu können. Die Gesetzmäßigkeit wird also unter der Überwachung der Ausgabe gelernt. Man unterscheidet zwischen Regressions- und Klassifikationsproblem, je nachdem, ob die Ausgabe stetig oder kategorial skaliert ist [5]. Da man es bei SMBO meistens mit stetigen Ausgabe-Werten zu tun hat, werden wir sowohl bei Gauss-Prozessen, als auch bei Random-Forests vor allem den Regressionsfall behandeln.

2.2.1 Gauss-Prozess-Regression

Gauss-Prozess-Regression (GPR) kann man auf verschiedene Art und Weise interpretieren. Bei den folgenden Erklärungen werden Gauss-Prozesse als eine Verteilung über Funktionen angesehen, bei der die Inferenz direkt im Funktionen-Raum vonstatten geht. Die Anfänge für die Verwendung von Gauss-Prozessen als Vorhersagemodelle lassen sich auf Arbeiten von [6] und [7] aus den 1990er Jahren zurückverfolgen. Weitere Arbeiten wurden sogar bis zum Jahr 1880 datiert. GPR entwickelte sich außerdem in verschiedenen wissenschaftlichen Teilgebieten. Durch die Geostatistik ist die GPR auch unter dem Namen Kriging bekannt geworden. Der Name stammt vom Bergbauingenieur D. G. Krige, aus dessen Arbeiten der Mathematiker Georges Matheron die theoretische Basis der Methode entwickelte.[8]
GPR und Kriging werden oft als Synonyme verwendet, wobei in den folgenden Beschreibungen überwiegend von GPR die Rede ist, da hier nicht die Herangehensweise aus dem geostatistischen Bereich verwendet wird. Wo es notwendig erscheint, wird außerdem die Charakteristik von GPR bei SMBO behandelt.
Wenden wir uns also nun der GPR als Verwendung eines Vorhersagemodells zu. Wir stellen uns als Beispiel eine eindimensionale Black-Box-Funktion vor, bei der wir an bestimmten Stellen Vorhersagen über die Funktionswerte treffen wollen. Falls noch

keine Trainingsdaten vorliegen, die Black-Box-Funktion also noch an keiner Stelle ausgewertet wurde, existiert kein konkretes Wissen über die wahren Funktionswerte. Die Idee ist nun, dass man zwar kein explizites Wissen der Funktionswerte besitzt, man aber dafür bereits Annahmen über die Werte treffen kann – das sogenannte a priori Wissen über die Funktionswerte. In unserem Fall überlegen wir uns, dass jeder Punkt die Realisation einer Zufallsvariablen $f(x)$ ist, die einer Normalverteilung mit Mittelwert μ und Varianz σ^2 folgt. Wir nehmen also an, dass die Punkte einen typischen Wert μ besitzen und innerhalb eines Intervalls (z. B. $[\mu - 2\sigma, \mu + 2\sigma]$) variieren können. So lässt sich unsere Unsicherheit über die Werte darstellen, die wir ohne Auswertung der Punkte haben. Beachtet man nun zusätzlich die Distanz der Punkte untereinander, so lässt sich vermuten, dass Punkte, die nahe beieinander liegen, ähnliche Funktionswerte aufweisen, wohingegen weit entfernte Punkte nicht mehr so viel miteinander zu tun haben. Wir nehmen also an, dass $f(x_i)$ und $f(x_j)$ umso stärker korrelieren, je kleiner die Distanz $|x_i - x_j|$ ist.[9]
Fassen wir unsere Annahmen zusammen, so lassen sich die Zufallsvariablen $f(x)$ durch einen Gauss-Prozess (GP)

$$f(x) \sim GP(\mu(x), k(x, x')) \tag{2}$$

mit Mittelwertsfunktion $\mu(x) = E[f(x)]$ und einer Kovarianzfunktion $k(x, x') = E[(f(x) - \mu(x))(f(x') - \mu(x'))]$ definieren. Auf die Wahl der Kovarianzfunktion wird später noch genauer eingegangen. Alle möglichen Eingabewerte x, lassen sich als Indexmenge $\mathcal{X}$ bezeichnen.[8]
Man kann sich den GP als eine Ansammlung normalverteilter Zufallsvariablen vorstellen, bei der für jeden Eingabewert x der Ausgabewert y durch eine Zufallsvariable repräsentiert wird. Bei einem GP für eine endliche Anzahl m an x-Werten $x_1, ..., x_m$, folgen die zugehörigen m Zufallsvariablen $f(x_1), ..., f(x_m)$ einer multivariaten Normalverteilung:

$$\begin{bmatrix} f(x_1) \\ \vdots \\ f(x_m) \end{bmatrix} \sim N\left(\begin{bmatrix} \mu(x_1) \\ \vdots \\ \mu(x_m) \end{bmatrix}, \begin{bmatrix} k(x_1, x_1) & \dots & k(x_1, x_t) \\ \vdots & \ddots & \vdots \\ k(x_t, x_1) & \dots & k(x_t, x_t) \end{bmatrix} \right) \tag{3}$$

Dabei kann der Vektor $(f(x_1), \cdots, f(x_m))^T$ als eine Funktion betrachtet werden, da jedem x-Wert genau ein y-Wert zugeordnet wird. Man kann also sagen, dass bei GP eine Beobachtung nicht aus einem einzigen Wert, sondern aus einer ganzen Funktion besteht. Deshalb spricht man auch von der Verteilung über Funktionen.

Jeder x-Wert erhöht die Dimension der multivariaten Normalverteilung. Im Falle unendlicher Indexmengen erhält man daraus eine unendlich-dimensionale Normalverteilung und für jede endliche Menge $x_1, ..., x_m \in \mathcal{X}$ ist $(f(x_1), ..., f(x_m))^T$ m-dimensional normalverteilt. Dies wird als Marginalitätseigenschaft von GP bezeichnet. Aufgrund der Marginalitätseigenschaft sind GP sehr gut zur Modellierung einer endlichen Anzahl an Variablen geeignet.[8]

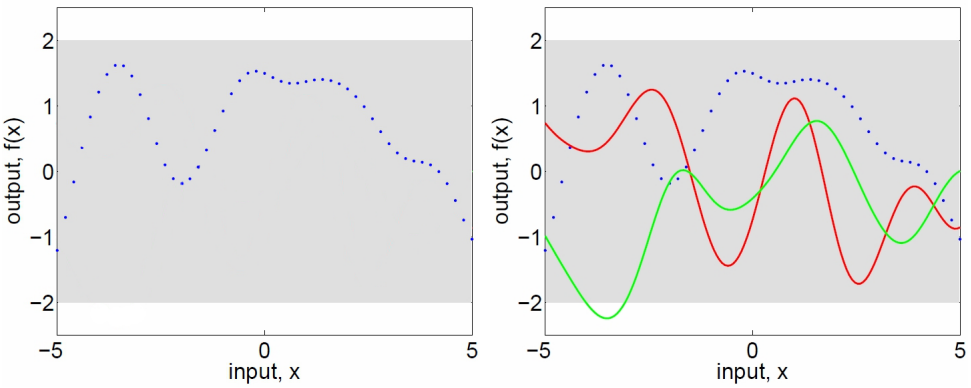

Abbildung 2 Ziehungen mit 50 Eingabewerten aus der a priori Verteilung eines GP. Links eine Ziehung. Rechts zwei zusätzliche Ziehungen, bei denen die Werte verbunden wurden. Die graue Schattierung repräsentiert die punktweisen Mittelwert plus/minus der zweimaligen Standardabweichung für jeden Eingabewert. [8]

Wollen wir eine zufällige GP Realisation von $f(x)$ erhalten, so ziehen wir auf einem Gitter von x-Werten die entsprechenden y-Werte aus der a priori Normalverteilung. In Abbildung 2, bei der linken Grafik, ist das Ergebnis bei einmaligem Ziehen aus einer a priori Verteilung mit Gittergröße $n = 50$ dargestellt. Dem Eingabewert wird ein gezogener Ausgabewert zugeordnet. Man erhält dadurch für jeden der 50 x-Werte eine Realisation der zugehörigen Zufallsvariablen. Normalerweise würde man für die endlich große Ziehung keine durchgezogene Linie erhalten, sondern wie dargestellt, nur die einzelnen Punkte der Realisationen. Es ist jedoch üblich, die Werte eines endlichen Vektors zu verbinden. In der rechten Grafik sind zwei weitere Ziehungen aus der multivariaten Normalverteilung veranschaulicht, wobei diesmal die Werte verbunden wurden.[8]
Ein GP wird mit der Wahl von $\mu(x)$ und $k(x, x')$ spezifiziert. In der eben behandelten Abbildung wurde eine Mittelwertsfunktion von $\mu(x) = 0$ gewählt. Alternative priors für den Mittelwert bei SMBO werden unter anderem in [10] und [11] behandelt [1]. Theoretisch könnte man für SMBO auch eine Mittelwertsfunktion mit Trend verwenden. Jedoch besteht in der Realität das Problem, dass man den wahren Verlauf des Mittelwerts nicht kennt. Die Gefahr ist deshalb, dass die Einführung eines Trends mehr Schaden als Nutzen bringen kann, da man bei einer Fehlspezifikation sozusagen

gegen die wahren Werte arbeitet. Vor allem in Bereichen, die weiter von den Trainingspunkten entfernt sind, hat die Annahme des Mittelwertes einen großen Einfluss auf spätere Vorhersagen. Gerade im höher dimensionalen Fall kann das zu starken Abweichungen der Prognose führen. Aus diesem Grund sei die Einführung einer Trendfunktion wohl überlegt. Durch das Unwissen bezüglich des Trends wird meistens eine konstante (häufig gleich Null), Mittelwertsfunktion für SMBO verwendet.
Wenden wir uns nun der Kovarianzfunktion $k(x, x')$ zu. Die Wahl der Kovarianz ist entscheidend, da sie die Glätte der gezogenen Funktionen bestimmt. In Abbildung 2 wurde eine sogenannte Gauß-Korrelationsfunktion

$$k(x, x') = \exp(-\frac{1}{2\theta^2} \quad ||x - x'||^2) \tag{4}$$

mit $\theta = 1$ verwendet [1]. Der Parameter θ stellt einen Gewichtsterm für die Distanz der Variablen dar. Er gibt an, wie schnell die Korrelation abnimmt, wenn man sich von einem Punkt auf der Koordinaten-Ebene, bzw. Achse, wegbewegt. Je kleiner Theta, desto schneller nimmt die Korrelation ab, da die Distanz ein höheres Gewicht bekommt.

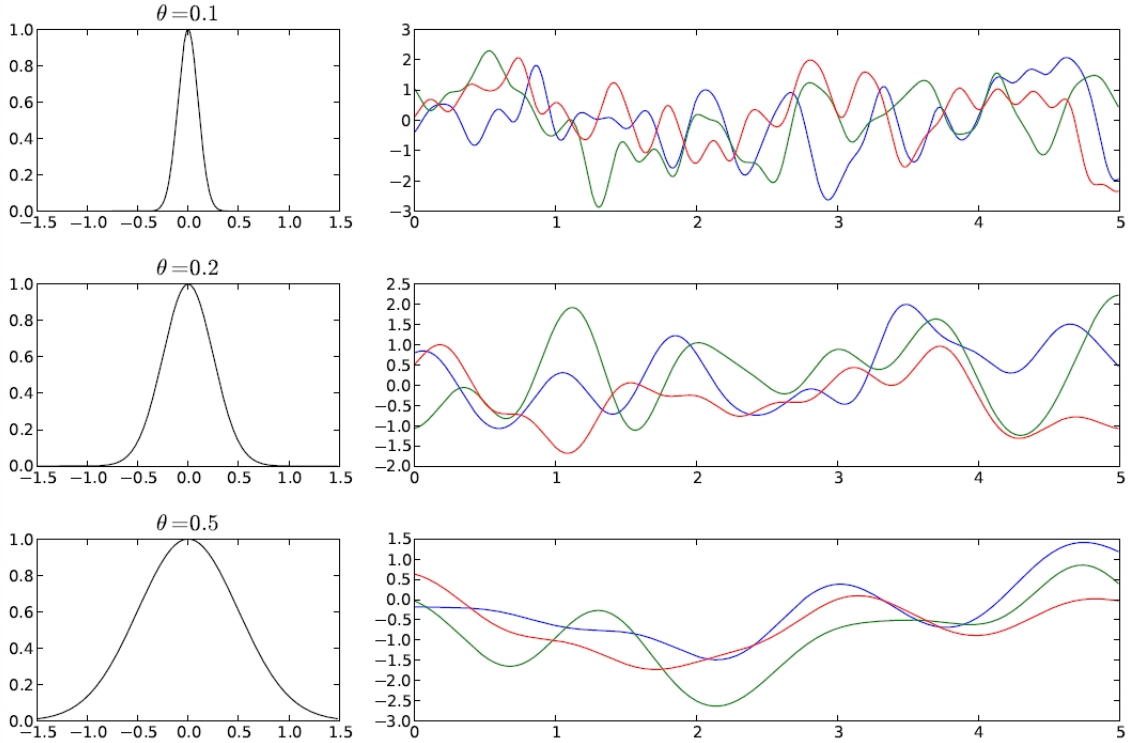

Abbildung 3 Auswirkung des Parameters θ der Gauss-Korrelationsfunktion. Links sind die Funktionen $k(0, x)$ zu sehen und rechts Ziehungen aus der priori Verteilung eines GP mit entsprechendem θ. Quelle: [1]

Schaut man sich Ziehungen mit verschiedenen Theta-Werten an (Abbildung 3), so kann man erkennen, dass die Funktionen mit $\theta = 0.1$ häufiger als bei $\theta = 0.5$ hin und her schwanken, da der vorherige Wert nun keine so große Rolle mehr spielt, wenn man sich nur ein kleines Stück an der Koordinaten-Achse entlang bewegt.[1]
Weitere häufig verwendete Korrelationsfunktionen sind Matérn-Korrelationsfunktionen. Diese sind wie die Gauss-Korrelationsfunktion stationär. Stationarität bedeutet, dass die Korrelationsfunktionen invariant gegenüber Verschiebungen sind, d. h. für jeden x-Wert gelten die gleichen Eigenschaften unabhängig von deren Index [1]. In Abbildung 4 sind Funktionen mit verschiedene Matérn-Korrelationsfunktionen abgebildet. Links ist das Profil und rechts Ziehungen aus der priori Verteilung eines GP zu sehen. Bei der Matérn-1-Korrelationsfunktion (gelb) wird die schnellste Abnahme der Korrelation angenommen. Daraus folgen Funktionen mit einer geringen Glätte. Bei der Gauß-Korrelationsfunktion, hier angegeben durch SQ-EXP und die Farbe grün, wird dagegen eine langsamere Abnahme der Korrelation erwartet.

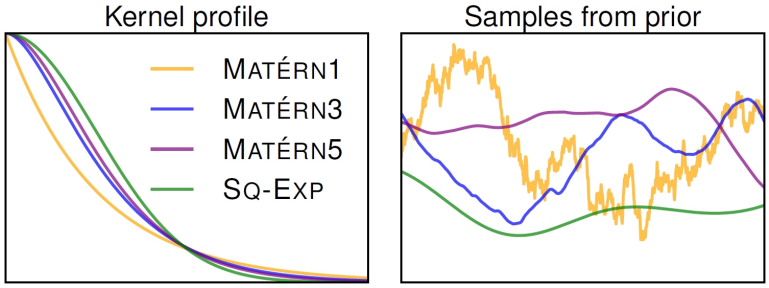

Abbildung 4 Darstellung von Matérn-Korrelationsfunktionen. Links sind die Profile dargestellt, wobei die horizontale Achse die Distanz angibt. Rechts werden Ziehungen aus der GP priori Verteilung bei Verwendung der entsprechenden Korrelationsfunktion gezeigt. Quelle: [2]

[1]Zusammenfassend können wir jetzt sagen, dass die Funktionswerte unserer Black-Box Funktion als a priori Annahme einem GP mit spezifizierter Mittelwerts- und Kovarianzfunktion folgen. Aber wie kommen wir nun zu einem Schätzer unserer Black-Box Funktion? Dafür wird die Black-Box-Funktion an einigen Stellen $\mathbf{x}_{1:\mathrm{m}} = (\mathrm{x}_1, ..., \mathrm{x}_\mathrm{m})$ ausgewertet. Die so erhaltenen Funktionswerte $\mathbf{f}_{1:m} = (f(x_1), ..., f(x_m))$ stellen die Trainingsdaten $\mathcal{D}_{1:m} = \{\mathbf{x}_{1:m}, \mathbf{f}_{1:m}\}$ dar. Mit dem Wissen über wahre Funktionswerte wissen wir nun, dass die Zielfunktion durch die ausgewerteten Punkte führen muss. Aus diesem Grund sind jetzt nur noch a priori Funktionen interessant, die durch diese Punkte führen. In Abbildung 5 ist das Szenario bei Auswertung von fünf Punkten dargestellt. Die Punkte sind wie Nadelöhre, durch die die Funktionen geführt werden. Als Beispiel sind drei Funktionen als Ziehungen aus der posteriori Verteilung von

[1]Das restliche Kapitel bezieht sich auf [1]

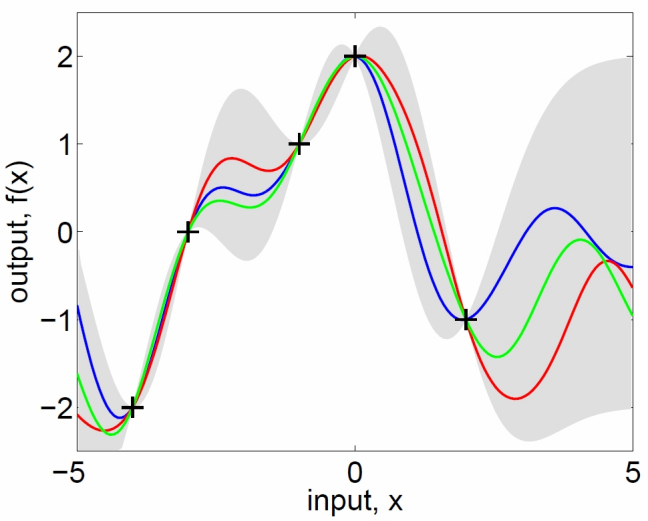

Abbildung 5 Drei zufällige Ziehungen aus der prädiktiven posteriori Verteilung bei fünf Beobachtungen in den Trainingsdaten. Die graue Schattierung repräsentiert den Mittelwert plus/minus der zweimaligen Standardabweichung für jeden Eingabewert. Quelle: [8]

$f(x)$ dargestellt. Wie schon bei der priori Verteilung wurden hier für jede Ziehung die selben 50 x-Werte ausgewählt. Die prädiktive Posteriori Verteilung lässt sich wie bei der Bayesischen linearen Regression über das Bayes-Theorem herleiten, bei der mit Hilfe der Berechnung der posteriori Verteilung die prädiktive Posteriori Verteilung für einen Testpunkt berechnet werden kann. Bei der GPR gibt es jedoch auch noch einen einfacheren Weg. Und zwar folgt bei GP, wie bereits erwähnt, jede endliche Teilmenge von Zufallsvariablen einer multivariaten Normalverteilung. Dies gilt sowohl für die Trainings- als auch für die Testdaten, also den Punkten, an denen wir Vorhersagen über die wahren Werte treffen wollen.
Nehmen wir an, wir wollen an der Stelle x_{m+1} eine Vorhersage über den Funktionswert $f_{m+1} = f(x_{m+1})$ treffen. Dann ergibt sich für die Daten eine gemeinsame Verteilung von

$$\begin{bmatrix} \mathbf{f}_{1:m} \\ f_{m+1} \end{bmatrix} \sim N\Big(0, \begin{bmatrix} \mathbf{K} & \mathbf{k} \\ \mathbf{k}^T & k(x_{m+1}, x_{m+1}) \end{bmatrix}\Big) \tag{5}$$

mit

$$\mathbf{k} = \begin{bmatrix} k(x_{m+1}, x_1) & k(x_{m+1}, x_2) & \dots & k(x_{m+1}, x_m) \end{bmatrix}$$

$$\mathbf{K} = \begin{bmatrix} k(x_1, x_1) & \dots & k(x_1, x_t) \\ \vdots & \ddots & \vdots \\ k(x_t, x_1) & \dots & k(x_t, x_t) \end{bmatrix}$$

Im nächsten Schritt muss die Verteilung auf Funktionen eingeschränkt werden, die durch die beobachteten Werte führen. Dies erreicht man, in dem man die Regeln für das Bedingen von Normalverteilungen anwendet und auf die beobachteten Werte (Trainingsdaten) bedingt. Dadurch ergibt sich die prädiktive Posterioriverteilung

$$P(f_{m+1}|\mathcal{D}_{1:m}, x_{m+1}) = N(\mu_m(x_{m+1}), \sigma_m^2(x_{m+1})) \tag{6}$$

mit

$$\mu_m(x_{m+1}) = \mathbf{k}^T\mathbf{K}^{-1}\mathbf{f}_{1:m}$$

$$\sigma_m^2(x_{m+1}) = k(x_{m+1}, x_{m+1}) - \mathbf{k}^T\mathbf{K}^{-1}\mathbf{k}$$

Durch diese Verteilung lässt sich nun für jeden beliebigen Punkt eine Vorhersage mit einem Intervall für die Unsicherheit treffen. In Abbildung 6 sehen wir die typische

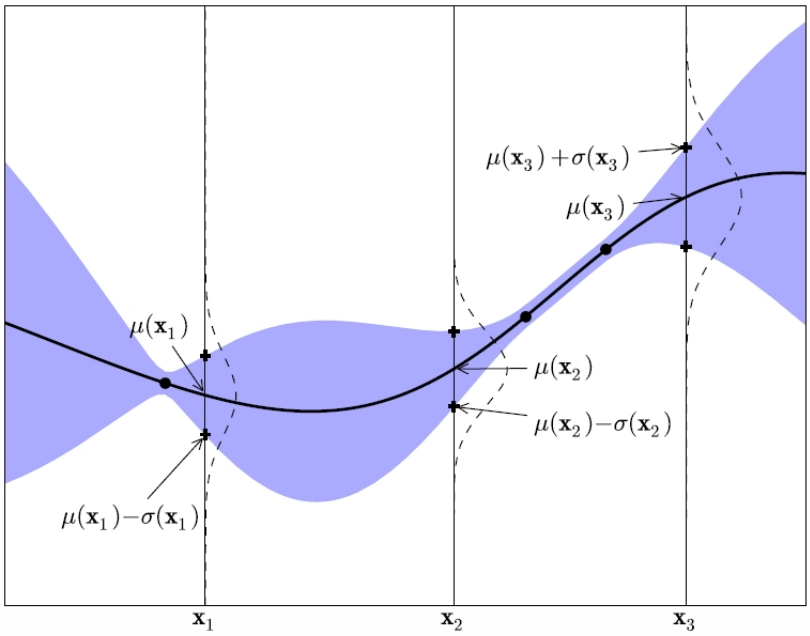

Abbildung 6 GPR mit drei Trainingspunkten. Die schwarze Linie stellt den geschätzten Mittelwert der Target-Funktion dar, die Schattierung die zugehörige Varianz. Für die Testpunkte $x_{1:3}$ ist zusätzlich die Normalverteilung mit Mittelwert und Standardabweichung der Vorhersage dargestellt. Quelle: [1]

Darstellung einer GPR. Die durchgezogene schwarze Linie zeigt den Mittelwert der prädiktiven posteriori Verteilung bei drei Trainingspunkten. Die Unsicherheit wird durch die blaue Schattierung dargestellt. An jeder Stelle $x \in \mathcal{X}$ ergibt sich damit eine normalverteilte Zufallsvariable $f(x)$, hier für x_1, x_2 und x_3 eingezeichnet. Die Varianz an den Trainingsdaten nimmt bei einer Target-Funktion ohne Rauschen den Wert null an. In der Praxis liegt bei der Verwendung von GPR für SMBO jedoch meistens

der Fall mit Rauschen vor, z. B. bei der Hyperparameter-Optimierung. Auch in obiger Abbildung kann man erkennen, dass an den Trainingspunkten noch eine Varianz vorhanden ist. Die exakten Werte von $f(x)$ können also nicht beobachtet werden, sondern nur eine verrauschte Transformation von $f(x)$. Die einfachste Transformation tritt auf, wenn $f(x) + \lambda$ mit unabhängig von anderen Stellen normalverteiltem Fehler $\lambda \sim N(0, \sigma_{noise})$, d. h. additives Rauschen, beobachtet wird. In diesem Fall kann die Verteilung des Rauschens auf die Normalverteilung $N(0, \mathbf{K})$ addiert werden. Zur Matrix $\mathbf{K}$ wird der Term $\sigma^2_{noise}\boldsymbol{I}$ hinzu addiert. Es ergibt sich dann eine prädiktive Verteilung von

$$P(f_{m+1}|D_{1:m}, x_{m+1}) = N(\mu_m(x_{m+1}), \sigma^2_m(x_{m+1}) + \sigma^2_{noise}) \tag{7}$$

mit

$$\mu_t(x_{m+1}) = \mathbf{k}^T[\mathbf{K} + \sigma^2_{noise}\boldsymbol{I}]^{-1}\mathbf{f}_{1:m}$$
$$\sigma^2_m(x_{m+1}) = k(x_{m+1}, x_{m+1}) - \mathbf{k}^T[\mathbf{K} + \sigma^2_{noise}\boldsymbol{I}]^{-1}\mathbf{k}$$

Aus der Geostatistik ist der Effekt des Rauschens auch unter dem Namen Nugget-Effekt bekannt.
GPR ist aufgrund der guten Varianzeigenschaft für SMBO gut geeignet. Die Varianz sollte in Bereichen ohne Trainingspunkte groß sein, wohingegen sie in Bereichen mit Trainingspunkten klein sein sollte. Durch diese Eigenschaft lassen sich später mit Hilfe eines passenden Infill-Kriteriums sehr gut neue Punkte finden, deren Auswertung eine starke Verbesserung in Richtung Optimum versprechen. Im Kapitel 2.3 wird näher darauf eingegangen. Wegen der Verwendung des Bayes-Ansatzes bezeichnet man SMBO manchmal auch als Bayesian-Optimization. Eine weitere Möglichkeit, ein Surrogat-Modell zu fitten, ist der Random Forest, mit dem wir uns im folgenden Kapitel beschäftigen werden.

2.2.2 Random Forests

Bei Random Forests werden Klassifikations- oder Regressionsbäume für die Schätzung verwendet. Aus diesem Grund wollen wir uns zuerst kurz die wichtigsten Grundlagen zur Erstellung eines Baumes ansehen. Wie schon bei der GPR wird nur der Regressionsfall behandelt. Falls nicht anders vermerkt, bezieht sich das Kapitel auf [5].
Zur Erklärung von Bäumen wird im Folgenden ein Beispiel herangezogen. Und zwar sollen bei einer eindimensionalen Funktion an bestimmten Stellen Vorhersagen über den Responsewert y getroffen werden. Wir nehmen an, dass bereits $N = 7$ Punkte ausgewertet wurden und somit als Trainingsdaten zur Verfügung stehen. Als erstes überlegen wir uns, dass der Response y durch einen konstanten Wert c geschätzt werden könnte. Für die Berechnung bietet sich dafür das arithmetische Mittel der Daten an ($\hat{c} = \frac{1}{N}\sum_{i=1}^{N} y_i$). Ohne Unterteilung der Daten wäre das natürlich ein eher schlechter Schätzer, da innerhalb des ganzen Parameterraums der gleiche Responsewert prognostiziert werden würde. Die Idee ist deshalb, dass man die Menge an Punkten so einteilt, dass möglichst homogene Mengen bezüglich der Responsevariablen entstehen. Das heißt, dass sich innerhalb einer Teilmenge Punkte mit ähnlichen Responsewerten befinden sollen. Das arithmetische Mittel kann dann in jeder Teilmenge getrennt berechnet werden und als Schätzer für den jeweiligen Funktionsabschnitt dienen. Der am häufigsten verwendete Algorithmus zur Erstellung von Bäumen (CART) erzeugt dabei schrittweise immer nur zwei Teilmengen. In Abb. 7 ist ein Regressionsbaum mit zugehöriger Funktion abgebildet. Die roten Punkte stellen die Trainingsdaten und die roten Linien die Prognosewerte dar. Betrachten wir zuerst die Ausgangsdaten, die als Oval ganz oben dargestellt sind. Diese Menge wird auch als Wurzel bezeichnet. In ihr sind alle Trainingsdaten enthalten. Im ersten Schritt wird nun ein Splitpunkt in unserer Variablen x gesucht um zwei Teilmengen zu erzeugen. Die Responsewerte innerhalb der Teilmengen sollen dabei so ähnlich wie möglich sein. Um den bestmöglichen Splitpunkt zu finden, addieren wir von beiden Teilmengen die Summe der quadratischen Abweichungen zwischen Responsewerten und arithmetischem Mittel der Teilmenge. Diesen Wert ermitteln wir bei allen möglichen Splitpunkten. Als mögliche Splitpunkte werden natürlich nur die Trainingsdaten berücksichtigt, nicht aber alle unendlich möglichen Werte der x-Variablen. Bezeichnen wir $R_1(j, s)$ und $R_2(j, s)$ als unsere Teilmengen bei Verwendung der Splitvariablen j (in unserem Fall gibt es nur eine Variable) und des

Splitpunkts s. So lässt sich die Suche des Splitpunktes mathematisch durch

$$\min_{j,s}[\min_{c_1}\{\sum_{x_i \in R_1(j,s)} (y_i - c_1)^2\} + \min_{c_2}\{\sum_{x_i \in R_2(j,s)} (y_i - c_2)^2\}]$$

darstellen. Dabei ist c_1 und c_2 jeweils das arithmetische Mittel in Teilmenge 1 und 2. Der beste Splitpunkt ist in der Abbildung als gestrichelte Linie von Wurzel bis x-Achse dargestellt. Die anderen drei gestrichelten Linien denken wir uns fürs erste weg. Es ist nun eine Teilmenge entstanden, die alle Punkte links der gestrichelten Linie enthält und eine Teilmenge die alle Punkte rechts der Linie enthält. Im Baum werden die gebildeten Teilmengen als Kreise links und rechts der Wurzel veranschaulicht.

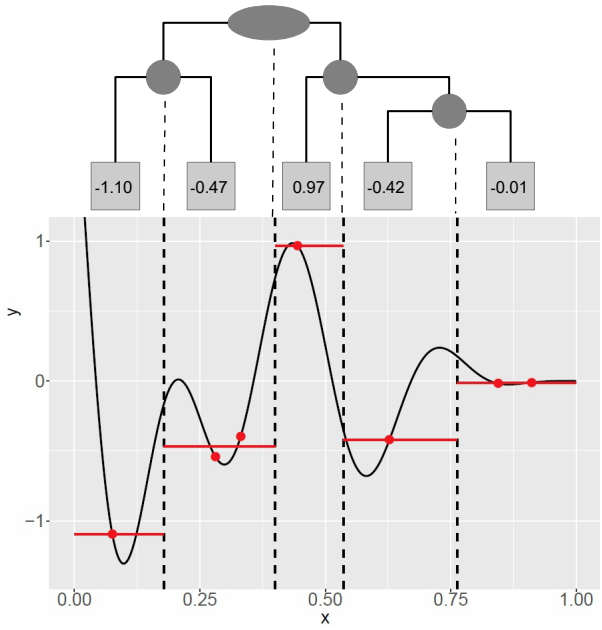

Abbildung 7 Oben ist die schematische Darstellung eines Regressionsbaumes zu sehen. Darunter die zugehörigen Splitpunkte (gestrichelte Linien) und Schätzungen innerhalb der Funktion. Durchgezogene schwarze Linie: Beispielfunktion. Rote Linien: Schätzungen des Baums. Rote Punkte: Trainingsdaten.

Als nächstes werden nun diese Teilmengen aufs Neue versucht, bestmöglich aufzuteilen. Teilmengen, die weiter aufgeteilt werden, werden auch als Knoten bezeichnet. Die Splits sind wieder durch gestrichelte Linien von Kreis zu x-Achse dargestellt. Wie man gut erkennen kann, fasst der Baum Punkte mit ähnlichen Responsewerten zusammen. Und zwar wird der zweite und dritte Punkt von links zusammengefasst und ebenso der höchste Punkt von den anderen Punkten im rechten Teil isoliert, da er sich stark von den anderen unterscheidet. Die Teilmenge ganz rechts wird anschließend nochmals in zwei Teilmengen gesplittet, wobei wieder Punkte mit ähnlichen Responsewerten zusammengefasst werden. Am Ende dient nun in jeder Teilmenge das arithmetische Mittel als Vorhersagewert für neue Werte, die in das zugehörige Intervall der Teilmenge fallen. In der Grafik sind die Vorhersagewerte im End-Knoten, dem sogenannten Blatt, als Werte eingetragen und in den dazugehörigen Funktionsabschnitten durch rote Linien gekennzeichnet. Aufgrund von Overfitting möchte man Bäume normalerweise nicht bis zur perfekten Homogenität verzweigen lassen. Oft werden Abbruchkriterien

wie die Voraussetzung einer Mindestmenge an Punkten in der Teilmenge, oder ein erreichter Wert des Fehlers eingesetzt. Ein weiteres Verfahren ist das Zurückschneiden eines voll verzweigten Baumes, auch Pruning genannt. Wir wollen hier aber nur auf die absoluten Grundlagen zur Erstellung eines Regressionsbaums eingehen, die notwendig sind, um die nun folgenden Erklärungen zu Random Forests verstehen zu können.
Wenden wir uns also nun dem Random Forest zu. Random Forests gehören zu den Ensemble Methoden. Wie der Name sagt, bilden Ensemble Methoden ein Ensemble an Schätzern, den sogenannten Base-Learnern. Der eigentliche Schätzer wird gebildet, in dem über das Ensemble aggregiert wird. Durch diese Aggregation vieler schwächerer Schätzer soll die Vorhersagegenauigkeit des Modells verbessert werden. Random Forests sind nahe verwandt zu Bagging. Die Idee bei dieser Methode ist, dass durch Base-Learner mit hoher Varianz aber geringem Bias, durch die Aggregation die Varianz reduziert werden kann und sich so die Vorhersagegenauigkeit verbessert.
Das übliche Vorgehen bei Random Forests sieht so aus, dass zuerst Bootstrap-Stichproben durch Ziehen mit Zurücklegen aus den Trainingsdaten gebildet werden. Meistens ist dabei eine Bootstrap-Stichprobe genauso groß wie die Trainingsdaten. Anschließend wird nun auf jeder Bootstrap-Stichprobe ein Base-Learner gefittet. Als Base-Learner werden bei Random Forests einzelne Bäume verwendet. Entscheidend ist, dass bei der Erstellung der Bäume nicht wie normalerweise alle vorhandenen Variablen als mögliche Splitvariablen in einem Knoten zur Verfügung gestellt werden, sondern nur eine begrenzte Anzahl. Diese Splitvariablen p werden bei jedem Knoten zufällig aus allen m Kovariablen gezogen. Falls $p = m$ gilt, entspricht der Random Forest dem Bagging. Am Schluss wird über alle Base-Learner aggregiert und so der eigentliche Schätzer gebildet.
Bei Random Forests ist das Prinzip, dass sich durch eine hohe Dekorrelation der Bäume die Stabilität des Schätzers verbessert. Das liegt daran, dass die Varianz der Vorhersage

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2 \tag{8}$$

von der paarweisen Korrelation ρ zwischen den Bäumen abhängt, mit σ^2 als der Varianz der Bäume und B als Anzahl der Bootstrapstichproben, bzw. Bäumen. Bei einer hohen Anzahl an Bäumen kann der hintere Term vernachlässigt werden und man kann sich auf den vorderen Term konzentrieren. Hier zeigt sich, dass die Korrelation zwischen den Bäumen eine große Rolle spielt. Je kleiner sie ist, desto größer ist der Profit der durch die Ensemble Bildung entsteht, da dadurch die Varianz der Vorhersage reduziert werden kann. Deshalb wird bei Random Forests versucht,

die Korrelation zwischen den Bäumen möglichst stark zu reduzieren, ohne dabei jedoch die Varianz der Bäume zu erhöhen. Erreicht wird das zum Beispiel durch eine Reduzierung der Anzahl möglicher Splitvariablen und die Verwendung von Bootstrapstichproben. In Abbildung 8 ist der Aufbau eines Random Forests grafisch

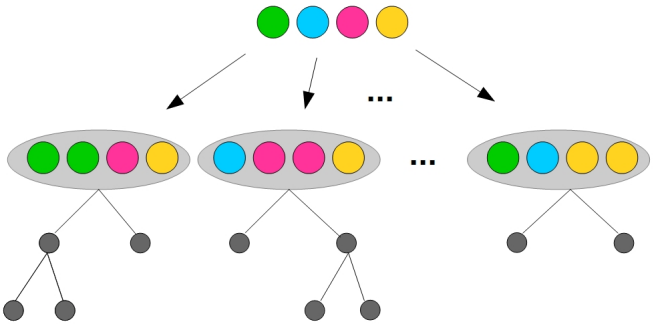

Abbildung 8 Schematische Darstellung eines Random Forests

veranschaulicht. Aus den Ausgangsdaten, dargestellt durch die obersten farbigen Kreise, wird eine bestimmte Anzahl an Bootstrapstichproben gebildet (farbige Kreise innerhalb der Ovale). Auf jeder Bootstrapstichprobe wird anschließend ein Baum gefittet. Jede Bootstrapstichprobe stellt bei Random Forests also eine Wurzel eines Baums dar. Der Random Forest Schätzer $\hat{\theta}^B(x)$ lässt sich aus den verschiedenen Bäumen folgendermaßen berechnen:

$$\hat{\theta}^B(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{t}^{*b}(x) \tag{9}$$

B: Anzahl der Bootstrapstichproben, bzw. Bäume
t^{*b}: Ein Baum auf der Bootstrapstichprobe b
x: Eingabewert

Der Random Forest Schätzer entsteht also durch die Durchschnittsbildung über alle gefitteten Bäume.
Schauen wir uns die Erstellung eines Random Forests am besten noch anhand eines Beispiels an, wobei wieder auf den Regressionsfall eingegangen wird. Als Beispiel sehen wir uns die gleiche Funktion wie bei der Erklärung der Bäume an. Wie vorhin liegen uns bereits sieben ausgewertete Punkte als Trainingsdaten vor. Zuerst einmal nehmen wir an, wir wollen nur zwei Bäume für unseren Random Forest Schätzer verwenden. Die obere Hälfte von Abbildung 9 zeigt die Bildung des ersten Baums. In der Bootstrapstichprobe sind nur die roten Trainingspunkte enthalten (manche doppelt), womit die grünen Punkte nicht für den Fit verwendet werden. In den Blättern steht wieder jeweils der Schätzer des Baums,

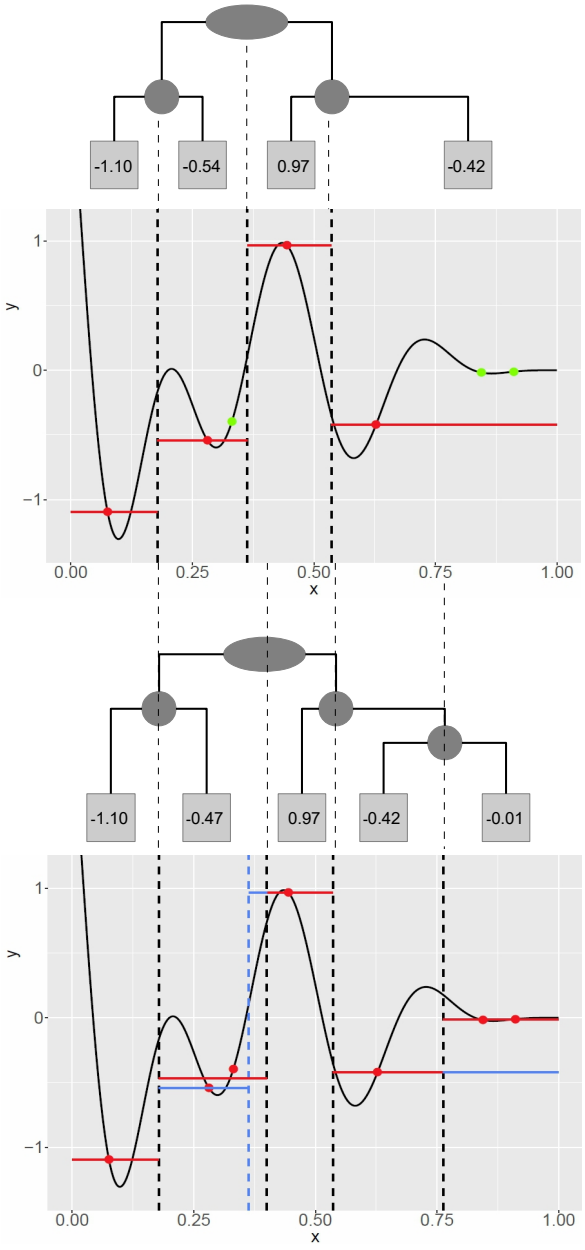

Abbildung 9 Zwei Bäume eines Random Forests. Splitpunkte: gestrichelte Linie, Trainingsdaten: rote Punkte, Trainingspunkte nicht in der Bootstrap-Stichprobe: grüne Punkte, Schätzungen: rote Linien und Werte in den Blättern. Beim zweiten Baum sind zusätzlich die Abweichungen im Vergleich zum ersten Baum blau eingezeichnet.

der sich aus dem Mittelwert aller Beobachtungen innerhalb des Blattes ergibt. Die untere Hälfte der Grafik zeigt den Fit des zweiten Baums. Wie man erkennen kann, unterscheiden sich die Splitpunkte im Vergleich zum ersten Baum und damit auch die Mittelwerte in manchen Blättern. Die Unterschiede lassen sich hier auf die unterschiedlichen Bootstrapstichproben zurückführen. Eine vorherige Variablenselektion für die Berechnung der Splitpunkte findet hier nicht statt, da wir uns im eindimensionalen Fall befinden und somit nicht zufällig eine bestimmte Variablenanzahl auswählen können. Die blauen Linien zeigen die Schätzer des ersten Baumes, wenn sie sich von denen des zweiten Baums unterscheiden. Die blaue gestrichelte Linie zeigt den Splitpunkt des ersten Baumes der sich von denen des zweiten Baumes unterscheidet. Um nun den Random Forests Schätzer zu bilden, fasst man die Bäume mithilfe von Durchschnittsbildungen zusammen. Dazu müssen die Splitpunkte beider Trees berücksichtigt werden. In jedem Abschnitt gilt dabei der Mittelwert von den Blättern der beiden Bäume als Prognosewert. Übersicht halber sind in der Abbildung pro Intervall nur die Prognosewerte der einzelnen Bäume, aber nicht deren Durchschnittswerte eingezeichnet. Man kann sich nun schon denken, dass bei Verwendung von mehr als zwei Bäumen die Abschnitte noch feiner unterteilt werden, da noch mehr verschiedene

Splitpunkte hinzukommen. Die Übergänge des Schätzers werden also glatter, je mehr Bäume für die Schätzung verwendet werden. Am besten lässt sich die Auswirkung der Anzahl der Bäume im zweidimensionalen Fall erkennen. In Abbildung 10 ist als Beispiel die Hosaki-Funktion verwendet worden. Die linke Grafik zeigt die wahre Funktion, die nachgebildet werden soll. Splitpunkte können nun sowohl auf der Abzissenachse, als auch auf der Ordinatenachse gesetzt werden, da nun zwei mögliche Splitvariablen vorhanden sind. Die Farbskala zeigt den Wertebereich der Response-

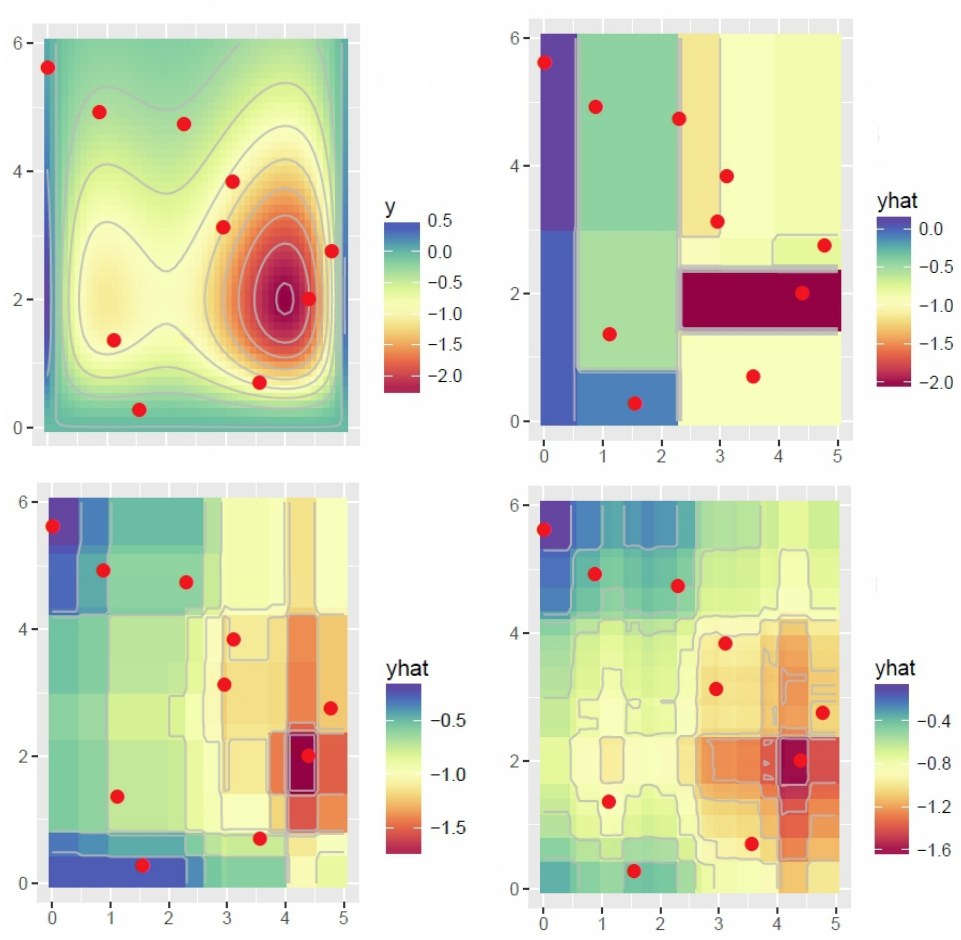

Abbildung 10 Random Forest Schätzung der Hosaki-Funktion bei Verwendung unterschiedlicher Anzahlen an Bäumen. Links oben sind die wahren Responsewerte der Hosaki-Funktion dargestellt, rechts oben die Schätzung mit RF bei Verwendung von zwei Bäumen, links unten bei neun und rechts unten bei Verwendung von 500 Bäumen.

werte. Rechts sehen wir die Schätzung mit dem Random Forest bei der Verwendung von zwei Bäumen, links unten bei neun und rechts unten mit 500 Bäumen. Man kann hier sehr gut erkennen, dass die Übergänge der Schätzungen weicher werden, je mehr Bäume man verwendet. Aus diesem Grund empfiehlt es sich eine große Anzahl an Bäumen zu verwenden, da man dadurch zu besseren Prognosen kommt. Erinnern

wir uns hier auch nochmal an die Formel zur Varianz der Vorhersage, bei der der hintere Term nur bei einer großen Anzahl an Bäumen eine unbedeutende Rolle spielt. Allerdings ist die Verbesserung der Schätzung ab einer gewissen Anzahl minimal bis nicht mehr gegeben. Dafür kann sich die Rechenzeit stark erhöhen, weshalb eine zu hohe Anzahl eher negative Auswirkungen haben kann. In der Praxis gibt es keine feste Regel, wie viele Bäume sinnvoll sind. Eine interessante Untersuchung hierzu gibt es von T. Oshiro, P. Perez und J. Baranauskas [12], die Experimente mit 29 realen Datensätzen durchführten. Sie fanden heraus, dass es ab 128 Bäumen keinen signifikanten Unterschied mehr zwischen einer Verwendung von 1024, 2048 oder 4096 Bäumen gibt. Aufgrund der Ergebnisse aus den Experimenten schlagen die Wissenschaftler eine Anzahl zwischen 64 und 128 Bäumen vor. In der Praxis wird jedoch meistens eine größere Anzahl an Bäumen verwendet.
Was man bei der Grafik auch erkennen kann ist, dass Random Forests nicht so gut geeignet sind, um die runde Form der Hosaki-Funktion wirklich gut abzubilden. Für SMBO brauchen wir jedoch nicht unbedingt eine gute Nachzeichnung der wahren Funktion [13]. Wichtig ist nur, dass das Optimum gefunden werden kann.
Ein zweiter Aspekt, der bei Random Forests beachtet werden kann, ist die Tiefe der Bäume. Wir haben uns bereits angesehen, dass die Bäume wenig korreliert sein sollten, um die Varianz der Prognose so gut wie möglich zu reduzieren. Aus diesem Grund empfiehlt es sich auch, die Bäume bis zum Ende wachsen zu lassen und nicht zurückzuschneiden, da sich bei einer geringeren Baumtiefe die Bäume weniger voneinander unterscheiden, bzw. eine höhere Korrelation aufweisen. Allerdings ist zu erwähnen, dass es bei einer maximalen Tiefe der Bäume zu einem, wenn auch geringem, Overfitting kommen kann. Jedoch ist der Effekt, den man durch eine Anpassung der Baumtiefe erzielen kann, meistens nur sehr gering [5].
Ein weiterer Hyperparameter bei Random Forests ist die Anzahl möglicher Splitvariablen p. Je kleiner p gewählt wird, desto dekorrelierter sind die Bäume, da sie sich stärker voneinander unterscheiden können. Ein sehr kleiner p Wert liefert allerdings nicht immer die besten Ergebnisse. Falls nämlich die Anzahl der Variablen groß ist, die Anzahl relevanter Variablen jedoch klein ist, ist die Wahrscheinlichkeit, bei einem Split eine relevante Variable zu wählen, sehr gering. Hier wäre die Performance eines Random Forests mit sehr kleinem p wahrscheinlich eher schlecht. In diesem Fall sollte p nicht zu klein gewählt werden [14, 15]. Für Klassifikationsbäume wird meistens ein Wert von $\sqrt{p}$ und für Reggressionsbäume ein Wert von $\frac{p}{3}$ verwendet.
Auch bei Random Forests ist die Schätzung der Varianz von Vorhersagen von Bedeutung. Eine Möglichkeit ist es, die Varianz zwischen den einzelnen Schätzungen der Bäume zu bestimmen. Und zwar durch $\hat{\sigma}^2(x) = \frac{1}{B}\sum_{b=1}^{B}(\hat{t}^{*b}(x) - \hat{\theta}^B(x))^2$, wobei

$\hat{t}^{*b}(x)$ der Schätzer des Baums auf der Bootstrap-Stichprobe b ist und $\hat{\theta}^B(x)$ der Random Forest Schätzer. Im Folgenden ist diese Methode als einfache Berechnung der Varianz, bzw. Standardabweichung ($\hat{\sigma}(x) = \sqrt{\hat{\sigma}^2(x)}$), bezeichnet. Weitere Methoden sind zum Beispiel der Noisy-Bootstrap Schätzer oder die weniger rechenintensiven Schätzer Jackknife-after-Bootstrap und Infinitesimal-Jackknife. Einen Überblick verschiedener Unsicherheitsschätzer findet man in der Seminararbeit [16], oder genauere Ausführungen in den Papern [15, 17].
Die Varianzschätzer haben eine Gemeinsamkeit. Und zwar nimmt die Varianz in unbeobachteten Bereichen nicht zu, sondern zeigt sich eher gleichmäßig im Parameterraum. Die Probleme, die dadurch bei SMBO entstehen, werden in Kapitel 3.1 aufgezeigt. Im folgenden kommen wir zu einem Auszug aus der Seminararbeit, der sich auf [15] bezieht. Dieser behandelt die Varianzschätzung mit dem Jackknife-after-Bootstrap-Schätzer, der später bei den Experimenten (Kapitel 4) Verwendung fand. Für die Varianzschätzung mit dem Jackknife-after-Bootstrap-Schätzer werden bei Random Forests die Bootstrap-Stichproben des aggregierten Schätzers verwendet. Die Idee dabei ist, dass eine Bootstrap-Stichprobe nur im Schnitt aus 2/3 der Beobachtungen der Ausgangsdaten besteht. Somit kann eine Bootstrap-Stichprobe wie eine Jackknife-Stichprobe behandelt werden. Eine Jackknife-Stichprobe ist normalerweise eine Leave-One-Out Stichprobe, bei der in jeder Stichprobe eine andere Beobachtung ausgeschlossen wird. Der Jackknife-Varianzschätzer mit Leave-One-Out Stichproben wird auch als „delete-one“ Jackknife bezeichnet. Diese Variante findet in der Praxis am häufigsten Verwendung. Es ist jedoch auch ein „delete-d“Jackknife möglich, bei dem eine beliebige Anzahl an Beobachtungen in jeder Stichprobe weggelassen wird. Der passende Name des Jackknife-Varianzschätzers für Random Forests kommt daher, da man den Jackknife Schätzer nach der Bootstrapziehung anwendet, also Jackknife-after-Bootstrap. Zur Veranschaulichung sind in Abbildung 11 Bootstrap-

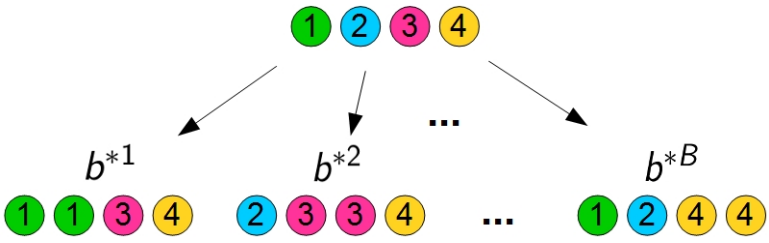

Abbildung 11 Bootstrap-Stichproben n=4

Stichproben dargestellt. Die erste Stichprobe kann beispielsweise als Leave-one-out Stichprobe angesehen werden, bei der die zweite, bzw. blaue Kugel ausgeschlossen wird. Beim Jackknife-after-Bootstrap Varianzschätzer lässt sich die Varianz

folgendermaßen schätzen:

$$\hat{V}_J^B = \frac{n-1}{n} \sum_{i=1}^{n} [\hat{\theta}_{(-i)}^B(x) - \hat{\theta}^B(x)]^2 \tag{10}$$

mit

$$\hat{\theta}_{(-i)}^B(x) = \frac{\sum_{\{b:N_i^{*b}=0\}} \hat{t}^{*b}(x)}{|\{N_i^{*b} = 0\}|}$$

Dabei gibt B die Anzahl der Bootstrap-Stichproben, n die Anzahl der Beobachtungen in den Ausgangsdaten, $\hat{\theta}_{(-i)}^B(x)$ den Aggregierten Schätzer auf allen Stichproben ohne Beobachtung i, $\hat{\theta}^B(x)$ den Aggregierten Schätzer, N_i^{*b} die Anzahl der Beobachtung i in der Bootstrap-Stichprobe b und $\hat{t}^{*b}(x)$ den Base-Learner auf der Bootstrap-Stichprobe b an. In obiger Abbildung würde zum Beispiel der Aggregierte Schätzer ohne die zweite Beobachtung ($\hat{\theta}_{(-2)}^B(x)$) die Bootstrap-Stichprobe b^{*1} verwenden. Außerdem natürlich noch alle anderen Bootstrap-Stichproben, die ebenfalls die zweite Beobachtung nicht enthalten und hier zwischen b^{*2} und b^{*B} auftreten.
Der Varianzschätzer berechnet dann die quadratische Abweichung zwischen dem Aggregierten Schätzer ohne die i-te Beobachtung und dem normalen Aggregierten Schätzer, der über alle Stichproben berechnet wird. Also wird beim Jackknife-after-Bootstrap untersucht, wie viel Variabilität die i-te Beobachtung beträgt. Zu beachten ist, dass das B in der Formel keinen Exponenten darstellt, sondern nur zur Information dient, wie viele Stichproben verwendet wurden.
Da für die Schätzung nur eine endliche Anzahl an Bootstrap-Stichproben verwendet werden kann, entsteht zusätzlich zur Stichproben-Variabilität ein Monte-Carlo-Fehler als Quelle der Variabilität. Um dem Fehler zu begegnen, empfiehlt es sich, eine Bias-Korrektur in den Varianzschätzer einzufügen. Der Jackknife-after-Bootstrap Schätzer mit Korrektur ergibt sich als

$$\hat{V}_{J-U} = \hat{V}_J^B - (e-1)\frac{n\hat{\sigma}^2}{B}, \tag{11}$$

mit

$$\hat{\sigma}^2 = \frac{1}{B} \sum_{b=1}^{B} (\hat{t}^{*b}(x) - \hat{\theta}^B(x))^2.$$

Durch den Varianzschätzer kann nun bestimmt werden, wie unsicher die Schätzung in bestimmten Bereichen ist.

Nach den Random Forests wenden wir uns nun Extra-Trees als ein weiteres mögliches Surrogat-Modell zu. Dieses Modell ist stark verwandt zu Random Forest.

2.2.3 Extra-Trees

Falls nicht anders vermerkt, bezieht sich das folgende Kapitel auf [14].
Wie bei Random Forest wird auch bei Extra-Trees ein Ensemble an Bäumen gebildet und durch deren Aggregation der eigentliche Schätzer erstellt. Der Unterschied liegt darin begründet, dass Extra-Trees die Splitpunkte bei den Variablen völlig zufällig setzt. Außerdem erstellt der Algorithmus die Bäume nicht auf Bootstrap-Stichproben, sondern verwendet dafür alle Trainingsdaten. Der Gedanke dahinter ist zum einen, dass dadurch die Varianz der Schätzung stärker reduziert werden kann und zum anderen, dass sich eine geringere Rechenzeit ergibt.
Bei der Teilung eines Knotens für die Baumerstellung geht Extra-Trees so vor, dass zuerst eine bestimmte Anzahl p an Kovariablen aus allen m vorhandenen Kovariablen zufällig gezogen werden. Im Anschluss legt Extra-Trees einen Splitpunkt bei jeder Variablen per Zufall fest. Die Variable, bei der der Split die homogensten Teilmengen bezüglich der Responsevariablen bildet, wird schließlich als Splitvariable für die Teilung des Knotens verwendet.
Falls $p = 1$ gilt, also nur eine Kovariable ausgewählt wird, findet die Teilung des Knotens total zufällig statt und hängt nicht mehr vom Responsewert ab. In diesem Fall spricht man auch von total randomisierten Bäumen. Werden dagegen alle vorhandenen Kovariablen verwendet ($p = m$), so wird der Zufall alleinig durch die zufällige Auswahl der Splitpunkte erzeugt.
Die stärkere Varianzreduzierung bei Extra-Trees gegenüber Random Forests soll aufgrund der Randomisierung der Splitpunkte erreicht werden. Durch den größeren Zufall bei der Baumerstellung sollte es möglich sein, die Varianz durch die Ensemble-Bildung noch mehr zu senken. Die Verwendung des ganzen Trainingssatzes, anstatt der Bootstrap-Stichproben dient dazu, zusätzlich den Bias klein zu halten.
Aufgrund der zufälligen Wahl der Splitpunkt spart man außerdem Rechenzeit ein, da keine Berechnungen für die Suche durchgeführt werden müssen.
Wie bereits in Kapitel 2.2.2 beschrieben, hängt die Güte der Konfiguration des Hyperparameters p auch von der Anzahl irrelevanter Kovariablen in den Daten ab. Bei einer hohen Anzahl irrelevanter Kovariablen kann ein höherer Wert von p zu einer besseren Güte der Schätzung führen. Begründen lässt es sich damit, dass unbedeutende Kovariablen herausgefiltert werden, wenn sich der Algorithmus zwischen mehreren Kovariablen entscheiden kann. Durch einen größeren Wert p erhöht sich zwar die Varianz, aber es lässt sich der Bias senken, der bei Verwendung

von irrelevanten Variablen als Splitvariablen entstehen würde. Untersuchungen in [14] zeigten, dass bei Regression ein Wert von $p = m$ in den meisten Fällen zu den besten Ergebnissen bei Extra-Trees führt.
Eine Erweiterung für Extra-Trees wird im R Paket *extraTrees*[39] vorgeschlagen. Anstatt einen einzigen zufälligen Splitpunkt auszuwählen, besteht hier die Möglichkeit, mehrere zufällige Splitpunkte zu setzten. Anschließend kann man daraus den besten Splitpunkt verwenden. Durch dieses Vorgehen soll die Güte von Extra-Trees verbessert werden können, da die Chance geringer ist, einen sehr schlechten Splitpunkt aufgrund der zufälligen Legung zu erhalten.

2.3 Infill-Kriterien

Infill-Kriterien dienen bei SMBO dazu, neue Funktionsstellen vorzuschlagen, an denen die Target-Funktion ausgewertet werden soll (s. h. Kapitel 2.1). Da man möglichst wenige teure Auswertungen der Target-Funktion durchführen möchte, soll das Infill-Kriterium möglichst nur Punkte vorgeschlagen, die den größten Gewinn in Richtung Optimum versprechen. Dafür ist einerseits das Wissen über bereits ausgewertete Punkte hilfreich, andererseits ist die Erkundung unbekannter Bereiche der Target-Funktion interessant, da sich hier ein Optimum verbergen könnte. Um diesem Trade-Off zwischen der Ausnutzung von bisherigem Wissen und Erforschung des Raums zu begegnen, verwenden Infill-Kriterien häufig den Mittelwert und die Varianz des Surrogat-Modells.
Um die Funktionsstelle zu finden, die das Infill-Kriterium als bestmöglichen Vorschlag für eine neue Auswertung ansieht, muss es optimiert werden. Auf die Optimierung des Infill-Kriteriums wird in Kapitel 2.4 eingegangen. Im Folgenden werden nun zuerst verschiedene Infill-Kriterien behandelt, darunter die Verwendung des Mittelwerts (Kapitel 2.3.1), Probability of Improvement (Kapitel 2.3.2) und das Expected Improvement (Kapitel 2.3.3).

2.3.1 Mittelwert

Bei Verwendung des Mittelwerts als Infill-Kriterium wird nur der geschätzte Mittelwert des Surrogat-Modells verwendet und nicht dessen Standardabweichung bzw. Varianz. Man versucht hier, das globale Optimum der Target-Funktion zu finden, indem bei jeder SMBO Iteration die Funktionsstelle mit dem besten Wert des Surrogat-Modells für eine neue Auswertung der Target-Funktion vorgeschlagen wird. Wenn die Target-Funktion also minimiert werden soll, schlägt das Infill-Kriterium eine neue Auswertung am Minimum des Surrogat-Modells vor. Bei Maximierung

der Target-Funktion schlägt es die neue Auswertungsstelle beim Maximum des Surrogat-Modells vor. Ein Nachteil dieser Methode ist, dass sie die Unsicherheit des Surrogat-Modells nicht miteinbezieht. Somit berücksichtigt sie nicht, dass es passieren kann, dass das Surrogat-Model die Target-Funktion in manchen Bereichen nicht ausreichend nachbildet. Dieser Fall ist in Abbildung 12 dargestellt. Zu sehen sind

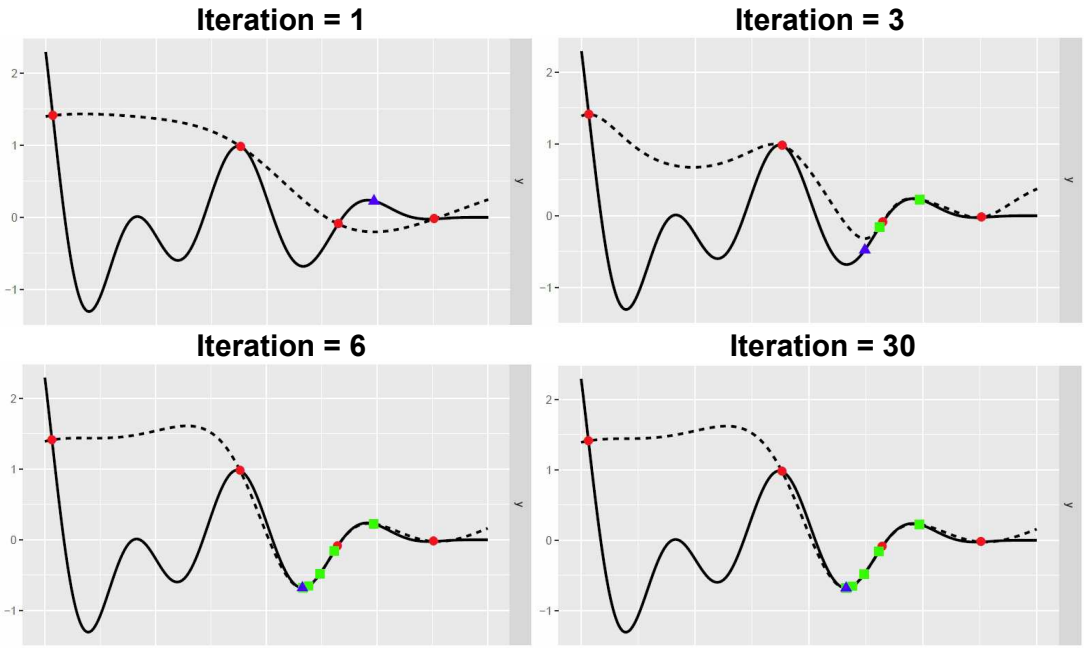

Abbildung 12 SMBO mit GPR als Surrogat-Modell und der Verwendung des geschätzten Mittelwerts (gestrichelte Linie) als Infill-Kriterium. Optimiert wird eine multimodale Target-Funktion[1] (durchgezogene Linie). Initiale Designpunkte sind als rote Punkte, Vorschläge als blaue Dreiecke und Designpunkte als grüne Quadrate dargestellt.

hier verschiedene SMBO Iterationen einer Ausführung mit GPR und dem Mittelwert der prädiktiven Posterioriverteilung als Infill-Kriterium. Die Target-Funktion ist hier durch die schwarze Linie dargestellt. Die gestrichelte Linie zeigt jeweils den geschätzten Mittelwert des Surrogat-Modells auf den bereits ausgewerteten Punkten. In der ersten Iteration wird der Vorschlag (blaues Dreieck) zwischen die beiden rechten Punkte gesetzt. Das ist keine schlechte Entscheidung, da der Vorschlag nicht allzu nahe an den Designpunkten liegt. In den folgenden Iterationen erfolgt die Suche jedoch nur sehr lokal. Der Algorithmus tastet sich Schritt für Schritt in das lokale Optimum der Target-Funktion und setzt sich darin schließlich fest. Der linke Bereich der Target-Funktion wird nicht weiter untersucht, da das Surrogat-Model die Funktionswerte hier höher einschätzt. Die größere Unsicherheit, die hier besteht, wird nicht berücksichtigt. Dadurch, dass nicht weiter exploriert wird, kann in diesem

[1] Die genaue Beschreibung der Target-Funktion ist dem Anhang zu entnehmen.

Fall das globale Optimum nicht gefunden werden. Das initiale Design spielt bei diesem Infill-Kriterium eine große Rolle. Liegen die Punkte des Designs nicht schon in der Nähe des globalen Optimums, läuft man Gefahr, dass das Surrogat-Model das globale Optimum nicht erfassen kann. Dadurch werden neue Vorschläge nur in einem lokalen Optimum gemacht und das globale Optimum verfehlt. Somit können vor allem bei einem kleinen initialen Design in Kombination mit einer multimodalen Target-Funktion Probleme bei Verwendung des Mittelwerts als Infill-Kterium auftreten.

2.3.2 Probability of Improvement

Falls nicht anders vermerkt, bezieht sich das folgende Kapitel auf [9].
Wie in Kapitel 2.3.1 zu sehen ist, besteht bei Verwendung des Mittelwerts als Infill-Kriterium die Gefahr, dass der Optimierungslauf von SMBO in einem lokalen Optimum stagniert. Aus diesem Grund hat man Kriterien entwickelt, die die Varianz des Surrogats mit einbeziehen, um dadurch eine bessere Exploration des Raums zu erreichen. Eines solcher Kriterien ist das Probability of Improvement, das 1964 von Kushner vorgeschlagen wurde. Übersetzen lässt es sich als die Wahrscheinlichkeit der Verbesserung. Und damit sagt es bereits aus, dass man die Funktionsstelle finden möchte, bei der es am wahrscheinlichsten ist, dass die Target-Funktion einen besseren Responsewert annimmt, als die bereits ausgewerteten Funktionsstellen. Mit besser ist dabei gemeint, dass man z. B. bei Minimierung der Target-Funktion einen kleineren Responsewert erhält.
Wir nehmen an, dass die Target-Funktion minimiert werden soll und $f(x^-)$ dabei den aktuell besten Wert der ausgewerteten Punkte darstellt. Legt man nun einen Zielwert $T < f(x^-)$ fest, den man mindestens als Verbesserung erreichen möchte, so ist die Wahrscheinlichkeit einer Verbesserung nichts anderes als die Wahrscheinlichkeit $f(x) \leq T$, d. h. die Wahrscheinlichkeit, dass der wahre Responsewert an der Stelle x kleiner ist als der festgelegte Targetwert T.
Man modelliert nun die Unsicherheit, die man über die Funktionswerte der Target-Funktion hat, indem man sie als Realisationen einer normalverteilten Zufallsvariablen $f(x)$ mit Mittelwert $\mu(x)$ und Varianz $\sigma(x)^2$ bzw. Standardabweichung $\sigma(x)$ betrachtet. Durch diese Annahme lässt sich die Wahrscheinlichkeit der Verbesserung, bzw. das Probability of Improvement, darstellen durch

$$PI(x) = \Phi\Big(\frac{T - \hat{\mu}(x)}{\hat{\sigma}(x)}\Big), \tag{12}$$

wobei Φ die Verteilungsfunktion der standard Normalverteilung symbolisiert. Mittel-

wert und Varianz können mit Hilfe des Surrogat-Modells geschätzt werden.
Wenn man die Target-Funktion in der direkten Nähe des besten Punktes mehrmals weiter auswertet, wird die Standardabweichung in diesem Bereich sehr klein. Daraus folgt, dass auch das $PI(x)$ in diesem Bereich klein wird, da der Term $\frac{T-\hat{\mu}(x)}{\hat{\sigma}(x)}$ einen stark negativen Wert annimmt (normalerweise gilt $T < \hat{\mu}(x)$). Man ist sich aufgrund der mehrmaligen Auswertungen in einem Bereich also ziemlich sicher über die dortigen wahren Werte der Target-Funktion. Bereiche mit einer höheren Unsicherheit werden nun interessanter, da hier noch die Chance besteht sich stärker gegenüber den aktuellen Werten zu verbessern. Auch wenn es sein kann, dass in diesem Bereich der geschätzte Mittelwert schlechter als der aktuell beste Wert ist.

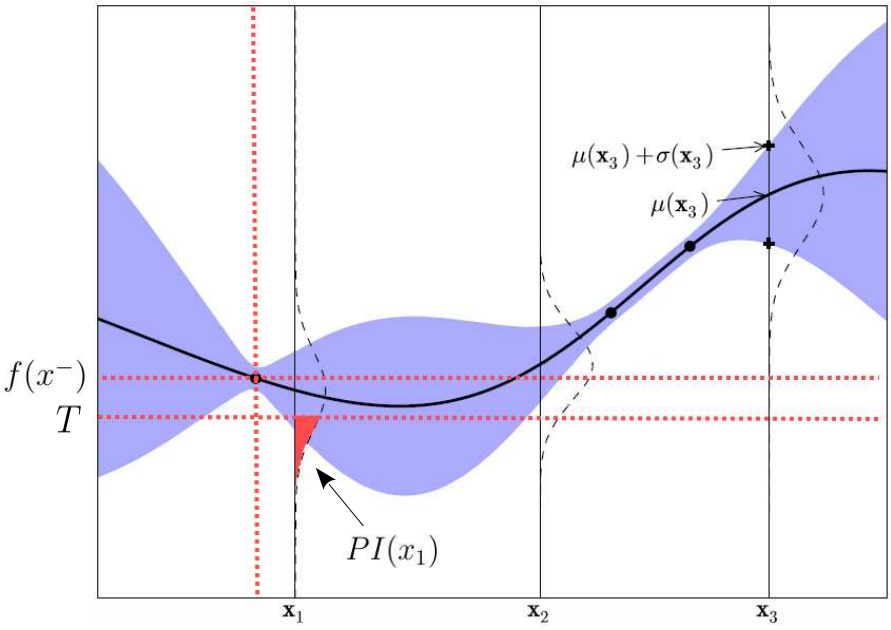

Abbildung 13 GPR mit PI. $f(x^-)$ nimmt den kleinsten Wert der drei Trainingspunkte an. T bezeichnet den festgelegten Zielwert, der mindestens erreicht werden soll. Das PI an der Stelle x_1 ist rot eingefärbt. Die Durchgezogene schwarze Linie ist der geschätzte Mittelwert der GPR, die blaue Schattierung die Standardabweichung. Quelle: [1][2]

Zur Veranschaulichung des PI ist in Abbildung 13 die gleiche Grafik mit einer GPR wie in Kapitel 2.2.1 dargestellt, nur dass nun das PI eingezeichnet ist. Der linke Designpunkt besitzt den niedrigsten Responsewert aller bereits ausgewerteten Punkte. Somit ist das der aktuell beste Wert $f(x^-)$. Darunter wird ein Zielwert T festgelegt, den man mindestens als Verbesserung erreichen möchte. Die Wahrscheinlichkeit, bei Punkt x_1 besser als T zu sein, d.h. einen niedrigeren Wert der Target-Funktion als T zu erhalten, ist in der Grafik rot eingefärbt. Das PI ist hier größer als bei den anderen

[2]Die Grafik aus [1] wurde bearbeitet.

beiden Punkten x_2 und x_3. Die größte Wahrscheinlichkeit für eine Verbesserung ergäbe sich in etwa der Mitte zwischen x_1 und x_2. Der Vorschlag wäre also in diesem Fall nicht viel anders als bei Verwendung des Mittelwerts als Infill-Kriterium. Falls es jedoch einen bereits ausgewerteten Punkt im aktuellen Minimum gäbe, so wäre dieser Bereich für das PI nicht mehr der interessanteste, da dort die Varianz nur noch sehr gering wäre. Man würde dann eher nach ganz links außen explorieren, obwohl der Mittelwert hier höher ist.
Auf den ersten Blick wirkt die Methode ideal, um sowohl das Wissen über bereits ausgewertete Punkte, als auch die weitere Erkundung des Raums mit einzubeziehen. Das Probability of Improvement hat jedoch den großen Nachteil, dass es sehr sensibel auf die Wahl des Targetwerts reagiert. Wird die gewünschte Verbesserung zu klein gewählt, erfolgt kaum die Erkundung des Raums. Erst nach einer umfangreichen Suche in der Nähe des aktuell besten Punktes käme es wieder zu einer globalen Suche. Aber auch die Wahl eines großen Wertes ist ungünstig, da die Suche überwiegend global erfolgt und eine Feinkonfiguration einer vielversprechenden Lösung nur sehr langsam vonstatten geht. Um die Wahl von T zu umgehen empfiehlt es sich, ein alternatives Infill-Kriterium zu verwenden. Eine Möglichkeit ist das im nächsten Kapitel vorgeschlagene Expected Improvement.

2.3.3 Expected Improvement

Das Probability of Improvement berücksichtigt die Wahrscheinlichkeit einer Verbesserung, wohingegen das Expected Improvement auf den erwarteten Wert der Verbesserung eingeht. Im Folgenden nehmen wir wieder an, dass die Target-Funktion minimiert werden soll. Die Verbesserung lässt sich dann durch die Differenz des aktuell besten Funktionswerts aller Designpunkte $f(x^-)$ und des zu untersuchenden Funktionswerts $f(x_{t+1})$ ausdrücken $(f(x^-) - f(x_{t+1}))$. Logischerweise sind nur positive Werte einer Verbesserung sinnvoll, womit sich die Verbesserung (Improvement) durch

$$I(x) = \max\{0, f(x^-) - f(x_{t+1})\} \tag{13}$$

beschreiben lässt [1]. Nun ist wie schon beim PI die Annahme, dass es sich bei den unbekannten Funktionswerten um normalverteilte Zufallsvariablen mit Mittelwert $\mu(x)$ und Varianz $\sigma^2(x)$ handelt. Mit Hilfe dieser Annahme lässt sich für die Verbesserung I die Wahrscheinlichkeit über die Dichtefunktion der Normalverteilung

$$\frac{1}{\sqrt{2\pi}\hat{\sigma}(x)} \exp\Big[- \frac{(f(x^-) - I - \hat{\mu}(x))^2}{2\hat{\sigma}^2(x)}\Big] \tag{14}$$

darstellen. Der Mittelwert und die Varianz werden dabei vom Surrogat-Modell geschätzt. Durch eine Integration über diese Dichte erhält man den Erwartungswert der Verbesserung, also das Expected Improvement [9]:

$$E(I) = \int_{I=0}^{I=\inf} I\Big\{\frac{1}{\sqrt{2\pi}\hat{\sigma}(x)} exp\Big[-\frac{(f(x^-) - I - \hat{\mu}(x))^2}{2\hat{\sigma}^2(x)}\Big]\Big\} dI \tag{15}$$

$$= \hat{\sigma}(x)\Big[\frac{f(x^-) - \hat{\mu}(x)}{\hat{\sigma}(x)}\Phi\Big(\frac{f(x^-) - \hat{\mu}(x)}{\hat{\sigma}(x)}\Big) + \phi\Big(\frac{f(x^-) - \hat{\mu}(x)}{\hat{\sigma}(x)}\Big)\Big] \tag{16}$$

Dabei symbolisiert ϕ die Dichtefunktion und Φ die Verteilungsfunktion der standard Normalverteilung. Das Expected Improvement lässt sich damit darstellen durch [1]:

$$EI(x) = \begin{cases} \hat{\sigma}(x)[u\Phi(u) + \phi(u)] & \text{wenn } \hat{\sigma}(x) > 0 \\ 0 & \text{wenn } \hat{\sigma}(x) = 0 \end{cases}, \tag{17}$$

mit

$$u = \frac{f(x^-) - \hat{\mu}(x)}{\hat{\sigma}(x)}.$$

Kommen wir nun noch einmal zurück zu dem Beispiel, das bei der Verwendung des Mittelwerts als Infill-Kriterium verwendet wurde. Das globale Optimum der Target-Funktion konnte hier bei alleiniger Verwendung des Mittelwerts als Infill-Kriterium nicht gefunden werden. Es soll nun untersucht werden, ob es möglich ist, das globale Optimum dieser Target-Funktion mit Hilfe des Expected Improvements zu finden. Abbildung 14 zeigt ein paar Ausschnitte der Optimierung mit SMBO bei Verwendung des EI. Unter jedem Plot ist jetzt jeweils das Expected Improvement als gestrichelte Linie dargestellt. Der Vorschlag in Iteration eins ist identisch mit dem Vorschlag, den wir bei Verwendung des Mittelwerts als Infillkriterium erhalten hatten. In Iteration drei bewegen sich die Vorschläge ebenfalls in Richtung lokales Optimum. Die Varianz zwischen den beiden linken Designpunkten ist in diesem Schritt schon relativ groß. Sie reicht jedoch noch nicht aus, um den höchsten Wert des EI zu erreichen, da hier der Schätzer des Mittelwerts sehr groß ist. In Iteration sieben kann man dann aber erkennen, dass das EI einen neuen Punkt in diesem Bereich vorschlägt. Die mehrmaligen Auswertungen um das lokale Optimum haben dazu geführt, dass die Varianz hier sehr klein geworden ist und damit auch das EI, wodurch es zu einer Exploration des Raums kommt. Durch diese Erkundung des Raums ist es möglich, das globale Optimum zu finden. In Iteration 21 liegt der Algorithmus erstmals im globalen Optimum, nachdem er davor auch das zweite lokale Optimum gefunden hatte.

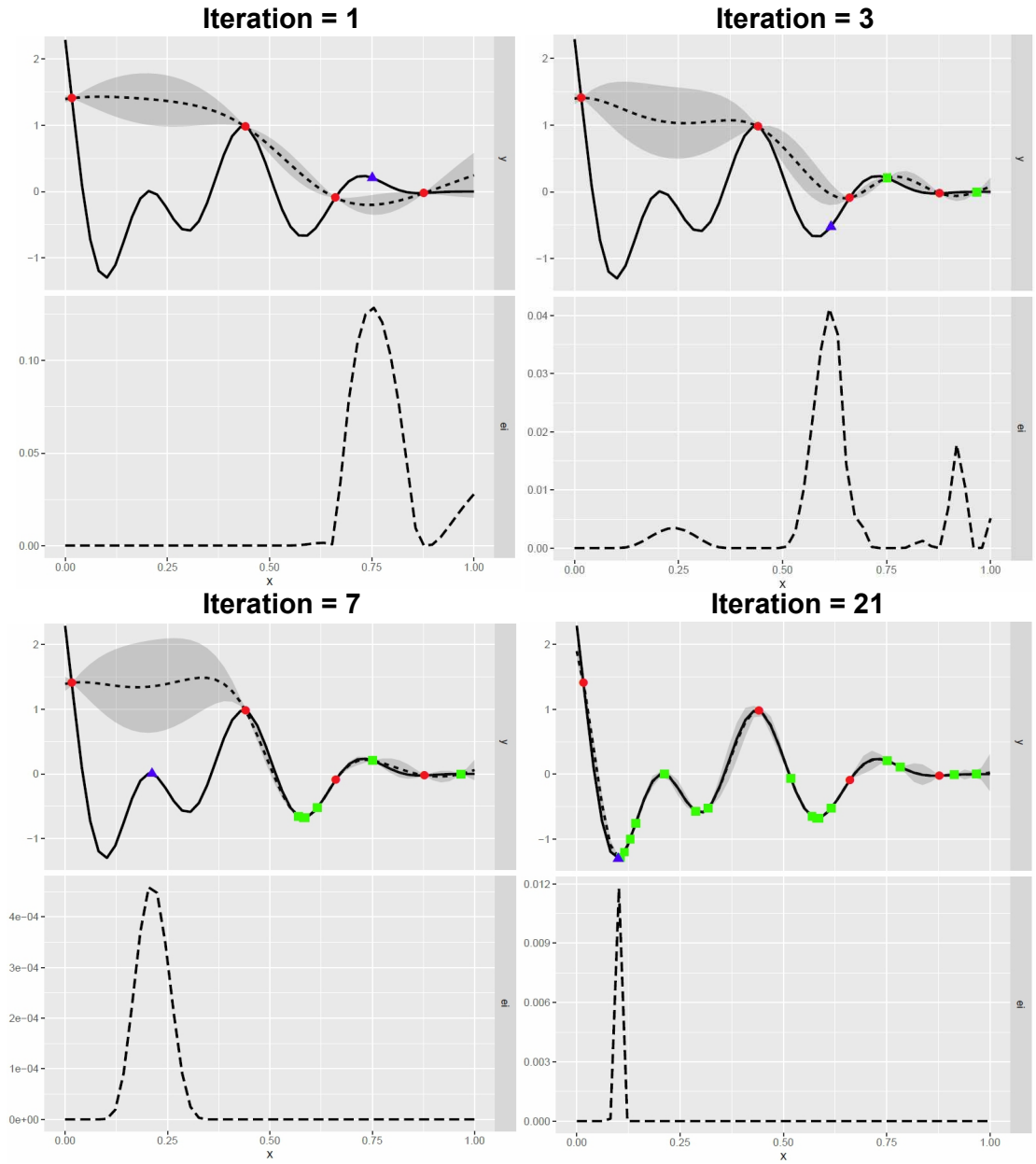

Abbildung 14 SMBO mit GPR als Surrogat-Modell und der Verwendung des EI als Infill-Kriterium. Optimiert wird eine multimodale Target-Funktion (durchgezogene Linie). Initiale Designpunkte sind als rote Punkte, Vorschläge als blaue Dreiecke und Designpunkte als grüne Quadrate dargestellt. Der geschätzte Mittelwert der GPR ist als gestrichelte Linie eingezeichnet und die Varianz als graue Schattierung. Die untere Grafik bei jeder Iteration zeigt das zugehörige EI.

Anhand dieses Beispiels wird deutlich, dass eine Erforschung des Raums entscheidend sein kann, um das globale Optimum der Target-Funktion zu finden.
Damit man den Wert finden kann, den das Infill-Kriterium als am besten für eine neue Auswertung der Target-Funktion ansieht, muss es optimiert werden. Im nun folgenden Kapitel werden einige Möglichkeiten für die Optimierung des Infill-Kriteriums genannt.

2.4 Optimierung des Infillkriteriums

Im Gegensatz zur Target-Funktion haben Infillkriterien den Vorteil, dass ihre Auswertung wenig Kosten verursacht und somit für die Optimierung viele Auswertungen durchgeführt werden können. Für die Optimierung gibt es verschiedene Ansätze. Jones schlägt vor, die Optimierung über einen branch-and-bound Algorithmus zu lösen [18]. Aufgrund der geringen Kosten einer Auswertung sind aber auch eine Grid-Search oder Latin-Hypercube-Search denkbar. Bergstra et al. schlagen vor, im diskreten Teil des Parameterraums einen Estimation-of-Distribution Algorithmus (EDA) anzuwenden und im stetigen Teil die Covariance-Matrix-Adaption Evolutions

Algorithm 1 Infill Optimization: Focus Search.

Require: black-box function $b : \mathcal{X} \to \mathbb{R}$, control parameters n_{restart}, n_{iters}, n_{points}

for $u \in \{1, ..., n_{\text{restart}}\}$ **do**
  Set $\tilde{\mathcal{X}} = \mathcal{X}$
  for $v \in \{1, ..., n_{\text{iters}}\}$ **do**
    generate random design $\mathcal{D} \subset \tilde{\mathcal{X}}$ of size n_{points}
    compute $\boldsymbol{x}^*_{u,v} = (x^*_1, ..., x^*_d) = \arg\min_{\boldsymbol{x} \in \mathcal{D}} b(\boldsymbol{x})$
    shrink $\tilde{\mathcal{X}}$ by focusing on $\mathbf{x}^*$:
    for $\tilde{\mathcal{X}}_i$ in $\tilde{\mathcal{X}}$ **do**
      if $\tilde{\mathcal{X}}_i$ numeric: $\tilde{\mathcal{X}}_i = [l_i, u_i]$ **then**
        $l_i = \max\{l_i, x^*_i - \frac{1}{4}(u_i - l_i)\}$
        $u_i = \min\{u_i, x^*_i + \frac{1}{4}(u_i - l_i)\}$
      end if
      if $\tilde{\mathcal{X}}_i$ categorical: $\tilde{\mathcal{X}}_i = \{v_{i1}, \ldots, v_{is}\}$, $s > 2$ **then**
        $\bar{x}_i$ = sample uniformly from $\tilde{\mathcal{X}}_i \backslash x^*_i$
        $\tilde{\mathcal{X}}_i = \tilde{\mathcal{X}}_i \backslash \bar{x}_i$
      end if
    end for
  end for
end for
Return $\boldsymbol{x}^* = \underset{u \in \{1,...,n_{\text{restart}}\}, v \in \{1,...,n_{\text{iters}}\}}{\arg\min} b(\boldsymbol{x}^*_{u,v})$

Abbildung 15 Pseudocode der Methode Focus Search im R Package mlrMBO [4]

Strategie (CMA-ES) [19]. Hutter et al. verfolgen den Ansatz einer Multistart Local-Search bei der tausende von Punkten des EI berechnet werden [20].
Eine andere Möglichkeit ist die im R Package mlrMBO vorgeschlagene Focus Search. Diese Optimierung des Infill-Kriteriums wird für das in dieser Arbeit vorgeschlagene alternative Infill-Kriterium verwendet. Die Focus Search geht so vor, dass sie zuerst zahlreiche Punkte in den Parameterraum legt, zum Beispiel mit Hilfe des Latin Hypercube Designs oder per Zufall. Anschließend werden die Punkte mit dem entsprechenden Infill-Kriterium ausgewertet und der beste Punkt daraus ermittelt. Als nächstes erfolgt eine Einschränkung des Parameterraums. Bei numerischen Parametern wird der Raum um die Hälfte der Spannweite eingeschränkt, und zwar zentriert um den besten Punkt. Bei diskreten Parametern schließt die Focus Search eine zufällige Kategorie aus. Innerhalb des eingeschränkten Raums werden nun aufs Neue Punkte gelegt. Der Prozess der Einschränkung des Raums und der Punktlegung wird mehrmals wiederholt. Ebenso kann auch die gesamte Focus Search häufiger durchgeführt werden. Als Optimum des Infill-Kriteriums wird schließlich der beste Wert aus allen Iterationen und Wiederholungen angesehen. In Abbildung 15 ist der Algorithmus der Focussearch durch Pseudocode dargestellt.[4]

2.5 SMBO bei der Hyperparameter-Optimierung

Vor allem im Bereich des maschinellen Lernens verwendet man Algorithmen, die die Wahl von Hyperparametern voraussetzen. Diese Parameter können nicht während des Lernprozesses geschätzt werden, sondern müssen bereits im Vorfeld festgelegt werden. Da die Wahl der Hyperparameter große Auswirkungen auf die Güte des Models haben kann, möchte man die optimalen Parameter für den Algorithmus bezüglich eines bestimmten Problems finden. Hat man einen Algorithmus A mit Hyperparametern $\lambda_1, ..., \lambda_n$ und den zugehörigen Bereichen $\Lambda_1, ..., \Lambda_n$ gegeben, so lässt sich der Hyperparameterraum durch $\Lambda = \Lambda_1 \times \cdots \times \Lambda_n$ darstellen. Für alle Hyperparametereinstellungen $\lambda \in \Lambda$ wird der zugehörige Lernalgorithmus, der die entsprechende Einstellung verwendet mit A_λ bezeichnet. Um die Güte verschiedener Konfigurationen vergleichen zu können, führt man normalerweise eine k-fache Kreuzvalidierung mit jeder einzelnen Konfiguration, die getestet werden soll, durch. Bei der Kreuzvalidierung wird zuerst der Datensatz in k zufällige Teile zerlegt und anschließend das Model auf den Daten bestehen aus $k-1$ Teilen trainiert. Diesen Datensatz bezeichnen wir als Trainingsdatensatz D_{train}. Mit dem k-ten Teil wird das Model schließlich validiert. Der k-te Teil stellt somit den Validierungsdatensatz D_{valid} dar. Für die Validierung kann ein Validierungsmaß $L(A_\lambda, D_{train}, D_{valid})$, wie zum Beispiel die Missklassifikationsrate dienen. Training und Validierung führt man

jeweils k-mal durch, damit jeder Teil einmal den Validierungsdatensatz darstellt.
Das Problem der Hyperparameter-Optimierung lässt sich durch die Optimierung der Black-Box-Funktion

$$f(\lambda) = \frac{1}{k} \sum_{i=1}^{k} L(A_\lambda, D_{train}^{(i)}, D_{valid}^{(i)}) \tag{18}$$

beschreiben [21]. Sind nur wenige Kombinationsmöglichkeiten der Hyperparameter vorhanden, reicht menschliches Expertenwissen meistens aus, um ein gutes Ergebnis zu erzielen. Sobald mehr Kombinationsmöglichkeiten ins Spiel kommen, ist die manuelle Hyperparameter-Optimierung jedoch zeitaufwändig und mühsam. Aus diesem Grund wurden automatisierte Methoden entwickelt, die zu schnelleren und besseren Ergebnissen führen sollen. Häufig eingesetzte Methoden sind Grid-Search und Random-Search, wobei zum Beispiel die Untersuchungen in [22] anhand von Neural-Networks und Deep-Belief-Networks zeigen, dass Random-Search zu besseren Ergebnissen als Grid-Search oder die manuelle Einstellung führen kann. Nicht so verbreitete Methoden für die Hyperparameter-Optimierung sind Evolutionäre Algorithmen wie die Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [23, 24], Racing Strategien wie F-Race und iterated F-Race [25, 26], Genetische Algorithmen (GGA) [27] oder auch die iterative Local-Search ParamILS [28, 29]. Vor kurzem vorgestellt wurde der Ansatz, die Hyperparameter über approximierte Gradienten mit dem HOAG Algorithmus zu optimieren [30], oder einer weitereren Gradienten basierten Methode beschrieben in [31].
Eine weitere Methode, die sehr erfolgreich in der Hyperparameter-Optimierung ist, ist die in den vorherigen Kapiteln beschriebene Methode Sequential-Model-Based-Optimization (SMBO), die auch als Bayesian-Optimization bekannt ist. Die Sichtweise ist die, dass die Hyperparameter-Optimierung als eine zu optimierende Black-Box Funktion angesehen werden kann (s. o.). Gerade bei komplexen Problemen, die pro Konfiguration eine hohe Rechenzeit beanspruchen bietet sich SMBO an, da die Methode auch mit wenigen Auswertungen der Target-Funktion ein gutes Ergebnis erzielen kann. Es hat sich bereits in vielen Fällen gezeigt, dass SMBO eine bessere Performance liefert, als Grid-Search, Random-Search oder die Konfiguration durch Expertenwissen [3].
In den Entwicklungsanfängen von SMBO standen Black-Box Funktionen im Fokus, die sich in einigen Eigenschaften von den Black-Box Funktionen der Hyperparameter-Optimierung unterscheiden. So wurde der Efficient-Global-Optimization (EGO) Algorithmus, der 1998 von Jones vorgestellt wurde, für stetige Parameter und rauschfreie Funktionen entwickelt [18]. Bei der Hyperparameter-Optimierung hat man es

jedoch mit Target-Funktionen zu tun, bei der die wahren Werte nur mit einem gewissen Rauschen bestimmt werden können. Außerdem möchte man die Optimierung häufig in kategorialen oder gemischten Parameterräumen durchführen. Ein einfaches Beispiel ist hier die gleichzeitige Bestimmung des optimalen Kernels (kategorial) und des Kostenparameters (stetig) einer Support Vector Machine (SVM). Ein weiterer Aspekt ist eine mögliche Abhängigkeitsstruktur in den Parametern. Manche Hyperparameter treten nur auf, wenn ein anderer Hyperparameter einen bestimmten Wert annimmt. Bei SVMs gibt es zum Beispiel unter anderem eine Abhängigkeit zwischen Kernel und dem Parameter γ. Die Abhängigkeitsstruktur ist in Abbildung 16 veranschaulicht. Bei Verwendung des Radial-Kernels möchte man zusätzlich den zugehörigen Parameter γ optimieren. Bei einem linearen Kernel benötigt man γ dagegen nicht, wodurch hier ein fehlender Wert entsteht. Der Kosten-

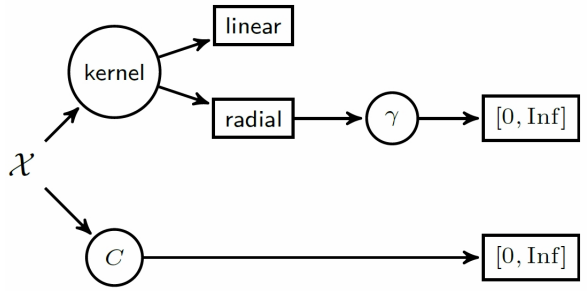

Abbildung 16 Abhängigkeitsstruktur der Hyperparameter einer SVM. Quelle: [4]

parameter c muss dagegen für beide Kernels optimiert werden, wobei es sein kann, dass der Kostenparameter je nach Kernel ein anderes Optimum einnimmt. Ein weiteres Problem bei der Hyperparameter-Optimierung ist, dass je nach konfiguriertem Wert eines Parameters, unterschiedliche Kosten entstehen können. Es kann sogar soweit kommen, dass für bestimmte Parameterkombinationen keine Werte bestimmt werden können, weil zum Beispiel die Berechnung zu viel Zeit beansprucht. Der klassische EGO Algorithmus beachtet diese Eigenschaft während den Vorschlägen neuer Auswertungen jedoch nicht.

Um die Besonderheiten bei der Hyperparameter-Optimierung berücksichtigen zu können, wurden neue SMBO Algorithmen entwickelt, bzw. der EGO Algorithmus erweitert. Ein Ansatz ist die Verwendung von RF, anstatt GPR als Surrogat-Modell. RF können mit gemischten Parameterräumen umgehen. Außerdem können die fehlenden Werte, die durch eine Abhängigkeitsstruktur entstehen, berücksichtigt werden. Hierbei wird bei kategorialen Parametern eine neue Kategorie für die fehlenden Werte gebildet. Fehlende Werte bei stetigen Parametern kodiert man dagegen mit einem Wert außerhalb des Wertebereichs des jeweiligen Parameters. Durch dieses Vorgehen kann die Abhängigkeitsstruktur in den Bäumen miteinbezogen werden [4].

Die zur Zeit bekanntesten SMBO Algorithmen für die Hyperparameter-Optimierung, sind Spearmint [32], Sequential Model-based Algorithm Configuration (SMAC) [20] und Tree Parzen Estimator (TPE) [19]. Spearmint verwendet als Surrogat-Modell Gauss-Prozesse, kann aber aufgrund einer Anpassung trotzdem sowohl mit stetigen, als auch mit kategorialen Parametern umgehen [21]. Außerdem können bei Spearmint die Kosten der Parameterwerte bei neuen Vorschlägen berücksichtigt werden. Abhängigkeitsstrukturen unterstützt dieser Algorithmus allerdings nicht. TPE konstruiert eine Dichteschätzung über gute und schlechte Initialisierungen für jeden Hyperparameter [3]. Der Algorithmus unterstützt stetige, kategoriale und abhängige Parameter [21]. Der von Hutter et al. entwickelte Algorithmus SMAC verwendet RF als Surrogat-Modell. Aufgrund dessen kann er mit gemischten Parameterräumen und Abhängigkeitsstrukturen umgehen. Hutter schlägt die Annahme von normalverteilten Werten der Target-Funktion vor ($N(\mu, \sigma^2)$). Der Mittelwert μ wird dabei durch den RF-Schätzer $\hat{\mu} = \frac{1}{B}\sum_{b=1}^{B} \hat{\mu}_b$ geschätzt, mit $\hat{\mu}_b$ als der Schätzung des b-ten Baums. Die Varianz wird aus dem Mittelwert der Varianzen der Bäume $\hat{\sigma}_b^2$ und der Varianz der $\hat{\mu}_b$ gebildet:

$$\hat{\sigma}^2 = \left(\frac{1}{B}\sum_{b=1}^{B} \hat{\sigma}_b^2\right) + \frac{1}{B}\left(\sum_{b=1}^{B} \hat{\mu}_b^2\right) - \hat{\mu}^2. \tag{19}$$

Des Weiteren schlägt Hutter eine Erweiterung für die Festlegung der Splitpunkte bei der Erstellung der Bäume vor. Wenn die Beobachtungen in zwei Mengen geteilt werden, wird der Splitpunkt in einem Knoten, bei stetigen Variablen, normalerweise in der Mitte der zwei nächstliegenden Beobachtungen gesetzt. Der Vorschlag ist nun der, die Splitpunkte in dem Intervall zwischen den beiden Beobachtungen per Zufall festzulegen. Dadurch entsteht später beim RF-Schätzer eine lineare Interpolation zwischen den beiden Datenpunkten. Lässt man nun zusätzlich das Bootstrapping bei verrauschten Funktionen weg, ähnelt das Verhalten der Varianz dem Varianzschätzer einer GPR.[33]
Die Verwendung von Random Forests als Surrogat-Modell ist ein guter Ansatz um Problemstellungen der Hyperparameter-Optimierung zu begegnen. Dennoch zeigt in manchen Fällen der Random Forest bei SMBO im Vergleich zur GPR Schwächen beim Optimierungslauf. Ein Grund dafür sind die Infill-Kriterien, die für die Anwendung bei GPR entwickelt wurden und deshalb nicht optimal auf Random Forests abgestimmt sind. Im nächsten Teil dieser Arbeit wird ein neues Infill-Kriterium für Random Forests vorgestellt.

3 Alternatives Infill-Kriterium für Random Forests

Nachdem wir im Kapitel 2 auf die wichtigsten theoretischen Grundlagen von SMBO eingegangen sind, werden wir uns nun dem neuen Infillkriterium für Random Forests zuwenden. Zuerst wird das Hauptproblem bisheriger Infill-Kriterien bei Random Forests erörtert (3.1), bevor anschließend das neue Kriterium näher beschrieben wird (3.2).

3.1 Problem bisheriger Infill-Kriterien für Random Forests

Die Verwendung von Random Forests (RF) als Surrogat-Modell hat den großen Vorteil, im Gegensatz zur klassischen GPR, dass SMBO damit auch bei kategorialen oder gemischten Parameterräumen eingesetzt werden kann. Des Weiteren ist es mit RF möglich, Abhängigkeitsstrukturen in den Parametern zu berücksichtigen.
Erinnert man sich an die Infill-Kriterien Probability-of-Improvement (Kapitel 2.3.2) und Expected Improvement (Kapitel 2.3.3), so ist dort die Standardabweichung des Schätzers mitentscheidend. Und damit kommen wir zu einem großen Nachteil bei der Verwendung von RF als Surrogat-Modell. Die Varianzschätzung bei RF ist zur Verwendung für die Vorhersage neuer Auswertungen nicht optimal geeignet, da sie häufig nicht das gewünschte Verhalten im Parameterraum zeigt. Sehen wir uns dazu ein Beispiel an. In Abbildung 17 ist dieselbe Funktion wie bei der Erklärung des EI verwendet worden (Kapitel 2.3.3), nur dass nun ein RF anstatt GPR als Surrogat-Modell verwendet wurde. Bei der ersten Iteration (linke Grafik) sieht man, dass bei der Varianzschätzung nicht die Distanz der Punkte, bzw. der x-Werte zueinander, beachtet wird. Somit kann es passieren, dass die Schätzung der Varianz auch zwischen weit entfernten Punkten nicht zunimmt. Außerdem erhöht sich die Varianz auch nicht an Funktionsgrenzen, an denen sich keine Datenpunkte befinden. Zusätzlich ist die Varianz an den Trainingspunkten nicht am kleinsten, sondern sie nimmt den selben Wert innerhalb des ganzen zugehörigen Intervalls der Schätzung an. Dieses Verhalten der Varianz kann negative Auswirkungen für den Vorschlag neuer Auswertungsstellen bei SMBO haben. Bei Verwendung der GPR mit EI hatten wir bei der Funktion in Abbildung 17 gesehen, dass der Raum weiter exploriert wird und schließlich das globale Optimum gefunden wurde (siehe Abbildung 14). Bei Verwendung des RF in Kombination mit EI ist es dagegen so, dass sich der Algorithmus im lokalen Optimum festsetzt. Selbst nach der 30. Iteration (rechte Grafik) werden neue Vorschläge immer noch im Bereich des lokalen Optimums gemacht. Erst nach 46 Iterationen (nicht in der Abbildung dargestellt) liegt ein Vorschlag das erste Mal im globalen Optimum. Natürlich ist es je nach initialem Design und Target-Funktion

nicht immer so extrem wie in diesem Beispiel gezeigt. Jedoch besteht auch in anderen Fällen das Problem, dass bei RF die Suche nicht so zielgerichtet wie bei der GPR ist und deshalb eventuell mehr Iterationen benötigt werden, um ein ähnlich gutes Ergebnis zu erzielen. Infill-Kriterien, die die Varianz des Surrogat-Modells verwenden, sind deshalb nicht optimal für den Einsatz bei RF geeignet.
Im nächsten Kapitel befassen wir uns mit einem neuen Infill-Kriterium, das versucht, bei Verwendung von RF als Surrogat-Model zu besseren Resultaten zu kommen.

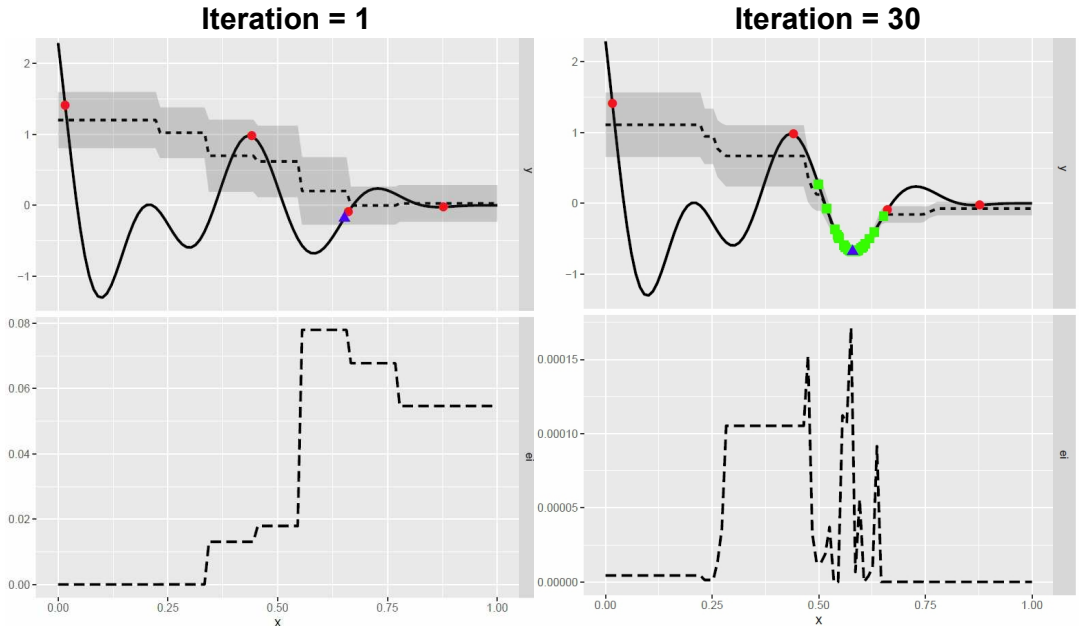

Abbildung 17 SMBO mit RF als Surrogat-Modell und EI als Infill-Kriterium. Als Target-Funktion wurde eine multimodale Funktion[3] verwendet. Die linke Grafik zeigt die 1. Iteration und die rechte Grafik die 30. Iteration. Darunter ist jeweils das EI dargestellt. Die roten Punkte zeigen die initialen Designpunkte und die grünen Punkte die Vorschläge aus den Iterationen.

3.2 Vorstellung des neuen Infill-Kriteriums für Random Forests

Bei SMBO mit RF und der Verwendung des EI als Infill-Kriterium kann es dazu kommen, dass der Optimierungslauf in einem Bereich stagniert (siehe Kapitel 3.1). Das im Folgenden vorgeschlagene neu entwickelte Infill-Kriterium soll dieses Verhalten verhindern.

[3]Die genaue Beschreibung der Target-Funktion ist dem Anhang zu entnehmen.

3.2.1 Beschreibung des Vorgehens

Die Varianzschätzung bei RF besitzt keinen guten Informationsgehalt, um neue Designpunkte vorzuschlagen, bei denen die größtmögliche Verbesserung in Richtung Optimum zu erwarten ist. Das neue Infill-Kriterium berücksichtigt aus diesem Grund nur den alleinigen RF-Schätzer, bzw. Mittelwert des Surrogat-Modells. Ohne weitere Einschränkungen würde allerdings das Problem bestehen, dass keine ausreichende Exploration des Raums stattfindet und die Gefahr groß ist, in ein lokales Optimum zu laufen (siehe Kapitel 2.3.1). Die Idee ist deshalb, einen Mindestabstand zwischen den Designpunkten und den Vorschlägen neuer Auswertungen festzulegen. Das soll verhindern, dass das Infill-Kriterium neue Punkte in unmittelbarer Nähe zu bereits ausgewerteten Punkten vorschlägt. Um die Designpunkte herum entstehen dadurch Bereiche, innerhalb derer keine neuen Punkte ausgewertet werden dürfen. Diese Bereiche bezeichnen wir im Folgenden als verbotene Räume. Zu Beginn sollen die verbotenen Räume relativ groß sein, um eine Exploration des Parameterraums zu erreichen. Eine schrittweise Verkleinerung über die Iterationen führt dazu, dass die letzten Iterationen einen relativ kleinen Mindestabstand beinhalten und dadurch eine Feinregulierung des bis dahin besten Wertes ermöglicht wird. Zu der Festlegung der Größe der verbotenen Räume zu Beginn (Startwert), sowie der Verkleinerung (Abnahmefunktionen des Mindestabstands) werden wir in Kapitel 3.2.3 kommen.
Anhand von Abbildung 18 soll die Idee des neuen Infill-Kriteriums verdeutlicht werden. Bei diesem Beispiel sollte SMBO, mit Hilfe von RF und dem neuen Infill-Kriterium, die Bird-Funktion (linke Grafik oben) optimieren. Es wurden dafür insgesamt 40 Iterationen veranschlagt. Die Farbskala gibt entweder den wahren Responsewert (linke Grafik), oder den mit dem RF geschätzten Responsewert an. Die verbotenen Räume sind hier grau schattiert und initale Designpunkte rot eingefärbt. Ein neuer Vorschlag wird durch das blaue Dreieck und neue bereits durchgeführte Auswertungen durch grüne Quadrate symbolisiert. In der 1. Iteration liegen die niedrigsten geschätzten Werte (dunkelroter Bereich) in unmittelbarer Nähe zum mittleren Designpunkt. Zu Beginn des Optimierungslaufs wäre es aber zu früh, diesen Bereich näher zu untersuchen. Man möchte als erstes den Raum weiter erkunden, um weitere Optima finden zu können. Wie man erkennen kann, wird das mit Hilfe des verbotenen Raums realisiert. Bei der Grafik mit der 23. Iteration ist zu sehen, dass der Raum in den vorherigen Iterationen abgetastet wurde und sich ausgewertete Punkte auch in der Nähe der drei tiefsten Optima der Target-Funktion befinden. Der verbotene Raum ist in der 23. Iteration schon relativ klein. Es wird aber weiterhin nicht erlaubt, Vorschläge in unmittelbare Nähe zu den Designpunkten zu legen. Zwischen Iteration 30 und 35 springen die neuen Auswertungen zwischen dem oberen

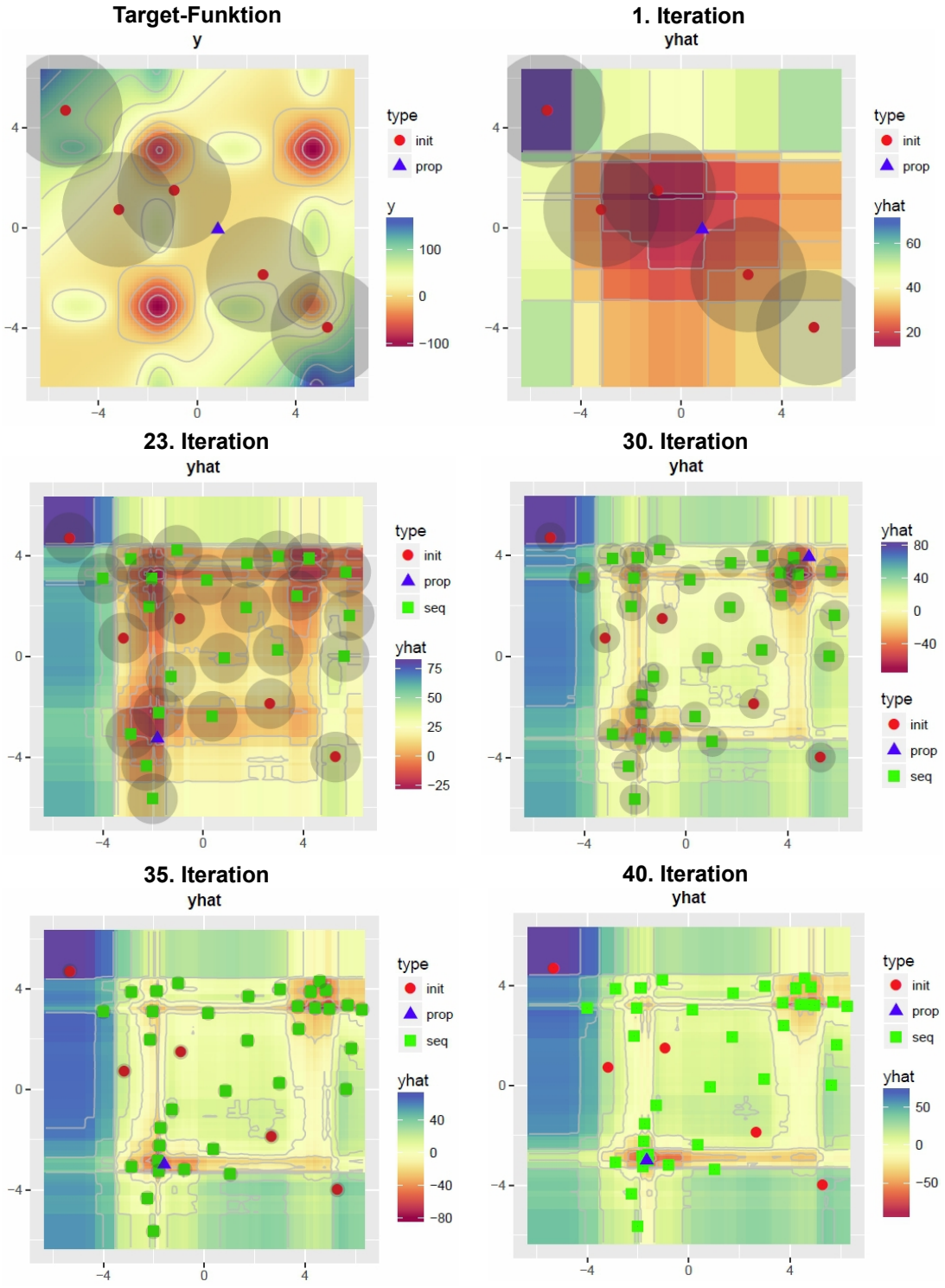

Abbildung 18 Optimierung der Bird-Funktion mit SMBO bei Verwendung des RF und des neuen Infill-Kriteriums. Initiales Design: Rote Punkte, verbotene Räume: Graue Schattierungen, neue Vorschläge: Blaue Dreiecke, neue ausgewertete Punkte: Grüne Quadrate. Die Farbskala zeigt die Responsewerte der Target-Funktion an, wobei links oben die wahren Werte der Bird-Funktion zu sehen sind und ansonsten die geschätzten Werte mit dem RF.

rechten und unteren linken Optimum hin und her, ehe dann in der 35. - 40. Iteration eine Feinjustierung des bis dahin besten Wertes durchgeführt wird.
Anhand dieses Beispiels wird deutlich, dass sowohl die Erforschung des Raums zu Beginn, als auch die Feinregulierung am Ende des Optimierungslaufs mit Hilfe des alternativen Infill-Kriteriums ermöglicht wird.
Für die Optimierung des neuen Infill-Kriteriums wird die Focus Search (Kapitel 2.5) verwendet . Die Focus Search hat den Vorteil, dass sie auch mit verbotenen Räumen ohne große Anpassungen umgehen kann. Hier muss einzig bei der zufälligen Legung der Punkte darauf geachtet werden, dass sich die Punkte nicht innerhalb eines verbotenen Raums befinden.
Für die verbotenen Räume gibt es drei verschiedene Parameter die konfiguriert werden müssen. Zum einen muss ein Distanzmaß festgelegt werden, das sich für die Bestimmung des Mindestabstands eignet. Zum anderen muss man die Größe der verbotenen Räume konfigurieren. Hierfür muss ein Startwert für den Mindestabstand zu Beginn der Iterationen ermittelt werden und außerdem die Art und Weise, wie der Mindestabstand über die Iterationen hinweg kleiner werden soll. In den folgenden Unterkapiteln werden wir uns mit den Konfigurationen beschäftigen, die auch später in den Experimenten (Kapitel 4) eingesetzt wurden.

3.2.2 Verwendete Distanzmaße

Zur Bestimmung des Mindestabstands dienten die Euklidische- und die Gower-Distanz.
Die Euklidische Distanz kann für Räume mit stetigen Parametern für die Distanzberechnung eingesetzt werden. Stellen x und y die Datenpunkte mit den Koordinaten $(x_1, ..., x_n)$ und $(y_1, ..., y_n)$ dar, ergibt sich die Euklidische Distanz aus

$$d(x, y) = ||x - y||_2 = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \quad , \tag{20}$$

wobei n gleich der Dimension des euklidischen Raums ist [34].
Für gemischte Parameterräume wurde die Gower-Distanz [35], eines der bekanntesten Ähnlichkeitsmaße für gemischte Parameterräume, verwendet. Die Ähnlichkeit zwischen zwei Punkten wird hierfür mit einem Score

$$S_{ij} = \frac{\sum_{k=1}^{\nu} s_{ijk}}{\sum_{k=1}^{\nu} \delta_{ijk}} \tag{21}$$

ermittelt. Auf s_{ijk} und δ_{ijk} gehen wir nun im weiteren Verlauf des Textes ein.
Möchte man von zwei Punkten i und j die Ähnlichkeit bestimmen, so wird zuerst

die Ähnlichkeit zwischen den beiden Punkten bezüglich jeder einzelnen Variablen $1, \ldots, \nu$, mit der Hilfe von s_{ijk}, untersucht. Nicht immer ist ein Vergleich bei einer Variablen zwischen zwei Punkten realisierbar, zum Beispiel bei fehlenden Werten. Ein Indikator δ_{ijk} gibt deshalb an, ob ein Vergleich möglich ist. Wenn ein Vergleich bei der Variablen k zwischen i und j durchgeführt werden kann, so nimmt der Indikator den Wert eins an, ansonsten den Wert null.
Der oben angegebene Gesamtscore S_{ij} ist somit der Durchschnitt der einzelnen Scores von allen möglichen Vergleichen bezüglich der Variablen. Wenn δ_{ijk} bei allen Variablen null beträgt, also die Punkte betreffend keiner Variablen verglichen werden können, ist S_{ij} nicht definiert. Falls alle Vergleiche möglich sind, ist er so groß wie die Variablenanzahl ν.
Je nach Skalenniveau gibt es Unterschiede bei der Bestimmung der Ähnlichkeit bezüglich der einzelnen Variablen. Bei einer dichotomen Variablen ist es so, dass eine Gleichheit nur angenommen wird, falls bei beiden Punkten der dichotome Wert eintritt. Der Score s_{ijk} ist in diesem Fall gleich eins und in allen anderen Fällen gleich null. Tritt bei beiden Punkten der Wert nicht ein, so wird der Vergleich nicht gezählt, weshalb δ_{ijk} in diesem Fall den Wert null annimmt. Bei kategorialen Variablen ist s_{ijk} gleich eins, wenn die Ausprägungen übereinstimmen. Ansonsten ist s_{ijk} gleich null. Bei stetigen Variablen mit den Ausprägungen $(x_1, ..., x_n)$ wird s_{ijk} durch

$$s_{ijk} = 1 - \frac{|x_{ik} - x_{jk}|}{R_k} \tag{22}$$

berechnet. R_k ist dabei die Spannweite der Variablen k und wird normalerweise über die gesamte Stichprobe berechnet. Die zweite Möglichkeit ist, die Spannweite der gesamten Population zu verwenden, falls diese zum Beispiel durch Vorwissen bekannt ist. Wenn x_i und x_j den gleichen Wert annehmen ist s_{ijk} gleich eins. Falls sich x_i und x_j maximal voneinander unterscheiden, also jeweils am gegenüberliegenden Ende der Spannweite liegen, nimmt s_{ijk} den Wert null an. Bei der Verwendung der Spannweite aus der gesamten Population kann es sein, dass der maximale Unterschied in den Elementen der Stichprobe nicht mit der Spannweite der Population übereinstimmt. Hier nimmt s_{ijk} dann normalerweise einen minimalen Wert an.
Durch diese Berechnungen der einzelnen Scorewerte liegt der gesamt Score S_{ij} im Wertebereich zwischen null und eins. Wobei null bedeutet, dass sich die Punkte maximal unterscheiden, wohingegen sie sich bei einem Wert von eins bei keiner Variablen unterscheiden. Um ein Distanzmaß mit dem Wertebereich $[0, 1]$ zu erhalten, bei dem bei einem Wert von eins die maximale Distanz zwischen zwei Punkten besteht, berechnet man $d_{gower}(x, y) = 1 - S_{ij}$ [36].

3.2.3 Größe der verbotenen Räume

Beschäftigen wir uns nun mit der Festlegung der Größe der verbotenen Räume. Die Größe ergibt sich aus dem Mindestabstand zwischen Designpunkten und dem neuen Vorschlag für eine Auswertung der Target-Funktion.
Zum einen ist hier zu entscheiden, wie groß die Räume zu Beginn der Iterationen sein sollen. Es ist also ein Startwert für den Mindestabstand festzusetzen. Der Mindestabstand sollte nicht zu groß gewählt werden, da es sonst passieren kann, dass die verbotenen Räume den ganzen Parameterraum überdecken. Dieses Ereignis kann vor allem bei größeren initialen Designs eintreffen. Die Auswahl eines zufälligen Punktes dient in diesem Fall dazu, trotzdem eine Funktionsstelle vorschlagen zu können. Zu klein sollte der Startwert natürlich auch nicht konfiguriert werden, da sonst die Gefahr besteht, dass die verbotenen Räume ihre Wirkung verfehlen und eine Erkundung des Parameterraums mit deren Hilfe nicht erzwungen werden kann. Es erscheint sinnvoll, sich für den Startwert am initialen Design zu orientieren. Bei den späteren Experimenten (Kapitel 4) wurden vier verschiedene Startwerte getestet. Und zwar wurde der Mittelwert aller möglichen Distanzen der initialen Designpunkte durch den Wert 2, 3, 4 oder 5 dividiert. Hierbei ist es natürlich wichtig, dass das initiale Design gleichmäßig in den Raum gelegt wird und sich die Punkte nicht in einem Bereich sammeln.
Neben der Größe der verbotenen Räume zu Beginn der Iterationen, muss außerdem festgelegt werden, wie die Größe von Iteration zu Iteration abnehmen soll. Es bietet sich hierfür eine lineare, parabelförmige oder negativ-parabelförmige Abnahme an. Bei allen drei Varianten sollte der Mindestabstand bei der letzten Iteration einen Wert von null annehmen.
Der Mindestabstand lässt sich für die drei Varianten der Abnahme durch folgende Funktionen in Abhängigkeit der aktuellen Iteration x darstellen. M gibt dabei den Mindestabstand an, It die Gesamtzahl der durchzuführenden Iterationen und S den Startwert des Mindestabstands.

- lineare Abnahme:

$$M(x) = -\frac{S}{(It-1)} * (It - x)$$

- parabelförmige Abnahme:

$$M(x) = \frac{S}{It^2 - 2It + 1} * (x^2 - 2It * x + 2It - 1) + S$$

- negativ parabelförmige Abnahme:

$$M(x) = -\left(S - \frac{2S * It - S * It^2}{2It - It^2 - 1}\right) * (x^2 - 2x) + \frac{2S * It - S * It^2}{2It - It^2 - 1}$$

.

Zur Veranschaulichung dient Abbildung 19, in der die drei Varianten dargestellt sind. Bei der ersten Iteration nimmt der Mindestabstand den Startwert an. Über die Iterationen folgt er dann dem entsprechenden Funktionsverlauf bis er bei der letzten Iteration einen Wert von null annimmt. Die negativ-parabelförmige Abnahme ist grün, die lineare Abnahme rot und die parabelförmige Abnahme blau eingezeichnet.

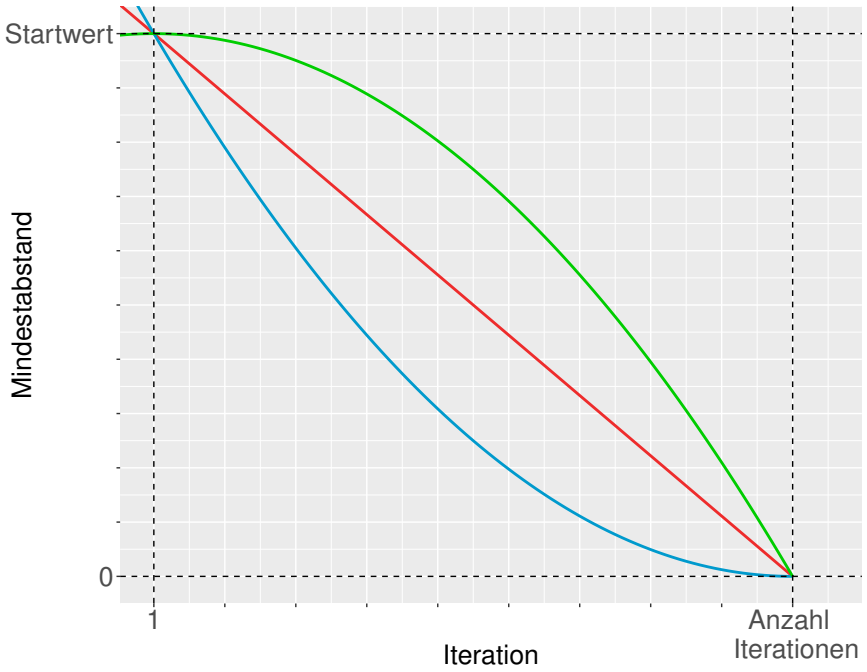

Abbildung 19 Verschiedene Funktionen für die Abnahme des Mindestabstands. Lineare (rot), parabelförmige (blau) und negativ-parabelförmige (grün) Abnahme. Die y-Achse gibt die Größe des Mindestabstands an. Die x-Achse die aktuelle Iteration eines SMBO Laufs. Bei der 1. Iteration nimmt der Mindestabstand einen festgelegten Startwert an.

Mit den in diesem Kapitel beschriebenen Konfigurationen wurde das neue Infill-Kriterium anhand von Experimenten näher untersucht. Die Ergebnisse dazu werden im nächsten Kapitel vorgestellt.

4 Experimente

Wir kommen nun zum letzten Teil dieser Arbeit, in dem auf die Experimente mit synthetischen Funktion und realen Datenbeispielen eingegangen wird. Es handelt sich dabei um eine Benchmark-Studie mit SMBO, in der das neue Infill-Kriterium mit den etablierten Infill-Kriterien verglichen wurde. Bei den realen Datenbeispielen wendete man SMBO für eine Hyperparameter-Optimierungen von einer Support-Vector-Machine an. Sowohl beim Kapitel zu den synthetischen Funktionen (Kapitel 4.3), als auch beim Kapitel zu den realen Datenbeispielen (Kapitel 4.4) gibt es ein Unterkapitel zum Versuchsaufbau der Experimente, in dem wichtige Modell Konfigurationen sowie der Aufbau der Experimente abgehandelt werden.

4.1 Rangverfahren

Bei den Experimenten wurden viele verschiedene Konfigurationen des neuen Infill-Kriteriums analysiert. Von Interesse war, welche Konfiguration die besten Ergebnisse über alle Probleme gesehen liefert. Für diese Analyse diente ein einfaches Mittelwerts-Rangverfahren, wie in [4], verwendet. Das Vorgehen war dabei wie folgt:
Die Methoden wurden auf jedem Problem mehrmals ausgeführt. Insgesamt gab es für jede Methode bei jedem Problem 20 Replikationen. Durch die Replikationen konnte man die Methoden nicht nur einmal miteinander verglichen, sondern für jede Replikation einzeln, also 20 mal. Für den Vergleich der Methoden wurden aufsteigende Ränge vergeben, wobei die Methode mit dem besten Wert den 1. Rang bekam. Als Vergleichswerte diente entweder der ermittelte Responsewert einer Methode auf den synthetischen Funktionen, oder das Vorhersagemaß bei den realen Datenbeispielen. Durch die einzelne Rangvergabe bei jeder Replikation erhielt jede Methode 20 Ränge auf jedem Problem. Aus diesen Rängen konnte man anschließend den Mittelwert für jede Methode bilden. Um nun über alle Probleme zu aggregieren, wurde für jede Methode der Mittelwert aus den Mittelwerten der Probleme bestimmt. Dieser aggregierte Wert ließ sich nun für die Ermittlung der besten Konfiguration verwenden. Mit Hilfe dieses Rangverfahrens war es möglich, die besten Konfigurationen aus den synthetischen Problemen zu ermitteln. Dadurch konnte eine übersichtlichere Darstellung der Ergebnisse realisiert werden, da nicht alle Konfigurationen aufgeführt werden mussten. Außerdem ließen sich für die realen Datenbeispiele nur die besten Konfigurationen, die sich aus den synthetischen Problemen ergaben verwenden, um einen hohen Rechenaufwand zu vermeiden.
Eine aggregierte Darstellung der Ergebnisse über alle Probleme kann man entweder über die Ränge erreichen, indem man die Ränge nicht weiter durch Mittelwertsbildung

zusammenfasst, oder aber durch eine Standardisierung, bzw. Normalisierung, der Werte. Eine Normalisierung lässt sich folgendermaßen durchführen:
Auf jedem Problem bestimmt man das Maximum und das Minimum aller Methoden über alle Replikationen hinweg. Anschließend lassen sich die normalisierten Werte durch die Formel

$$\frac{x - min}{(max - min)} \tag{23}$$

berechnen, wobei x den zu normalisierenden Wert angibt [37]. Durch die Normalisierung befinden sich die Werte bei jedem Problem im Wertebereich zwischen 0 und 1. Theoretisch kann es bei dieser Methode zu Verzerrungen durch Ausreißer kommen. Zum Beispiel würden die normalisierten Werte bei einem vorhandenen hohen Ausreißer-Wert nur im unteren Wertebereich liegen, da die Werte von der Spannweite (max - min) abhängen. Bei Herausnahme des Ausreißers würde dagegen ein anderes Bild entstehen, da sich dann die Werte gleichmäßiger über den Wertebereich verteilen. Bei dem Vergleich je Problem zwischen normalisierten und nicht normalisierten Werten zeigten sich jedoch keine Verzerrungen und ein nahezu identisches Bild. Anzumerken ist, dass bei den Ergebnissen zu den synthetischen Problemen nicht die Differenz zwischen globalem Optimum und bestem erreichten Responsewert, sondern nur die besten erreichten Responsewerte normalisiert wurden. Das Interesse lag hier nämlich nicht darin, wie gut die Methoden ein bestimmtes Problem lösen können, sondern nur um den Vergleich der Methoden untereinander. Hierfür reicht der Vergleich der Responsewerte. Dadurch konnte eine einheitliche Berechnung über synthetische und reale Probleme ermöglicht werden.

4.2 Verwendete Software

Die Experimente wurden mit der Software R[38] durchgeführt. Für die Berechnungen und Grafiken dienten die Pakete mlr[40], mlrMBO[41], DiceKriging[42], randomForest[43], extraTrees[39], e1071[44] und proxy[45]. Teil der Masterarbeit war es, das neue Infill-Kriterium in das Paket mlrMBO zu implementieren. Mit diesem Paket konnten alle SMBO Berechnungen durchgeführt werden. Die Experimente wurden auf dem Cluster des Leibnitz-Rechenzentrums mit Hilfe des Pakets batchtools[46] durchgeführt. Die synthetischen Funktionen stammen zum Teil aus dem Paket smoof[47]. Die Datensätze für die Hyperparameter Optimierung kamen dagegen aus OpenML[48].

4.3 Experimente mit synthetischen Funktionen

Bei den Benchmark Experimenten mit synthetischen Funktionen wurden sowohl Funktionen mit stetigem Parameterraum, als auch mit gemischtem Parameterraum ausgewählt. Die Auswirkung der Dimension auf die Methoden ließ sich mit den Funktionen mit stetigem Parameterraum untersuchen.
Zuerst befassen wir uns mit dem Versuchsaufbau, ehe wir zu den Ergebnissen kommen werden.

4.3.1 Versuchsaufbau synthetische Funktionen

Die Konfigurationen des neuen Infill-Kriteriums beruhen auf den Beschreibungen in Kapitel 3.2.3. Hier folgt eine kurze Zusammenfassung.
Für die Größe der verbotenen Räume musste zum einen ein Startwert für den Mindestabstand festgelegt werden. Man orientierte sich hier am initialen Design, indem man den Mittelwert aller möglichen Distanzen des initialen Designs durch 2, 3, 4 oder 5 dividierte. Zum anderen lässt sich die Größe durch die Abnahme des Mindestabstands regulieren. Hier leisteten eine lineare Abnahme (*lin*), eine parabelförmige Abnahme (*par*) und eine negativ-parabelförmige Abnahme (*neg.par*) ihren Dienst. Als Distanzmaße ließen sich die Euklidische-Distanz für die Probleme mit stetigem Parameterraum und die Gower-Distanz für die Probleme mit gemischtem Parameterraum verwenden.
Da nicht ausgeschlossen werden konnte, dass das Expected Improvement zu besseren Ergebnissen als der Mittelwert führt, wenn man hier ebenfalls den Raum für den Vorschlag neuer Punkte einschränkt, wurde sowohl der Mittelwert, als auch das Expected Improvement in Kombination mit dem verbotenen Raum untersucht.
Als Surrogat-Modelle setzte man bei allen Möglichkeiten den Random Forest und Extra-Trees ein. Da das R Paket *extraTrees* nur stetige Eingabe-Parameter behandelt, wurden die Extra-Trees bei den Experimenten auch nur im stetigen Fall angewandt.
Die Methoden konnte man somit durch den Startwert, die Abnahmefunktion, Mittelwert oder Expected Improvement und verwendetes Surrogat-Modell unterscheiden. Gegen diese Konfigurationen von SMBO mit dem neuen Infill-Kriterium, verglich man SMBO ohne verbotenem Raum. Hierbei wurden die gleichen Surrogat-Modelle, sowie Mittelwert und Expected Improvement als Infill-Kriterium verwendet, nur dass eben nun keine verbotenen Räume integriert waren. Zusätzlich diente SMBO mit GPR als Surrogat-Modell zum Vergleich. Bei Random Forest fand bei der Varianzberechnung für das Expected Improvement sowohl der Jackknife-after-Bootstrap Schätzer (*RF_EI*), als auch die einfache Form (*RF_EI_sd*, Kapitel 2.2.2) Verwendung.

Zusätzlich zu den verschiedenen SMBO Konfigurationen, wurde auch die Random Search als Optimierungsmethode auf allen Problemen angewendet.
Kommen wir nun zu den Testfunktionen. Die Folgenden synthetischen Testfunktionen mit stetigem Parameterraum wurden für die Experimente aus dem R Paket *smoof* verwendet:
$AlpineN.1$, $BohachevskyN.1$, $DeflectedCorrugatedSpring$, $Schwefel$, $DoubleSum$, $HyperEllipsoid$, $Sphere$, $SumofDifferentSquares$.
Alle Funktionen kamen als zwei-, fünf-, zehn- und 20-dimensionale Probleme zum Einsatz, womit die Methoden auf insgesamt 32 Problemen untersucht wurden.
Als synthetische Testfunktionen mit gemischtem Parameterraum wurden folgende Funktionen verwendet:
$mixedAckley$[3], $mixedQian$[3], $mixedQuadraticLiang$[3], $mixedRastrigin$[3], $mixedWeierstrassRastrigin$[3], $mixedLinear$[4], $mixedQuadratic$[4], $mixedTrigo$[4], $ComplexFunction1$[4], $ComplexFunction2Alpine$[4], $ComplexFunction4Ellipsoid$[4], $ComplexFunction4EllipsoidVariant2$[4]
Dabei gab es bei folgenden Funktionen mit gemischtem Parameterraum Abhängigkeiten unter den Parametern:
$oneDependentQuadraticLiang$[3], $oneDependentRastrigin$[3], $oneDependentTrigo$[4], $twoDependentLinear$[4], $twoDependentTrigo$[4], two- $DependentQuadratic$[4].
Auf jeder genannten Funktion wurden mit jeder Methode 20 Replikationen durchgeführt.
Beim neuen Infill-Kriterium hängt der Fortschritt innerhalb einer Iteration stark von der Gesamtzahl der eingestellten Iterationen ab, da sich die Abnahme des Mindestabstands je nach Anzahl der Iterationen anpasst, damit der Mindestabstand erst am Ende den Wert Null erhält. Aus diesem Grund kann man die Methoden nicht nach vorgegebenem Zeitbudget bewerten, sondern man muss die Methoden mit einer fest definierten Anzahl an Iterationen vergleichen. Die Anzahl der Iterationen wurde abhängig von der Dimension festgelegt. Pro Dimension erhielt eine Methode 20 Iterationen, wobei bei zwanzigdimensionalen Problemen nur insgesamt 200 Iterationen erlaubt wurden. Die Random Search bekam zusätzlich die Größe des initialen Designs von SMBO als Iterationen, um allen Methoden die gleiche Anzahl an Auswertungen zur Verfügung zu stellen. Um die Methoden außerdem vergleichbar zu machen, wurde bei allen Methoden die selbe Konfiguration des Surrogat-Modells, der Optimierung des Infill-Kriteriums und des initialen Designs verwendet. Die Konfigurationen sahen dabei folgendermaßen aus:

[3] Entnommen aus [49].
[4] Entnommen aus Arbeiten von Bernd Bischl, Institut für Statistik der LMU München.

Initiales Design:

- Anzahl Auswertungen = 5 * Dimension
- Methode = maximinLHS (Latin Hypercube Sampling)

Kriging:

- Nuggeteffekt = 1e-6
- Kovarianz = matern5-2

Random Forest:

- Varianzberechnung = Jackknife oder einfache Form
- Anzahl Bäume = 500
- min. Anzahl Beobachtungen im Knoten = 1
- Anzahl möglicher Splitvariablen = p/3

Extra-Trees:

- Anzahl Bäume = 500
- min. Anzahl Beobachtungen im Knoten = 1
- Anzahl möglicher Splitvariablen = p/3
- Anzahl zufälliger Splitpunkte pro Merkmal = 5

Optimierung Infillkriterium:

- Methode = Focus Search
- Anzahl Punkte der Focus Search = 100 * Dimension
- Anzahl Raumeinschränkungen = 5
- Restarts = 1

Für die Bezeichnung der Methoden wird im folgenden das Muster (Surrogat-Modell)_ (Infill-Kriterium, bzw. Infill-Kriterium in Kombination mit verbotenem Raum falls die zwei folgenden Informationen befüllt sind)_ (Abnahmefunktion)_ (Teiler beim Startwert) verwendet. Zum Beispiel besteht die Methode *RF_M_lin_3* aus dem Surrogat-Modell Random Forest und aus dem neuen Infill-Kriterium mit Mittelwert und linearer Abnahme des Mindestabstands, sowie aus dem Startwert des mittleren Abstands des initialen Designs dividiert durch drei.

4.3.2 Ergebnisse synthetische Funktionen

Als erstes wird auf die Auswirkung der Konfiguration des verbotenen Raums eingegangen. In Abbildung 20 sind Ergebnisse verschiedener Konfigurationen des neuen Infill-Kriteriums auf zweidimensionalen Problemen dargestellt. Für die Aggregation wurde die in Kapitel 4.1 beschriebene Normalisierung angewandt. Die Methoden verwendeten alle den Mittelwert in Kombination mit dem verbotenen Raum. Die Ergebnisse der Methode in Kombination mit EI liefern keine weiteren Erkenntnisse und sind hier aus darstellungstechnischen Gründen nicht aufgeführt.

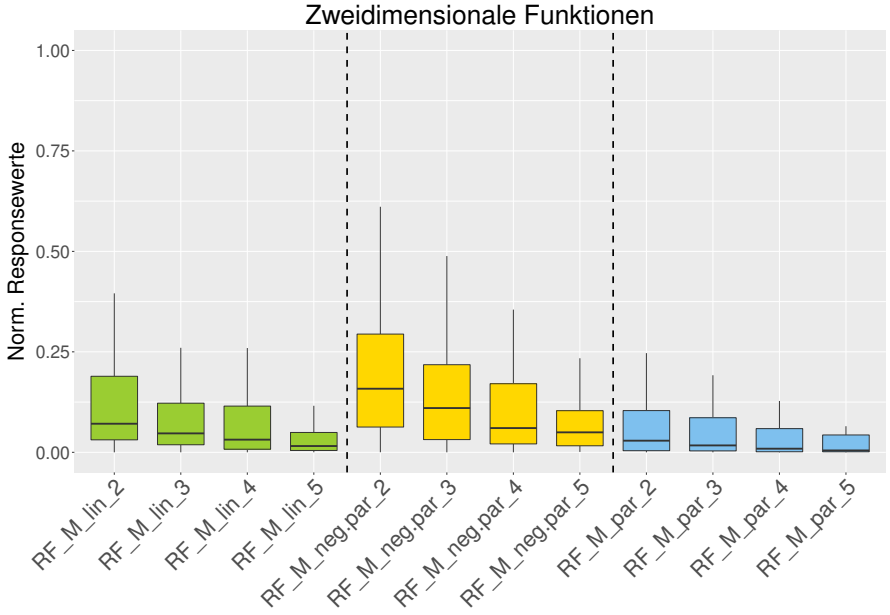

Abbildung 20 Vergleich verschiedener Konfigurationen des neuen Infill-Kriteriums bei zweidimensionalen Testfunktionen. Responsewerte wurden normalisiert. RF = Random Forest, M = Mittelwert, lin = lineare Abnahme des Mindestabstands, (neg.)par = (negativ-)parabelförmige Abnahme des Mindestabstands. Zahlen geben den Startwert des Mindestabstands an.

Die Infill-Kriterien sind von links nach rechts mit sinkendem Startwert angeordnet, wobei immer jeweils vier aneinander liegende Boxplots die selbe Abnahmefunktion verwenden. Was man auf der Abbildung erkennen kann ist, dass es einen leichten Trend bezüglich des Startwerts gibt. Methoden mit kleinerem Startwert liefern leicht bessere Resultate, als Methoden mit höheren Startwerten. Bei der Konfiguration der Abnahme des Mindestabstands liegt die parabelförmige Abnahme (*par*) minimal vorne, gefolgt von der linearen Abnahme (*lin*) und der negativ-parabelförmigen (*neg.par*). Die Unterschiede lassen sich jedoch als nur sehr gering bezeichnen. Schaut man sich die Ergebnisse der verschiedenen Konfigurationen im zwanzigdimensionalen Bereich an (Abbildung 21), so lässt sich hier kein Unterschied mehr feststellen. Alle Methoden liegen auf dem gleichen Niveau. Weder zwischen den verschiedenen Startwerten, noch zwischen den Abnahmefunktionen sind Differenzen erkennbar.

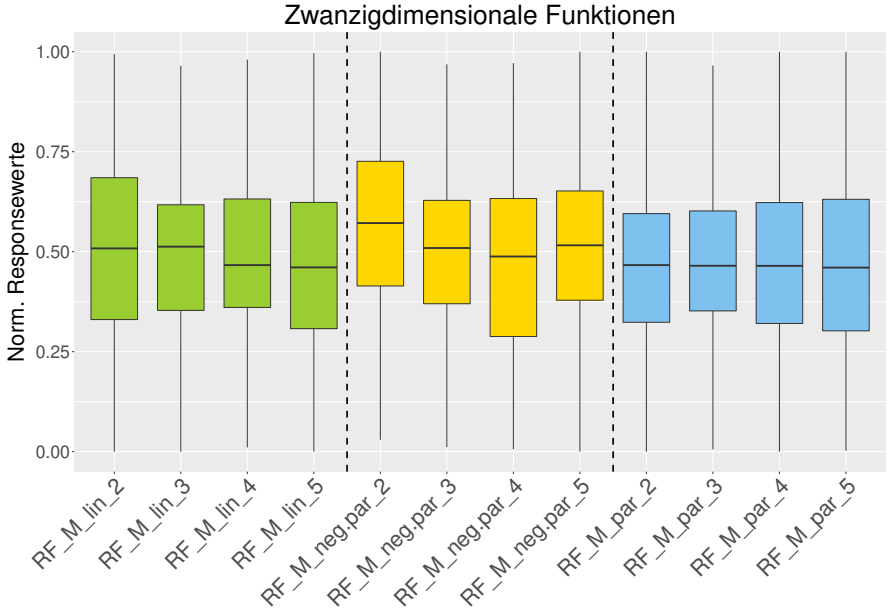

Abbildung 21 Vergleich verschiedener Konfigurationen des neuen Infill-Kriteriums bei zwanzigdimensionalen Testfunktionen. Responsewerte wurden normalisiert. RF = Random Forest, M = Mittelwert, lin = lineare Abnahme des Mindestabstands, (neg.)par = (negativ-)parabelförmige Abnahme des Mindestabstands. Zahlen geben den Startwert des Mindestabstands an.

Um die besten Konfigurationen aus allen Dimensionen zu bestimmen, wurde das in Kapitel 4.1 genannte Rangverfahren verwendet. Nur die Konfigurationen mit den niedrigsten Durchschnittsrängen wurden später gegen die Infill-Kriterien ohne verbotenen Raum getestet. In Tabelle 3 sehen wir die Ergebnisse der Ränge aufgelistet. Die Ränge wurden für stetige und gemischte Parameterräume getrennt berechnet, da nicht ausgeschlossen werden konnte, dass sich die Methoden bei gemischtem

Parameterraum anders verhalten. Bei der späteren Darstellung konnten dadurch sowohl die besten Konfigurationen für stetige, als auch für gemischte Parameterräume veranschaulicht und verglichen werden.
Wie es sich schon aus den eben behandelten Grafiken vermuten ließ, zeigen die Ränge in Tabelle 3, dass unter allen Konfigurationen offensichtlich die Verwendung des Mittelwerts als Infill-Kiterium in Kombination mit einer parabelförmigen Abnahme des Mindestabstands die besten Resultate liefert. Sowohl im stetigen, als auch im gemischten Parameterraum hat die Methode *RF_M_par_4* den niedrigsten Durchschnittsrang. Die Methoden unterscheiden sich bei Verwendung eines anderen Startwertes aber ansonsten gleicher Konfiguration allerdings nur sehr gering. Auch scheint das Skalenniveau des Parameterraums eine untergeordnete Rolle zu spielen. Für die weiteren Analysen werden aufgrund der Durchschnittsränge die Konfigura-

Tabelle 1 Übersicht des durchschnittlichen Rangs der einzelnen Methoden. Die Ränge wurden für Funktionen mit stetigen und gemischten Parameterräumen getrennt berechnet und sind hier von gut nach schlecht aufgelistet.

Stetiger Parameterraum		**Gemischter Parameterraum**	
Methode	**Rang**	**Methode**	**Rang**
RF_ M_ par_ 4	6.667	RF_ M_ par_ 4	8.037
RF_ M_ par_ 5	6.704	RF_ M_ par_ 3	8.519
RF_ M_ par_ 2	6.792	RF_ M_ par_ 5	8.720
RF_ M_ par_ 3	6.992	RF_ M_ par_ 2	9.402
RF_ M_ lin_ 5	7.144	RF_ M_ lin_ 5	10.981
RF_ M_ lin_ 4	7.782	RF_ M_ neg.par_ 5	11.631
RF_ M_ lin_ 3	8.137	RF_ EI_ par_ 5	11.954
RF_ M_ neg.par_ 5	8.999	RF_ M_ lin_ 3	12.153
RF_ M_ neg.par_ 4	9.347	RF_ M_ lin_ 4	12.242
RF_ M_ lin_ 2	9.704	RF_ M_ neg.par_ 3	12.284
RF_ M_ neg.par_ 3	10.051	RF_ M_ neg.par_ 4	12.527
RF_ M_ neg.par_ 2	11.427	RF_ EI_ par_ 4	12.550
RF_ EI_ lin_ 4	15.655	RF_ M_ lin_ 2	12.644
RF_ EI_ neg.par_ 5	15.685	RF_ EI_ par_ 3	13.080
RF_ EI_ par_ 3	15.798	RF_ M_ neg.par_ 2	13.196
RF_ EI_ par_ 4	16.066	RF_ EI_ par_ 2	13.511
RF_ EI_ par_ 5	16.270	RF_ EI_ lin_ 5	13.698
RF_ EI_ lin_ 5	16.307	RF_ EI_ lin_ 4	13.952
RF_ EI_ neg.par_ 4	16.528	RF_ EI_ neg.par_ 4	14.174
RF_ EI_ par_ 2	16.743	RF_ EI_ lin_ 3	14.529
RF_ EI_ lin_ 3	16.805	RF_ EI_ neg.par_ 5	14.583
RF_ EI_ neg.par_ 3	17.765	RF_ EI_ neg.par_ 3	14.873
RF_ EI_ lin_ 2	17.822	RF_ EI_ lin_ 2	15.370
RF_ EI_ neg.par_ 2	18.808	RF_ EI_ neg.par_ 2	15.391

tionen $RF\_M\_par\_4$ (niedrigster Durchschnittsrang), $RF\_M\_lin\_5$ (Konf. mit der zweitbesten Abnahmefunktion), $RF\_EI\_par\_5$ (beste Konf. mit EI als Infill-Kriterium bei gemischtem Parameterraum), $RF\_EI\_lin\_4$ (beste Konf. mit EI als Infill-Kriterium bei stetigem Parameterraum) ausgewählt. Für Extra-Trees als Surrogat-Modell wurde ebenfalls das Rangverfahren angewendet und zwar separat von den Random Forests. Ausgewählt wurden die Konfigurationen $ET\_M\_par\_3$ (niedrigster Durchschnittsrang) und $ET\_M\_lin\_4$ (Konf. mit der zweitbesten Abnahmefunktion). Die Tabelle zu allen Rängen befindet sich in Anhang A.1.
Kommen wir nun als erstes zu den Ergebnissen bei den synthetischen Funktionen mit stetigem Parameterraum. Von Interesse war, wie sich die Methoden über die verschiedenen Dimensionen der Probleme verhalten. Um eine Aggregation der Probleme bei jeder Dimension zu erreichen, wurden die Werte bei jedem Problem wie in Kapitel 4.1 beschrieben normalisiert. Die Grafiken zu den einzelnen Funktionen befinden sich in Anhang A.3.
In Abbildung 22 sind die normalisierten Werte der Probleme aufgeteilt in die verschiedenen Dimensionen dargestellt. Ein Boxplot enthält dabei von allen Problemen mit der entsprechenden Dimension die normalisierten Werte aller Replikationen der jeweiligen Methode. Zuerst wird bei dieser Abbildung der Gesamtvergleich der Methoden erläutert, bevor wir zu einem direkten Vergleich zwischen neuem und bisherigen Infill-Kriterien kommen.
Insgesamt gesehen schneidet wie erwartet die Random Search (RS) in allen Dimensionen am schlechtesten ab, wobei sie im zweidimensionalen Fall mit SMBO und Extra-Trees (ET_M) in etwa auf einer Höhe liegt.
GPR (GPR_EI, GPR_M) liefert bis auf den zweidimensionalen Bereich klar die besten Resultate, wobei die Verwendung des Mittelwerts als Infill-Kriterium aufgrund der geringeren Streuung etwas besser als die Verwendung des Expected Improvements ist. Auch bei Random Forests ist die Verwendung des Mittelwerts im Vergleich zum EI besser, unabhängig davon, ob die neue oder alte Version des Infill-Kriteriums eingesetzt wird. Eine Ausnahme bildet allerdings der zweidimensionale Bereich. Hier wird bei Verwendung des Mittelwerts ohne verbotenem Raum (RF_M) kein besseres Ergebnis erzielt, als bei Verwendung von EI ohne verbotenem Raum (RF_EI, RF_EI_sd). EI mit der einfachen Form der Varianzberechnung (RF_EI_sd) liefert hier im Vergleich sogar etwas bessere Ergebnisse als die Varianzberechnung mit Jackknife (RF_EI).
Bei der Analyse des Unterschieds zwischen Random Forests und Extra-Trees zeigt sich, dass es im zehndimensionalen Bereich keinen nennenswerten Unterschied gibt. Ebenso im zweidimensionalen Bereich bei der neuen Version des Infill-Kriteriums

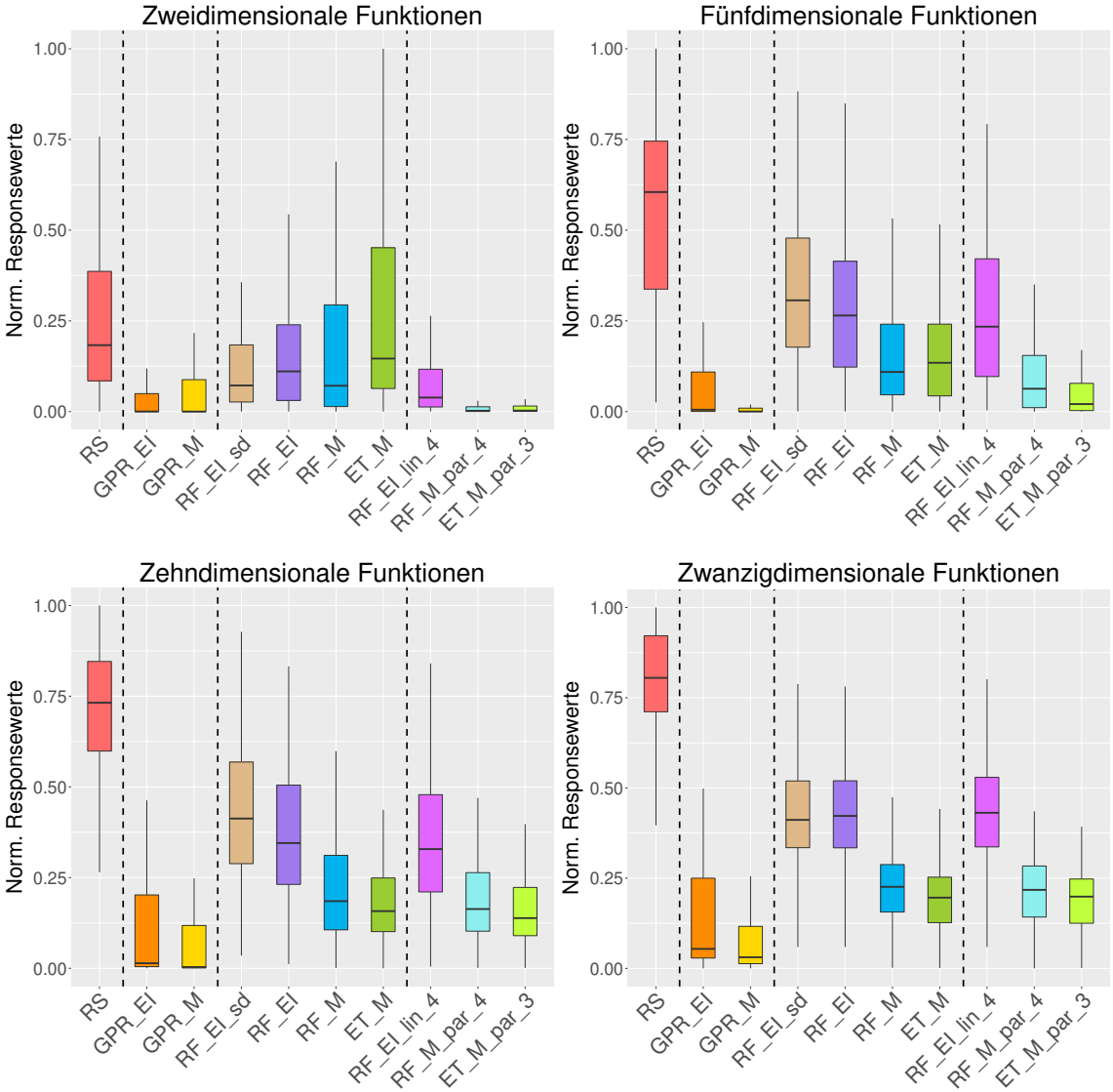

Abbildung 22 Vergleich verschiedener SMBO Konfigurationen. RS = Random Search, GPR = Gauss-Prozess-Regression, RF = Random Forest, EI = Expected Improvement, M = Mittelwert, lin = lineare Abnahme des Mindestabstands, par = parabelförmige Abnahme des Mindestabstands. Zahlen geben den Startwert des Mindestabstands an. Getestet wurde auf Zwei-, Fünf-, Zehn- und Zwanzigdimensionalen Synthetischen Funktionen mit stetigem Parameterraum, wobei hier die besten erreichten Responsewerte normalisiert wurden.

(*RF_M_par_4*, *ET_M_par_3*). Ohne verbotenem Raum (*ET_M*) schneidet Extra-Trees in dieser Dimension allerdings schlechter ab als der Random Forest (*RF_M*). Ersterer ermittelt sogar ähnliche Werte wie die Random Search (*RS*), womit es im zweidimensionalen Fall keinen Unterschied machen würde, ob man die Punkte die ausgewertet werden sollen zufällig auswählt, oder über SMBO mit Extra-Trees vorschlagen lässt. Im fünfdimensionalen Parameterraum liefert die Verwendung des Extra-Trees als Surrogat-Modell von allen Konfigurationen des neuen Infill-Kriteriums die besten Ergebnisse. Bei den Infill-Kriterien ohne verbotenem Raum liegen Random Forest mit Mittelwert (*RF_M*) und Extra Trees (*ET_M*) dagegen auf einem Niveau. Im 20-dimensionalen Fall zeigt sich bei dem Vergleich der beiden Surrogat-Modelle, dass Extra-Trees minimal besser abschneidet, als Random Forest und zwar sowohl beim neuen, als auch beim alten Infill-Kriterium. Nähere Untersuchungen haben gezeigt, dass die Verwendung von Extra-Trees bei manchen Problemen im zwanzigdimensionalen Bereich ein deutlich besseres Ergebnis liefert als die Verwendung des Random Forests. Die Grafiken zu den einzelnen Problemen befinden sich in Anhang A.3.
Der Vergleich zwischen neuem und altem Infill-Kriterium legt dar, dass die Verwendung des neuen Infill-Kriteriums mit Mittelwert (*RF_M_par_4*, *ET_M_par_3*) im zweidimensionalen Bereich besser als alle anderen Methoden abschneidet. Sogar GPR präsentiert sich hier aufgrund der größeren Streuung etwas schlechter. Im fünfdimensionalen Fall lässt sich jedoch feststellen, dass das neue Infill-Kriterium nur noch zu geringfügig besseren Ergebnissen führt, als die Verwendung des Infill-Kriteriums ohne verbotenem Raum. Außerdem ist hier GPR deutlich besser. Bei allen Dimensionen liefert die Kombination Mittelwert mit verbotenem Raum (*RF_M_par_4*, *ET_M_par_3*) bessere Ergebnisse, als die Verwendung von Expected Improvement mit verbotenem Raum (*RF_EI_lin_4*).
Im höherdimensionalen Bereich zeichnet sich ab, dass es hier keinen Unterschied zwischen altem und neuem Infill-Kiterium gibt. Es spielt in diesem Fall also keine Rolle, ob der verbotene Raum zusätzlich in das Infill-Kriterium integriert wird.
Als nächstes kommen wir zu den Ergebnissen der Probleme mit gemischtem Parameterraum. Es wurde für die Ergebnisdarstellung über die zwei- und dreidimensionalen Funktionen aggregiert, da sich das Verhalten der Methoden zueinander bezüglich dieser Dimensionen nicht groß unterschied. Eine Ausnahme bildeten zwei komplexe Funktionen deren Ergebnis deutlich von den anderen abwich. Aus diesem Grund wurden diese beiden Funktionen, sowie die vierdimensionale Funktion gesondert dargestellt. In Abbildung 23 sehen wir die einzelnen Boxplot zu den normalisierten Werten der zwei- und dreidimensionalen Funktionen. Es wurde hier die selbe Norma-

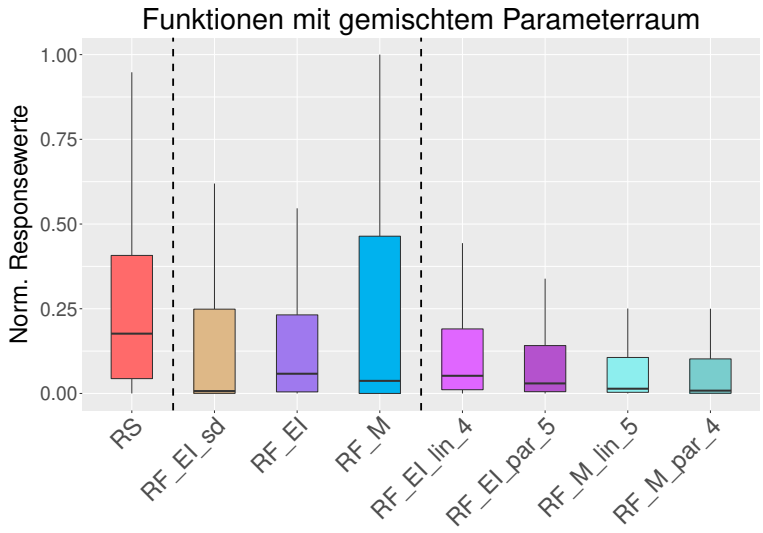

Abbildung 23 Vergleich verschiedener SMBO Konfigurationen bei zwei- und dreidimensionalen Funktionen mit gemischtem Parameterraum. RS = Random Search, RF = Random Forest, EI = Expected Improvement, M = Mittelwert, lin = lineare Abnahme des Mindestabstands, par = parabelförmige Abnahme des Mindestabstands. Zahlen geben den Startwert des Mindestabstands an. Getestet wurde auf Zwei- und dreidimensionalen Synthetischen Funktionen mit gemischtem Parameterraum, wobei hier die besten erreichten Responsewerte normalisiert wurden.

lisierung wie bei den Funktionen mit stetigem Parameterraum verwendet. SMBO mit GPR fällt bei diesen Problemen weg, da die klassische GPR nur im stetigen Parameterraum angewendet werden kann.

Bei den SMBO Methoden ohne verbotenem Raum schneidet der Random Forest mit der einfachen Form der Varianzberechnung (RF_EI_sd) am besten ab. Allerdings ist der Unterschied zwischen der Varianzberechnung mit Jackknife (RF_EI) nur sehr gering. Bei SMBO mit verbotenem Raum gewinnt die Konfiguration mit Mittelwert ($RF\_M\_lin\_5$, $RF\_M\_par\_4$) vor Expected Improvement ($RF\_EI\_lin\_4$, $RF\_EI\_par\_5$). Der Vergleich zwischen alten und neuen Infill-Kriterien zeigt, dass das neue Infill-Kriterium mit Mittelwert ($RF\_M\_lin\_5$, $RF\_M\_par\_4$) besser abschneidet. Das neue Infill-Krterium mit Expected Improvement ($RF\_EI\_lin\_4$, $RF\_EI\_par\_5$) liefert ähnliche Ergebnisse wie Expected Improvement ohne verbotenem Raum (RF_EI). Schaut man sich die Konfigurationen des neuen Infill-Kriteriums an, so fällt auf, dass nur die Wahl zwischen EI oder Mittelwert einen Unterschied bei den Ergebnissen ergibt (allerdings ist dieser gering). Die Konfiguration des Startwerts und der Abnahmefunktion scheint nebensächlich zu sein.

Zwei Funktionen zeigten deutlich andere Ergebnisse als die in Abbildung 23 aggregierten Probleme. Und zwar handelt es sich dabei um komplexere Funktionen im

dreidimensionalen Bereich. In Abbildung 24 ist für alle Replikationen jeweils der beste Responsewert, den die Methode erreichen konnte, dargestellt. Es zeigt sich hier, dass bei diesen Problemen Methoden die den Mittelwert im Infill-Kriterium verwenden, viel schlechter abschneiden, als die übrigen Methoden. Sogar die Random Search liefert bessere Ergebnisse. Bei der *ComplexFunction*1 (linke Grafik), liefert das neue Infill-Kriterium mit EI die besten Ergebnisse, da hier die Streuung nicht so weit ausgeprägt ist, wie bei den Infill-Kriterien ohne verbotenem Raum. Bei der *ComplexFunction*2*Alpine* (rechte Grafik), liegen die SMBO Methoden mit Expected Improvement bei Verwendung mit und ohne verbotenem Raum auf einer Höhe.
Bei diesen beiden Problemen wäre also im Gegensatz zu den anderen Problemen die Verwendung des neuen Infill-Kriteriums mit Mittelwert nicht so gut für die Lösung der Probleme geeignet.

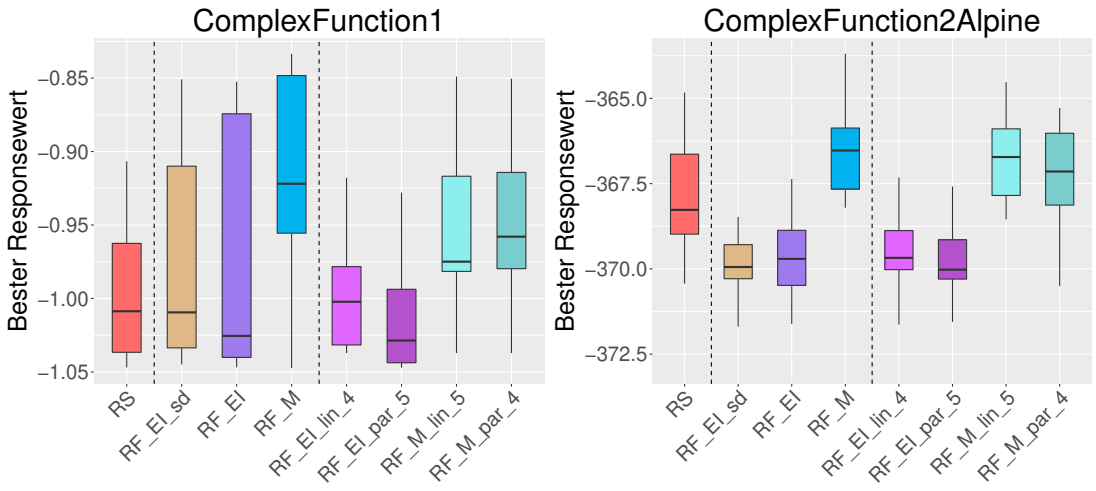

Abbildung 24 Ausreißer-Funktionen. RS = Random Search, RF = Random Forest, EI = Expected Improvement, M = Mittelwert, lin = lineare Abnahme des Mindestabstands, par = parabelförmige Abnahme des Mindestabstands. Zahlen geben den Startwert des Mindestabstands an. Funktionen mit gemischtem Parameterraum, wobei die y-Achse die besten erreichten Responsewerte zeigt.

Als nächstes schauen wir uns nun noch die Ergebnisse zu der vierdimensionalen Funktion *ComplexFunction*4*Ellipsoid* an. Die Ergebnisse sehen ähnlich zu denen im zwei- oder dreidimensionalen Fall aus. Wobei hier das neue Infill-Kriterium bei Verwendung des Mittelwerts sogar noch deutlicher vor allen anderen Methoden liegt. Das neue Infillkriterium mit Expected Improvement ist leicht schlechter als *RF_EI* und *RF_M*. Auffällig ist, dass hier Expected Improvement der einfachen Varianzberechnung deutlich schlechter ist, als bei Berechnung mit Jackknife.

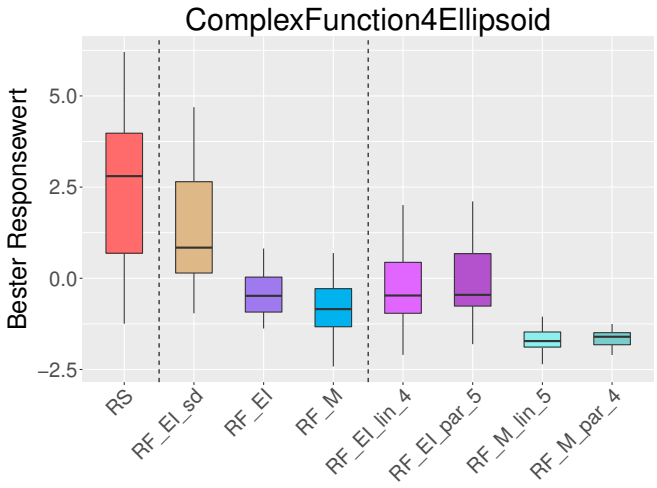

Abbildung 25 Vierdimensionale Funktion mit gemischtem Parameterraum, wobei die y-Achse die besten erreichten Responsewerte zeigt. RS = Random Search, RF = Random Forest, EI = Expected Improvement, M = Mittelwert, lin = lineare Abnahme des Mindestabstands, par = parabelförmige Abnahme des Mindestabstands. Zahlen geben den Startwert des Mindestabstands an.

4.3.3 Zusammenfassung synthetische Funktionen

Bei den synthetischen Funktionen mit stetigem Parameterraum zeigte sich, dass das neue Infill-Kriterium im zweidimensionalen Bereich gegenüber allen anderen Methoden am besten abschnitt. Die gute Wirkung des verbotenen Raums nahm jedoch bei zunehmender Dimension ab. Im höherdimensionalen Bereich ergab sich bei Verwendung der verbotenen Räume kein Unterschied mehr zu den alten Infill-Kriterien. Interessant ist, dass es aber in keinem Fall schlechter war, das Infill-Kriterium mit verbotenem Raum zu verwenden. Bei der Überprüfung, ob die verbotenen Räume bei höherer Dimension noch genügend Raum zu Beginn der Iterationen einnehmen zeigte sich, dass das Verhältnis von verbotenem und nicht verbotenem Raum ähnlich zu dem im zweidimensionalen Fall ist. Eine mögliche Ursache könnte aber sein, dass eventuell die Abnahme des Mindestabstands in höheren Dimensionen nicht so gut geeignet ist. Hier wäre es interessant weitere Untersuchungen durchzuführen. Allerdings war auch zu sehen, dass die Konfiguration des Startwerts und die Abnahmefunktion des Mindestabstands gerade im höherdimensionalen Bereich keine Rolle zu spielen scheint. Jedoch waren die Abnahmefunktionen hierfür eventuell zu ähnlich zueinander.
Aufgrund der Ergebnisse wäre der Einsatz des neuen Infill-Kriteriums bis zum fünfdimensionalen Fall geeignet. Vor allem, da es bei gemischtem Parameterraum sogar im vierdimensionalen Bereich noch einen deutlichen Unterschied zwischen neuem und

altem Infill-Kriterium gab. Ab dem zehndimensionalen Fall ist ein sinnvoller Einsatz jedoch fraglich.
Da die Verwendung des verbotenen Raums im zehn- und zwanzigdimensionalen Raum keine Auswirkung mehr zeigte, gibt es hier auch keinen Unterschied zwischen den Konfigurationen des Startwerts und der Abnahmefunktion. Im zweidimensionalen Bereich kristallisierte sich jedoch eine parabelförmige Abnahme als beste Variante heraus. Somit kommt die Methode zu besseren Resultaten, wenn der Raum zu Beginn kürzer exploriert wird und für das Finetuning mehr Iterationen verwendet werden können.
Allgemein ist es etwas überraschend, dass bei fast allen Funktionen, auch ohne verbotenem Raum, die Verwendung des Expected Improvements zu schlechteren Ergebnissen führte, als die Verwendung des Mittelwerts. Vermuten würde man, dass sich SMBO bei Verwendung des Mittelwerts als Infill-Kriterium in einem Bereich festsetzt und nicht ausreichend exploriert, um das Optimum zu finden. Es kann sein, dass die schlechte Varianzschätzung bei Random Forest im Expected Improvement dazu geführt hat, dass neue Punkte an ungünstigen Stellen vorgeschlagen wurden und deshalb EI schlechter gegenüber dem Mittelwert abschnitt. Allerdings zeigte sich auch GPR mit EI bei diesen Testfunktionen insgesamt gesehen leicht schlechter, als bei Verwendung mit Mittelwert.
Nachdem das neue Infill-Kriterium nun auf den synthetischen Funktionen getestet wurde, ist es interessant, wie gut sich das Kriterium auf realen Problemen zeigt.

4.4 Experimente mit realen Datenbeispielen

Um das Infill-Kriterium unter realen Bedingungen zu testen, wurde eine Hyperparameter-Optimierung mit SMBO durchgeführt. Dabei wurden verschiedene Dimensionen analysiert, da es bei den synthetischen Funktionen einen Unterschied bei den Ergebnissen zwischen den Dimensionen gab. Die Unterkapitel gliedern sich in den Versuchsaufbau (Kapitel 4.4.1), die Auswahl der Datensätze (Kapitel 4.4.2), die Darstellung der Ergebnisse (Kapitel 4.4.3), sowie einer kurzen Zusammenfassung (Kapitel 4.4.4).

4.4.1 Versuchsaufbau reale Datenbeispiele

Um sich bei den realen Datenbeispielen den hohen Rechenaufwand bei Durchführung aller Konfigurationen des neuen Infill-Kriteriums zu ersparen, verwendete man nur die Konfigurationen, die bei den synthetischen Funktionen über das Rangverfahren ausgewählt wurden (siehe Kapitel 4.3.2).

Für die Tests unter realen Bedingungen eignete sich eine Hyperparameter-Optimierung von Support Vector Machines (SVM) auf realen Datensätzen. Für die Berechnungen der SVM fungierte das R Paket $e1071$. Die Bezeichnungen der Parameter wurden hieraus übernommen.
Im ersten Schritt sollten die Hyperparameter $Cost$ und γ einer SVM mit Radial-Gauss-Kernel optimiert werden. Gesucht wurde bei beiden Parametern im Wertebereich -20 bis 20, wobei die Werte für die Berechnung mit der SVM mit 2^x transformiert wurden.
In einem zweiten Schritt sollte neben dem Kostenparameter $Cost$ auch der Kernel mit zugehörigen Parametern optimiert werden. Die Kernel lassen sich mit u und v als zwei Beobachtungen darstellen durch

- linear: $u * v$
- polynomial: $(gamma * u * v + coef0)^{degree}$
- radial basis: $exp(-gamma * |u - v|^2)$
- sigmoid: $tanh(gamma * u * v + coef0)$

Tabelle 2 gibt einen Überblick der zu optimierenden Hyperparameter. Das X gibt in der Tabelle jeweils an, wenn der Kernel den entsprechenden Parameter verwendet. Da nicht jeder Kernel alle Parameter benötigt, existieren Abhängigkeiten zwischen Kernel und den zugehörigen Parametern.

Tabelle 2 Übersicht der zu optimierenden Hyperparameter der SVM

Kernel	Cost	Gamma	Coef0	Degree
Gauss	X	X	-	-
Linear	X	-	-	-
Sigmoid	X	X	X	-
Polynomial	X	X	X	X

Der Wertebereich in dem die Optimierung durchgeführt wurde, liegt beim Kostenparameter zwischen -20 und 20 (Trafo 2^x), bei γ zwischen -20 und 15 (Trafo 2^x), bei $Coef0$ zwischen -50 und 50 und bei $Degree$ zwischen 1 und 5. Bei γ wurde als obere Grenze 15 verwendet, da es mit dem polynomial Kernel bei höherem γ Wert zu sehr vielen Abbrüchen der SVM Berechnung in R kam.
Bei der Optimierung des $Cost$ und $Gamma$ Parameters im ersten Schritt handelt es sich um ein zweidimensionales Problem, im zweiten Schritt bei zusätzlicher Optimierung des Kernels mit den zugehörigen Parametern, um ein fünfdimensionales Problem. Da bei der ersten Hyperparameter-Optimierung nur stetige Parameter optimiert

werden mussten, wurden hier auch Kriging und Extra-Trees als Surrogat-Modell verwendet.
Der Raum in dem optimiert werden soll, wird durch die Wertebereiche der Hyperparameter erzeugt. Man möchte nun die Kombination der Hyperparameter finden, mit der die SVM einen bestimmten Datensatz am besten klassifizieren kann. Dabei dient ein Vorhersagemaß dazu, die Güte der SVM bestimmen zu können. Bei den Experimenten wurde die Balanced-Error-Rate (BER) eingesetzt. Diese ermittelt den Durchschnitt der Anteile falsch klassifizierter Beobachtungen in jeder Klasse. Zum Beispiel wird das Maß bei zwei möglichen Klassen folgendermaßen berechnet:

$$BER = 0.5 * \left(\frac{Fehler_1}{Beob_1} + \frac{Fehler_2}{Beob_2} \right). \tag{24}$$

Die Zahlen 1 und 2 geben dabei die Klassen an, $Fehler$ die Anzahl falsch vorhergesagter Beobachtungen der entsprechenden wahren Klasse, $Beob$ die Anzahl aller Beobachtungen mit der entsprechenden wahren Klasse [50].
Da die Vorhersagegüte je nach verwendeten Beobachtungen variieren kann, wurde eine stratifizierte äußere und innere Kreuzvalidierung durchgeführt. Die innere stand SMBO für die Optimierung zur Verfügung. Die äußere diente später zur Validierung der Ergebnisse aus der Optimierung, um zum Beispiel eine Überanpassung der Methoden feststellen zu können. Für den Vergleich der Optimierungs-Methoden waren sowohl die Ergebnisse aus der inneren, als auch aus der äußeren Kreuzvalidierung von Interesse.
Für die Surrogat-Modelle, die Optimierung des Infill-Kriteriums und des initialen Designs dienten die selben Konfigurationen wie bei den synthetischen Funktionen (siehe Kapitel 4.3.1). Außerdem wurden auch bei den realen Beispielen 20 Replikationen pro Methode auf jedem Problem durchgeführt und pro Dimension 20 Iterationen erlaubt. Die Datensätze auf denen die SVMs Anwendung fanden, wurden aus OpenML [48] ausgewählt. Das Vorgehen für die Auswahl der Datensätze ist im nächsten Kapitel ausführlich beschrieben.

4.4.2 Auswahl der Datensätze

Für den Vergleich der Optimierungs-Methoden sind Datensätze geeignet, bei denen nur wenige Kombinationen der Hyperparameter der SVM zu guten Ergebnissen führen. Bei einem einfachen Problem würden die Methoden voraussichtlich zu sehr ähnlichen Ergebnissen führen und man könnte sie dadurch nicht gut gegeneinander vergleichen. Ein schwierigeres Problem kann dagegen nicht jede Methode gut lösen. Als schwierig werden Probleme angesehen, bei denen eine zufällige Auswahl an Werten für die

Parameter nur sehr selten zu einem guten Ergebnis führt. Die Optimierung über die Random Search sollte daher nur sehr langsam eine Verbesserung bei steigender Iterationenanzahl erzielen.
Um herauszufinden, welcher Datensatz für die Hyperparameter-Optimierung als schwierig gelten kann, wurde als erstes die SVM mit einer großen Anzahl an zufälligen Kombinationen der Hyperparameter berechnet. Anschließend konnte daraus ermittelt werden, wie wahrscheinlich es bei einer zufälligen Parameterauswahl ist, in die Nähe des besten Wertes zu treffen. Dieser Bereich um den besten Wert wird im folgenden als Target-Level bezeichnet. Der Bereich des Target-Levels ergab sich aus der kleinsten erreichten BER (BER_{min}) und einem festgelegten maximalem Abstand ϵ. Alle BER, die höchstens einen Wert von der kleinsten erreichten BER plus des festgelegten Abstands ϵ aufwiesen, lagen im Bereich des Target-Levels. Die Wahrscheinlichkeit, das Target-Level mit einer zufälligen Parameterauswahl zu treffen, konnte nun geschätzt werden, indem der Anteil aller Kombinationen, bei denen die BER in diesem Bereich lag, errechnet wurde.
Um sich einen Verlauf der Wahrscheinlichkeit bei einer Vergrößerung des Target-Levels ansehen zu können, wurden die Anteile für zehn verschiedene Target-Levels berechnet. Dabei addierte man für jedes neue Level den festgelegten Abstand ϵ erneut hinzu. Dadurch erhöhte sich der Bereich der getroffen werden musste Schritt für Schritt $(BER_{min} + \epsilon, BER_{min} + 2\epsilon, ..., BER_{min} + 10\epsilon)$.
Datensätze, die eine geringe Wahrscheinlichkeit aufwiesen, dass sich bei einer zufälligen Parameterauswahl die BER im ersten Target-Level Bereich befand, wurden als schwer eingestuft. Blieb die Wahrscheinlichkeit über mehrere Target-Levels hinweg niedrig, so handelte es sich um Datensätze, bei denen SMBO gegenüber Random Search voraussichtlich die größten Verbesserungen erzielen kann. War die Wahrscheinlichkeit dagegen hoch ins erste Target-Level zu treffen, so konnte man selbst mit einer zufälligen Suche relativ einfach die beste Kombination finden. Diese Datensätze waren somit nicht für den Vergleich der Algorithmen geeignet, da hier SMBO gegenüber Random Search möglicherweise zu keiner Verbesserung geführt hätte und die verschiedenen SMBO Konfigurationen voraussichtlich zu sehr ähnlichen Ergebnissen gekommen wären.
Für das zweidimensionale Optimierungsproblem wählte man 1000 zufällige Parameterkombinationen und ein ϵ von 0.025 aus. Für den fünfdimensionalen Fall legte man 2500 Kombinationen und ebenfalls ein ϵ von 0.025 fest.
Die Datensätze wurden von OpenML verwendet und dabei eine Vorauswahl getroffen. Und zwar kamen nur Datensätze zum Einsatz, die für ein Klassifikationsproblem geeignet sind und keine fehlenden Werte enthielten. Außerdem wurde auf die Anzahl

der Beobachtungen (1000-20000), Anzahl der Merkmale (10-2000) und Anzahl der Klassen (2-100) geachtet. Insgesamt erfüllten 92 Datensätze aus OpenML diese Kriterien (Stand: Oktober 2016). Um zu vermeiden, dass die späteren Berechnungen mit SMBO sehr viel Zeit in Anspruch nehmen, wurden Datensätze, die mehr als 40 Stunden Berechnung für die zufälligen Parameterkombinationen benötigten, aus den 92 Stück ausgeschlossen. Letzten Endes blieben durch diese Grenze 50 Datensätze im zweidimensionalen Fall und 30 Datensätze im fünfdimensionalen Fall übrig.

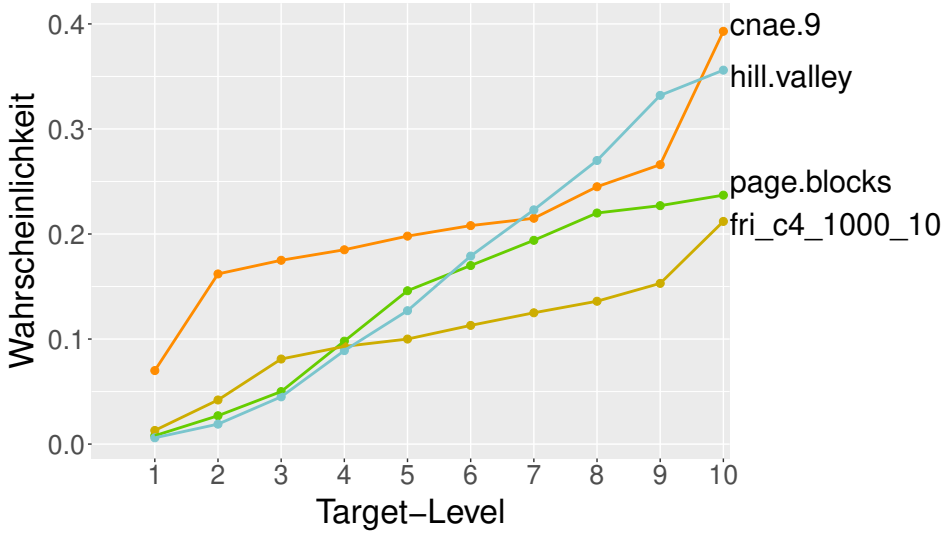

Abbildung 26 Verwendete Datensätze für das zweidimensionale Optimierungsproblem. y-Achse: Wahrscheinlichkeit in das entsprechende Target-Level zu treffen. Der Wert des Target-Levels ergibt sich aus $BER_{min} + 0.025 * TargetLevel$.

Abbildung 26 zeigt den Verlauf der vier ausgewählten Datensätze für den zweidimensionalen Fall. Die ausgewählten Daten befanden sich unter den 10 Datensätzen mit der niedrigsten Wahrscheinlichkeit, das 1. Target-Level zu erreichen. Zusätzlich befand sich die Wahrscheinlichkeit auch in den nächsten beiden Target-Levels noch unterhalb einer Wahrscheinlichkeit von 0.1. Eine Ausnahme bildet allerdings der Datensatz *cnae*.9. Dieser Datensatz wurde nicht aufgrund der niedrigen Wahrscheinlichkeit gewählt, sondern aufgrund dessen, dass sich hier bei dem Problem der Hyperparameter-Optimierung ein lokales Optimum mit der 3D-Visualisierung der zufälligen Konfigurationen zeigte.
In Abbildung 27 sehen wir den Verlauf der Wahrscheinlichkeiten der vier Datensätze, die für das fünfdimensionale Problem ausgewählt wurden. Der Datensatz $fri\_c4\_1000\_10$ zeigt sich hier auch noch bei einem hohen Target-Level unter einer Wahrscheinlichkeit von 0.1. Zum Beispiel ist die WSK eine BER von $BER_{min} + 0.2$ (Target-Level 8) zu erreichen bei ca. einer WSK von 0.1.

Auf den ausgewählten Datensätzen wurden nun die verschiedenen Methoden für die Optimierung der Hyperparameter einer SVM angewendet. Die Ergebnisse sind im nächsten Kapitel aufgeführt.

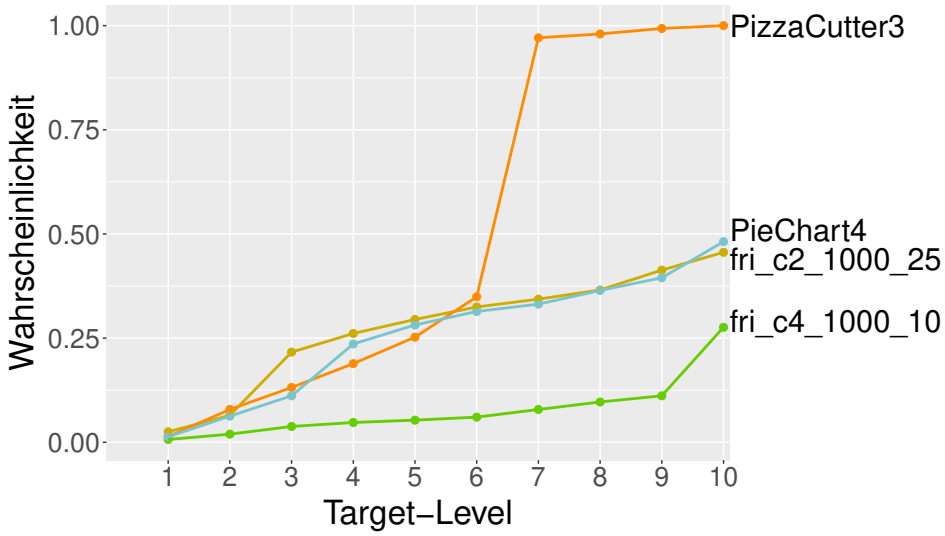

Abbildung 27 Verwendete Datensätze für das fünfdimensionale Optimierungsproblem. y-Achse: Wahrscheinlichkeit in das entsprechende Target-Level zu treffen. Der Wert des Target-Levels ergibt sich aus $BER_{min} + 0.025 * TargetLevel$.

4.4.3 Ergebnisse reale Datenbeispiele

Die Ergebnisse der zweidimensionalen Hyperparameter-Optimierung sind in Abbildung 28 zu sehen. Hierbei werden die Ergebnisse der einzelnen Replikationen bei jedem Datensatz gezeigt, wobei bei jeder Replikation jeweils der Mittelwert der Balanced-Error-Rates aus der inneren Kreuzvalidierung verwendet wurde. Es ergibt sich hier ein ähnliches Bild wie bei den zweidimensionalen synthetischen Funktionen. Bei allen Datensätzen bis auf *cnae*.9 liefert das neue Infill-Kriterium die besseren Ergebnisse gegenüber den Methoden ohne Verwendung des verbotenen Raums. Die Konfigurationen $RF\_M\_par\_4$ und $ET\_M\_par\_3$ schneiden dabei von allen Konfigurationen des neuen Infill-Kriteriums am besten ab. Bei dem Datensatz *cnae*.9 zeigt sich das Problem, das sich bei Verwendung eines einfach zu lösenden Problems ergeben kann. Die Methoden liegen hier sehr dicht beieinander und zwischen den meisten Methoden ist eine Unterscheidung nur schwer möglich.
Bei den Datensätzen *page.blocks* und $fri\_c4\_1000\_10$ schlagen $RF\_M\_par\_4$ und $ET\_M\_par\_3$ SMBO mit GPR (GPR_M, GPR_EI). Bei den anderen beiden Datensätzen liegen die beiden Methoden mit GPR auf einem Niveau.

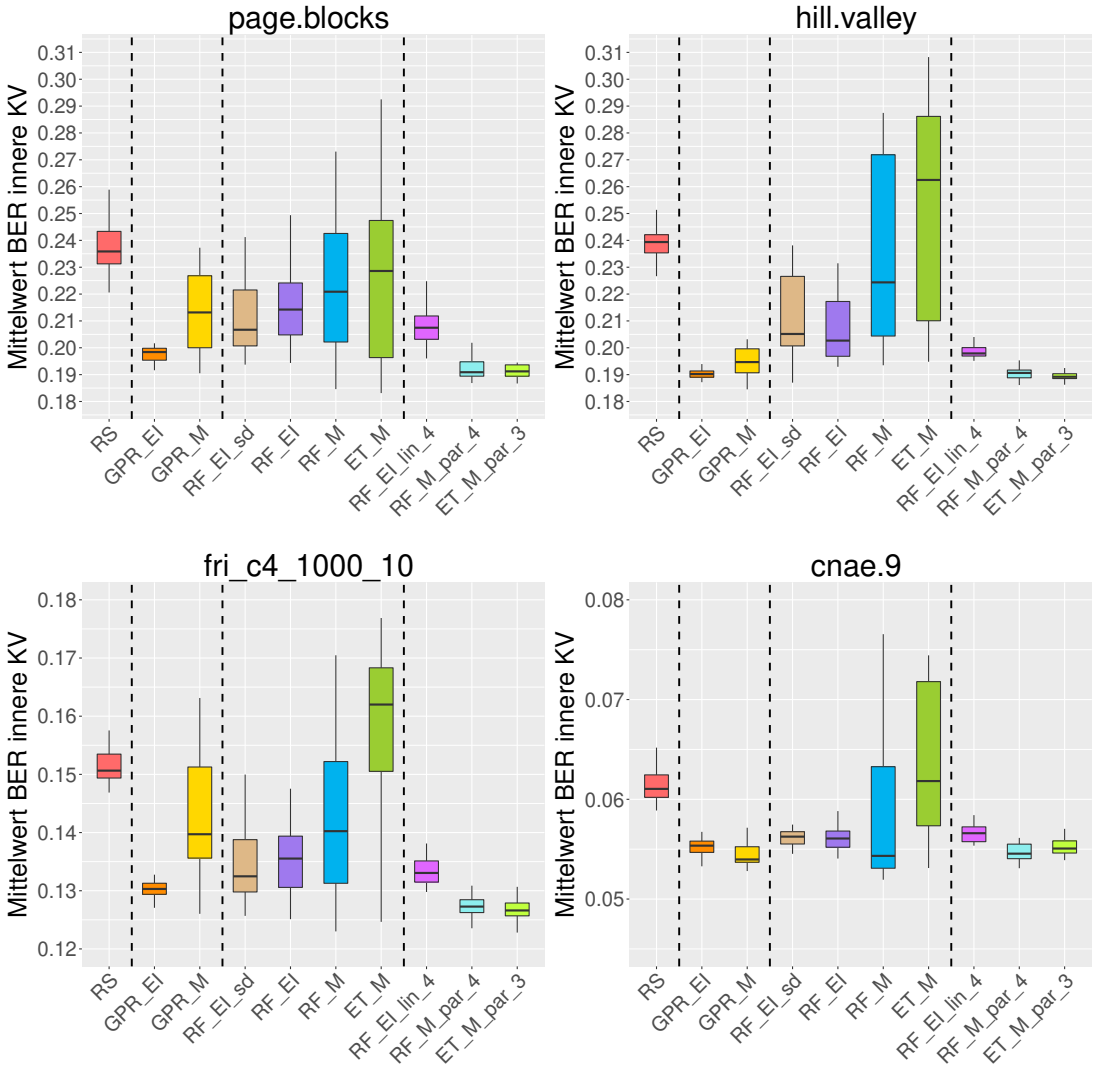

Abbildung 28 Ergebnisse der zweidimensionalen Hyperparameter-Optimierung. Die y-Achse zeigt die besten erreichten Responsewerte der Replikationen aus der inneren Kreuzvalidierung (KV). RS = Random Search, GPR = Gauss-Prozess-Regression, RF = Random Forests, ET = Extra-Trees, EI = Expected Improvement, M = Mittelwert, lin = lineare Abnahme des Mindestabstands, par = parabelförmige Abnahme des Mindestabstands, die Zahl am Ende des Namens der Methode gibt jeweils den Teiler bei der Festlegung des Startwerts an.

Das neue Infill-Kriterium ist bei Verwendung des EI ($RF\_EI\_lin\_4$) etwas schlechter als bei Anwendung des Mittelwerts ($RF\_M\_par\_4$, $ET\_M\_par\_3$). Bei den alten Infill-Kriterien ist es dagegen so, dass mit Hilfe des EI bessere Ergebnisse erzielt werden und zwar sowohl mit RF (RF_EI, RF_EI_sd) als auch mit GPR (GPR_EI). Eine Ausnahme bildet der Datensatz *cnae*.9, hier liegt bei GPR die Methode mit Mittelwert minimal unter der mit EI. Und auch bei RF liegt der Median der Methode mit Mittelwert als Infill-Kriterium (RF_M) etwas unter dem

der Methode mit EI (*RF_EI*, *RF_EI_sd*), allerdings ist die Streuung bei der Methode mit Mittelwert deutlich höher.
Das neue Infill-Kriterium schneidet bei Verwendung des RF (*RF_M_par_4*) im Vergleich zur Verwendung des Extra-Trees (*ET_M_par_3*) als Surrogat-Modell in etwa gleich gut ab.
Die Random Search (*RS*) und SMBO mit Extra-Trees ohne verbotenem Raum (*ET_M*) liefern die schlechtesten Ergebnisse aller Methoden, wobei die Random Search bis auf den Datensatz *page.blocks* sogar besser gegenüber Extra-Trees abschneidet.
Anhand der äußeren Kreuzvalidierung ließ sich eine mögliche Überanpassung der Methoden untersuchen. Die Ergebnisse zeigen keine Überraschungen. Wie erwartet erzielten alle Methoden auf Daten, die nicht für die Anpassung der Optimierung verwendet wurden, geringfügig schlechtere Werte der BER. Das Verhalten zwischen den Methoden änderte sich nicht. Die Grafik ist in Anhang A.4 zu finden.
Als nächstes kommen wir zur Auswertung der Hyperparameter-Optimierung mit fünf Parametern. Abbildung 29 visualisiert die Ergebnisse. Zwar schneiden im fünfdimensionalen Bereich die Methoden mit dem neuen Infill-Kriterium gegenüber der Random Search besser ab, jedoch präsentieren sie sich im Vergleich zu den übrigen Methoden bei allen verwendeten Datensätzen schlechter. Die Anwendung des verbotenen Raums führt somit zu keinen guten Ergebnissen.
Schaut man sich den Unterschied zwischen der Verwendung von EI und Mittelwert an, so sieht man, dass es bei allen Methoden, egal ob mit oder ohne verbotenem Raum, keinen großen Unterschied gibt. Allerdings ist die Streuung bei SMBO mit Mittelwert und ohne verbotenem Raum (*RF_M*) meist größer als bei EI (*RF_EI*, *RF_EI_sd*).
Bei der äußeren Kreuzvalidierung, zu sehen in Abbildung 30, ergab sich ein anderes Bild. Die SMBO Konfigurationen mit und ohne verbotenem Raum kommen hier zu sehr ähnlichen Ergebnissen. Auch die Random Search (*RS*) liegt bei den Datensätzen *PizzaCutter3* und *fri_c2_1000_25* auf einem Niveau mit den anderen Methoden. Somit erzielen die Methoden zwar auf der inneren KV Unterschiede in der Güte. Wenn man sie aber mit der besten Konfiguration, die aus der inneren KV ermittelt wurde, auf einen anderen Datensatz anwendet, so sind die Unterschiede hier zwischen den Methoden nicht mehr gegeben. Alle Methoden neigen somit dazu, sich zu stark an die Trainingsdaten anzupassen.

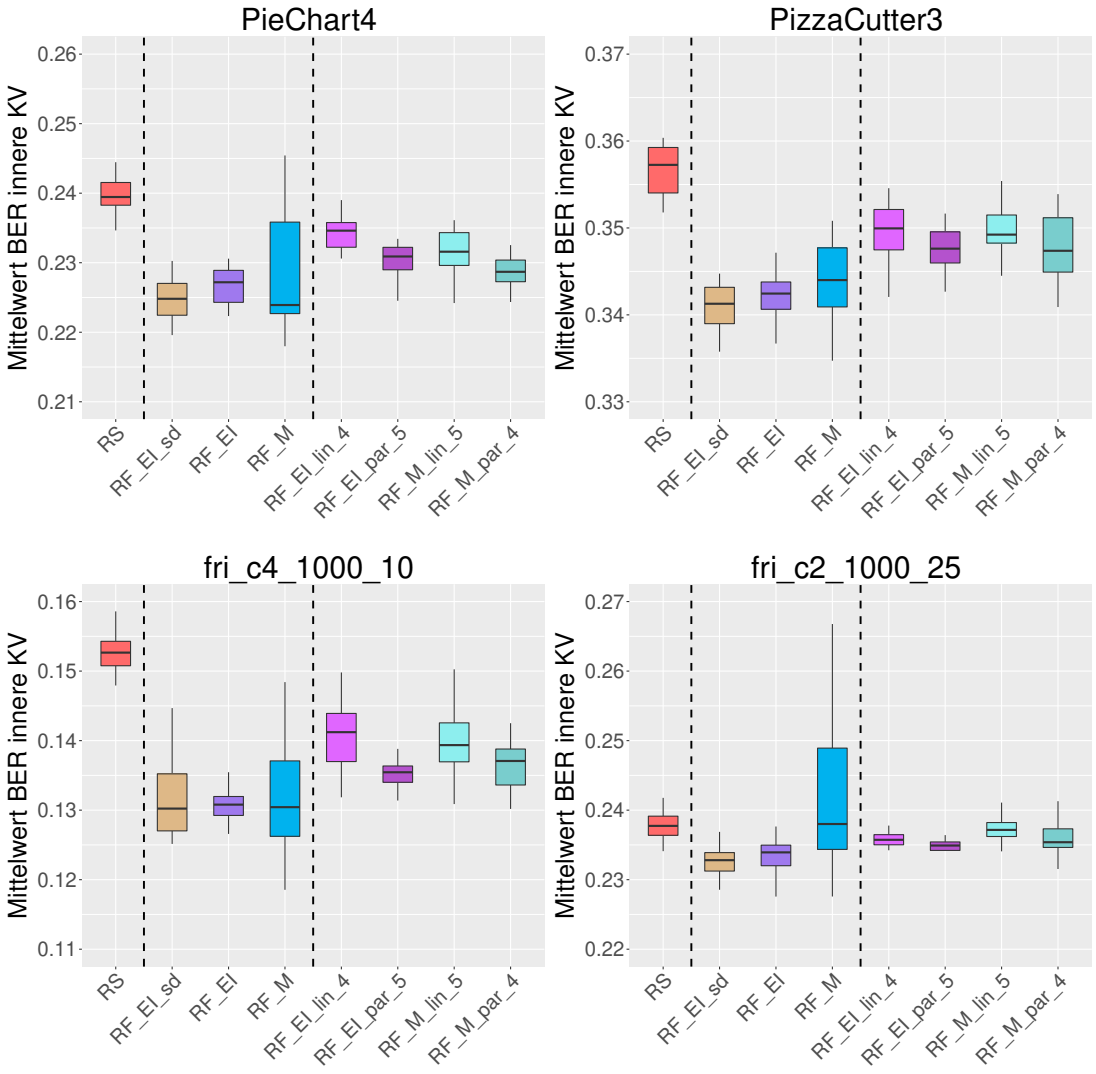

Abbildung 29 Ergenisse der fünfdimensionalen Hyperparameter Optimierung. Die y-Achse zeigt die besten erreichten Responsewerte der Replikationen aus der inneren Kreuzvalidierung (KV). RS = Random Search, GPR = Gauss-Prozess-Regression, RF = Random Forests, ET = Extra-Trees, EI = Expected Improvement, M = Mittelwert, lin = lineare Abnahme des Mindestabstands, par = parabelförmige Abnahme des Mindestabstands, die Zahl am Ende des Namens der Methode gibt jeweils den Teiler bei der Festlegung des Startwerts an.

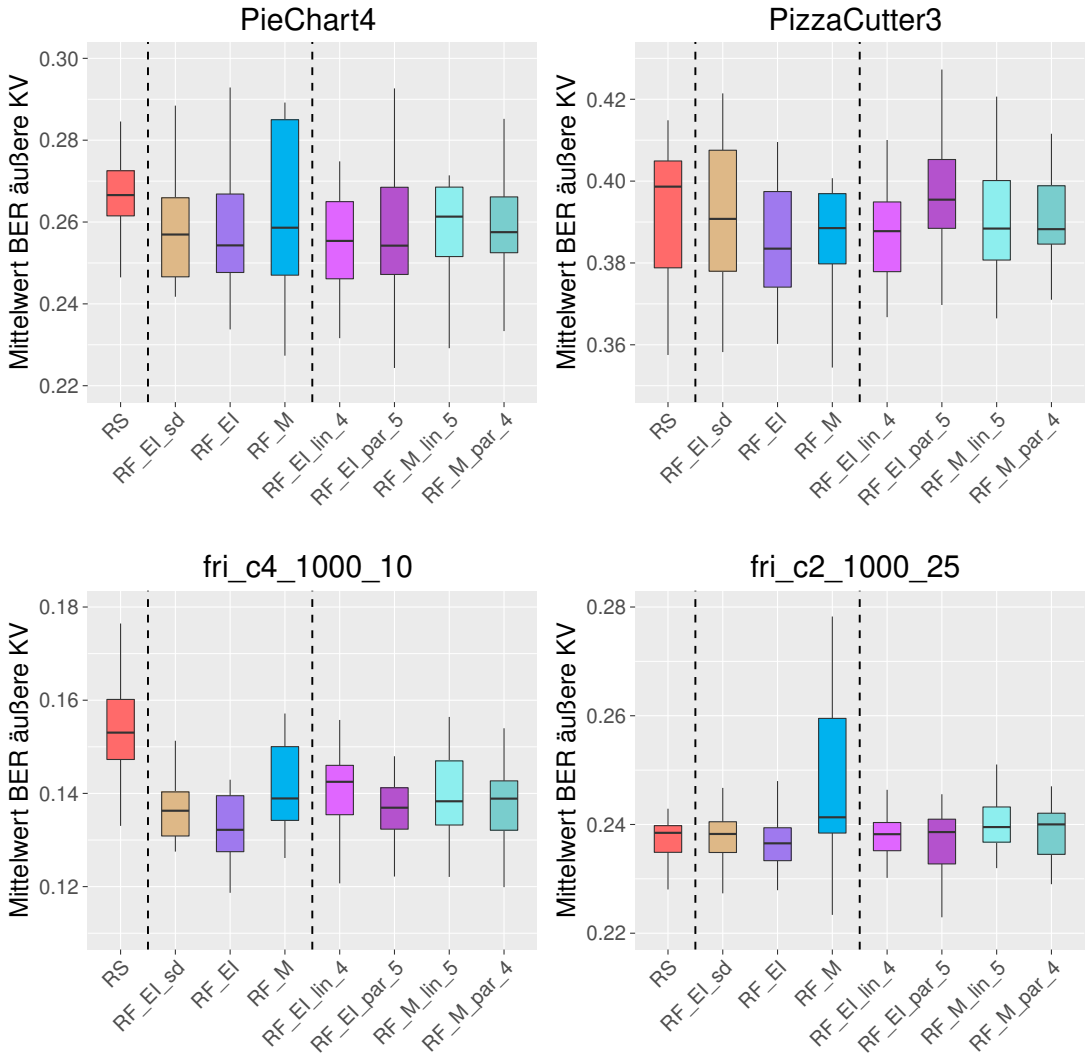

Abbildung 30 Ergenisse der fünfdimensionalen Hyperparameter Optimierung. Die y-Achse zeigt die besten erreichten Responsewerte der Replikationen aus der äußeren Kreuzvalidierung (KV). RS = Random Search, GPR = Gauss-Prozess-Regression, RF = Random Forests, ET = Extra-Trees, EI = Expected Improvement, M = Mittelwert, lin = lineare Abnahme des Mindestabstands, par = parabelförmige Abnahme des Mindestabstands, die Zahl am Ende des Namens der Methode gibt jeweils den Teiler bei der Festlegung des Startwerts an.

4.4.4 Zusammenfassung reale Datenbeispiele

Bei der Hyperparameter-Optimierung von zwei Parametern einer SVM zeigte SMBO in Kombination mit dem neuen Infill-Kriterium vielversprechende Ergebnisse. Das neue Infill-Kriterium mit Mittelwert präsentierte sich hier sogar besser oder gleich gut wie SMBO mit GPR als Surrogat-Modell. Zwischen den Surrogat-Modellen Random Forest und Extra-Trees gab es bei Verwendung des neuen Infill-Kriteriums keinen Unterschied, wobei man aufgrund der geringeren Rechenzeit Extra-Trees den Vortritt lassen würde.
Wie schon bei den synthetischen Funktionen nimmt die Wirkung des neuen Infill-Kriteriums bei höherer Dimension aber ab. So hat sich bei der Hyperparameter-Optimierung mit fünf Parametern gezeigt, dass das neue Infill-Kriterium schlechtere Ergebnisse als etablierte Infill-Kriterien liefert. Ein möglicher Hinweis für die Ursache, gibt die Tatsache, dass das neue Infill-Kriterium mit Mittelwert gegen das Infill-Kriterium mit Mittelwert und ohne verbotenem Raum schlechter abschneidet. Das bedeutet, dass eine geringere Exploration des Raums zu besseren Ergebnissen führt. Es könnte somit sein, dass in höherer Dimension die Methode mehr Iterationen für die Feinregulierung benötigt, um zu guten Ergebnissen zu kommen. Interessant wäre es, weitere Konfigurationen zu testen. Dadurch könnte abgeklärt werden, ob die Performance-Abnahme der Methoden in höherer Dimension im Zusammenhang mit der Konfiguration des verbotenen Raums steht. Innerhalb des Zeitrahmens der Masterarbeit war es jedoch nicht möglich, weitere Konfigurationen zu testen.

5 Zusammenfassung und Ausblick

Zu Beginn wurde in dieser Masterarbeit auf die Theorie von SMBO eingegangen. Dabei wurden auch die Surrogat-Modelle Gauss-Prozess-Regression, Random Forests und Extra-Trees behandelt. Anschließend wurde das alternative Infill-Kriterium für Random Forest vorgestellt. Dieses Kriterium trägt dem Problem Rechnung, dass die Varianzschätzung bei Random Forest nicht optimal geeignet ist, um sie für den Vorschlag neuer Auswertungsstellen zu verwenden. Die neu entwickelte Methode verzichtet auf die Varianzschätzung und benötigt nur den Random Forest Schätzer, bzw. Mittelwert des Surrogat-Modells. Bei bloßer Verwendung des Mittelwerts kann es jedoch dazu kommen, dass der Parameterraum nicht ausreichend erkundet wird. Aus diesem Grund legt man Bereiche, sogenannte verbotene Räume, um die Designpunkte fest, in denen keine neuen Auswertungen durchgeführt werden dürfen. Da sich die verbotenen Räume über die Iterationen hinweg immer weiter verkleinern, wird eine Feinregulierung des besten Wertes am Ende des Prozesses erlaubt.
Um das neue Infill-Kriterium zu testen, wurden Benchmark-Experimente mit synthetischen Funktionen und realen Beispielen durchgeführt. Die realen Beispiele beinhalteten eine zwei- und fünfdimensionale Hyperparameter-Optimierung einer SVM. Sowohl bei den durch die synthetischen Funktionen künstlich erzeugten Problemen, als auch bei den Tests unter realen Bedingungen, zeigten sich im zwei- und dreidimensionalen Fall vielversprechende Ergebnisse. Das neue Kriterium präsentierte sich hier meistens sogar besser, als SMBO mit GPR als Surrogat-Modell und EI oder Mittelwert als Infill-Kriterium. Bei steigender Dimension änderte sich jedoch das Bild. Bei den synthetischen Funktionen spielte es spätestens ab dem zehndimensionalen Fall keine Rolle mehr, ob man den verbotenen Raum in das Infill-Kriterium integrierte. Bei der Hyperparameter-Optimierung von fünf Parametern war es sogar so, dass das neue Infill-Kriterium schlechter als die Methoden ohne verbotenem Raum abschnitt. Aufgrund der Ergebnisse ist die Anwendung des alternativen Infill-Kriteriums nur im niedrigdimensionalen Fall empfehlenswert.
Um eventuell zu besseren Ergebnissen zu kommen, könnte man noch eine andere Abnahme des verbotenen Raums über die Iterationen betrachten. Womöglich kann es sein, dass im höherdimensionalen Fall eine andere Funktion für die Abnahme des Mindestabstands zu besseren Ergebnissen führt.
Außerdem ist eine weitere Idee, die neue Methode zu modifizieren, indem man den verbotenen Raum nicht nur ab- sondern auch wieder zunehmen lässt. Der verbotene Raum soll sich so schnell verkleinern, dass er schon nach wenigen Iterationen nicht mehr vorhanden ist. Sobald sich aber nun zum Beispiel zwei bis dreimal bei einer neuen Auswertung keine große Verbesserung ergibt, führt man wieder verbotene

Räume ein, um eine Exploration durchführen zu können. Anschließend verkleinern sich die verbotenen Räume aufs Neue schon nach wenigen Iterationen auf null. Die Bereiche, in denen aber mehrere Designpunkte sehr dicht beieinander liegen und ein geringer Unterschied in der Responsevariablen vorhanden ist, bleiben weiterhin durch einen verbotenen Raum gesperrt. Durch dieses Vorgehen kann erreicht werden, dass lokale Optima schon relativ früh entdeckt werden und sich die Suche auf ein möglicherweise noch unentdecktes globales Optimum konzentriert.
Eine weitere denkbare Möglichkeit, die zwar von der Idee mit den verbotenen Räumen wegführt, aber auch für die Optimierung in kategorialen und gemischten Parameterräumen geeignet wäre, ist der Ansatz über generalisierte additive Modelle für Lokations-, Skalen- und Schiefeparameter (GAMLSS). GAMLSS kann man als semi-parametrisches Regressionsmodell betrachten. Parametrisch, weil für die Responsevariable eine Verteilungsannahme vorausgesetzt wird und semi weil die Modellierung der Verteilungsparameter non-parametrische Funktionen beinhalten kann [51]. Entscheidend bei GAMLSS ist, dass sie nicht nur die Modellierung des Mittelwerts erlauben, sondern auch andere Parameter der Verteilung des Response. Das Modell erlaubt somit, jeden Verteilungsparameter einzeln an die Kovariablen anzupassen. Durch diesen flexiblen Ansatz lässt sich zum Beispiel die Varianz so modellieren, dass sie in Bereichen mit wenigen Beobachtungen zunimmt, wohingegen sie in Bereichen mit vielen Beobachtungen klein bleibt. Durch diese extra Modellierung der Varianz, könnte man sie vermutlich sehr gut in einem Infill-Kriterium verwenden. Die Regressionskoeffizienten lassen sich entweder über die Maximum-Likelihood-Methode, oder über Boosting (s. h. [52]) schätzen.
Die Masterarbeit hat gezeigt, dass die Verwendung eines verbotenen Raums im Infill-Kriterium zu guten Ergebnissen führen kann. Jedoch wurde auch klar, dass es für den höherdimensionalen Fall noch Anpassungsbedarf an der Methode gibt. Das Ziel weiterer Arbeiten wäre somit, das hier vorgestellte alternative Infill-Kriterium weiterzuentwickeln, oder aber, die daraus entstandenen neuen Ideen für die Optimierung mit SMBO umzusetzten.

A Anhang

Zur Bezeichnung der Grafiken im Anhang, wurden für die Methoden folgende Abkürzungen verwendet:
RS = Random Search, RF = Random Forest, ET = Extra-Trees, GPR = Gauss-Prozess-Regression, EI = Expected Improvement, M = Mittelwert, lin = lineare Abnahme des Mindestabstands, par = parabelförmige Abnahme des Mindestabstands, neg.par = negativ-parabelförmige Abnahme des Mindestabstands, sd = einfache Berechnung der Standardabweichung (Kapitel 2.2.2), Zahl = Mittelwert des initialen Designs dividiert mit dieser Zahl.

A.1 Ränge Extra-Trees

Tabelle 3 Übersicht des durchschnittlichen Rangs der einzelnen Methoden bei Verwendung von Extra-Trees als Surrogat-Modell. Je geringer der Durchschnittsrang, desto besser schnitt die Methode ab.

Methode	**Rang**
ET_ M_ par_ 3	5.290
ET_ M_ par_ 4	5.374
ET_ M_ par_ 5	5.394
ET_ M_ par_ 2	5.818
ET_ M_ lin_ 4	6.019
ET_ M_ lin_ 5	6.174
ET_ M_ lin_ 3	6.285
ET_ M_ lin_ 2	6.919
ET_ M_ neg.par_ 5	7.113
ET_ M_ neg.par_ 4	7.307
ET_ M_ neg.par_ 3	7.984
ET_ M_ neg.par_ 2	8.323

A.2 Beispielfunktion

Die in den Kapiteln 2.3.1, 2.3.3 und 3.1 verwendete Target-Funktion wurde aus der Forrester et. al Funktion [53]

$$f(x) = (6 * x - 2)^2 * sin(12 * x - 4)$$

entwickelt. Die multimodale Funktion ergibt sich daraus folgendermaßen:

$$f(x) = sin(4 * x - 4) * (2 * x - 2)^2 * sin(20 * x - 4)$$

A.3 Einzelne Ergebnisse der synthetischen Funktionen

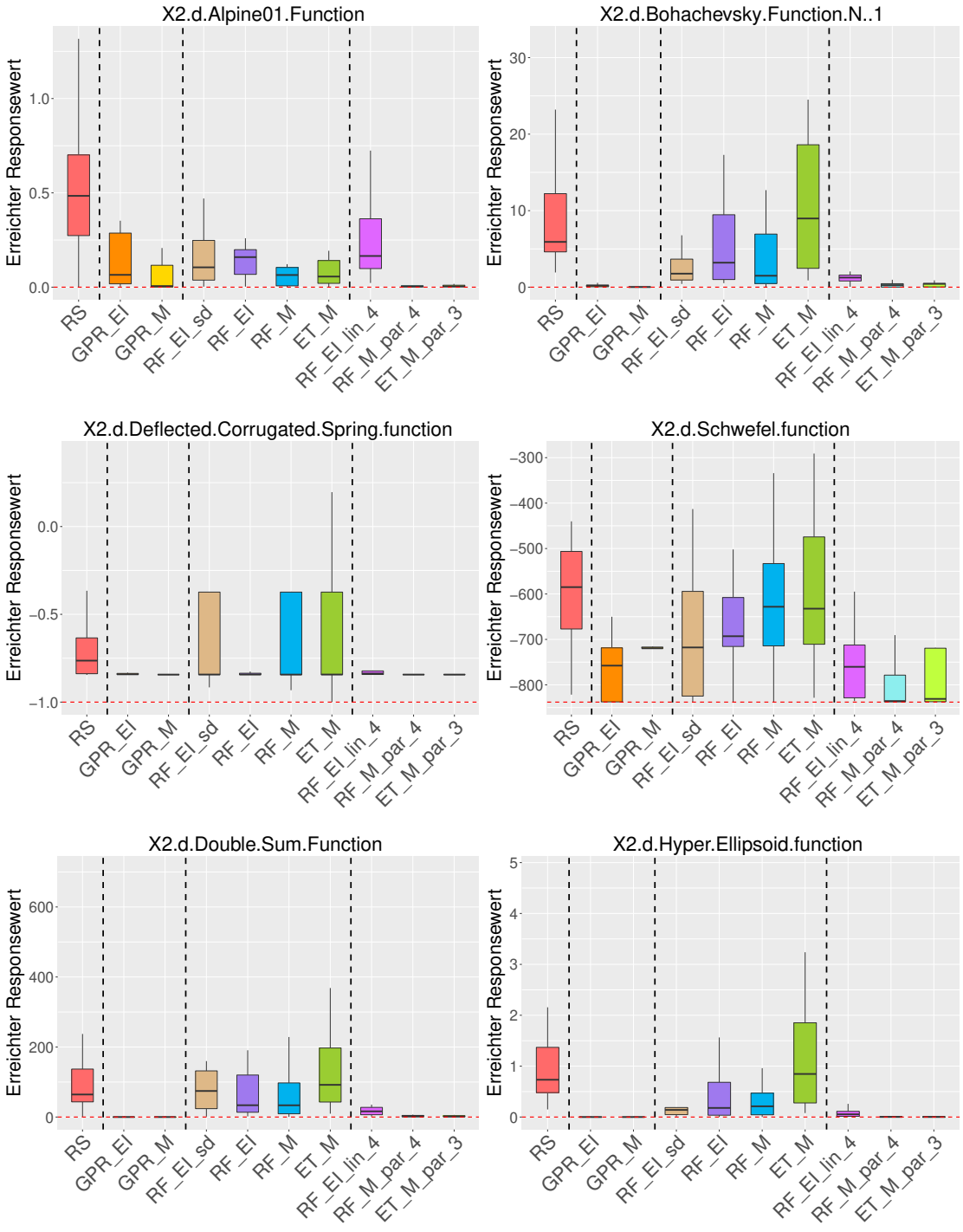

Abbildung 31 Ergebnisse zweidimensionaler synthetischer Funktionen mit stetigem Parameterraum. Die y-Achse gibt den besten erreichten Responsewert an, wobei bei jeder Methode die einzelnen Replikationen zu sehen sind. Die rote gestrichelte Linie zeigt das globale Optimum.

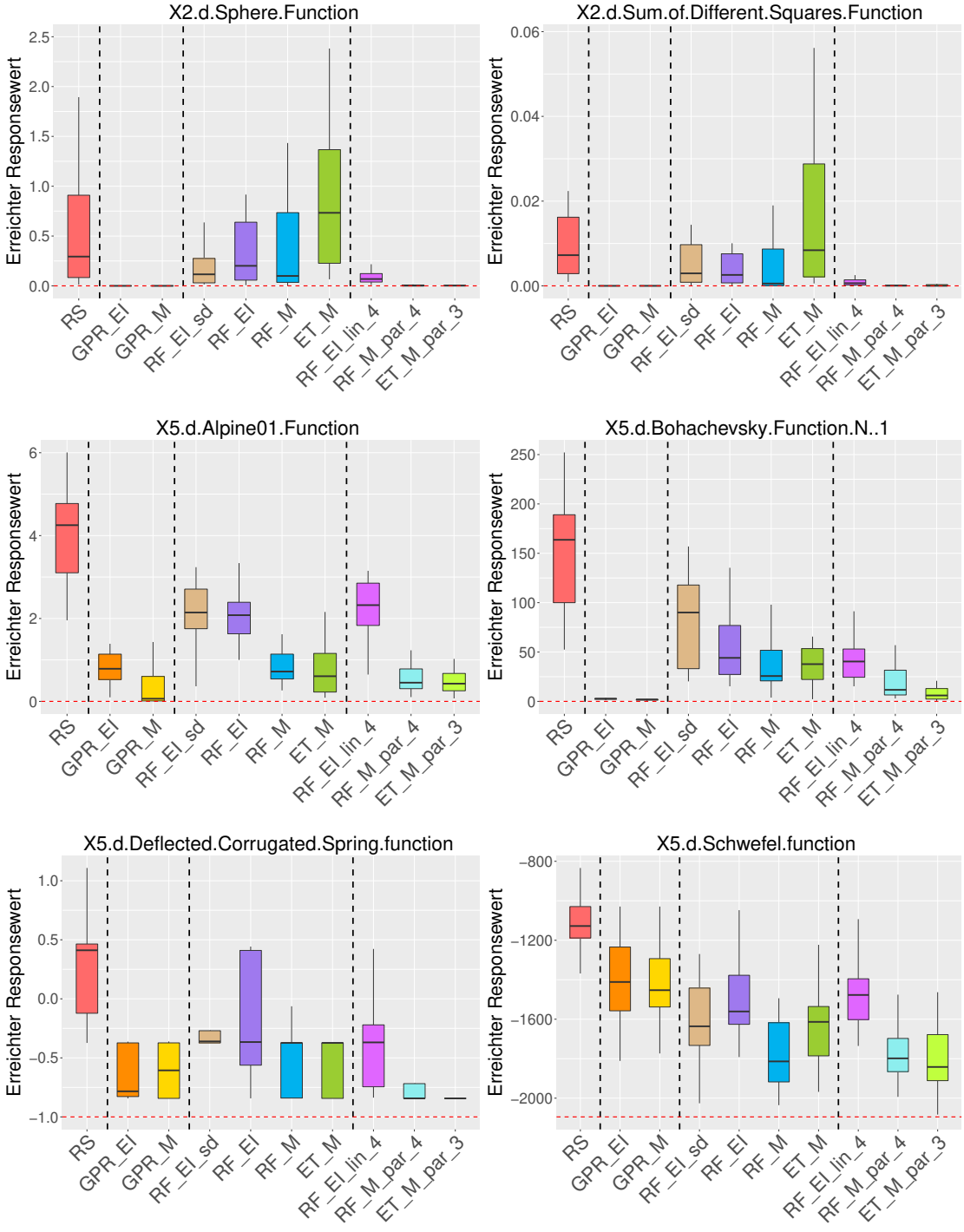

Abbildung 32 Ergebnisse zwei- und fünfdimensionaler synthetischer Funktionen mit stetigem Parameterraum. Die y-Achse gibt den besten erreichten Responsewert an, wobei bei jeder Methode die einzelnen Replikationen zu sehen sind. Die rote gestrichelte Linie zeigt das globale Optimum.

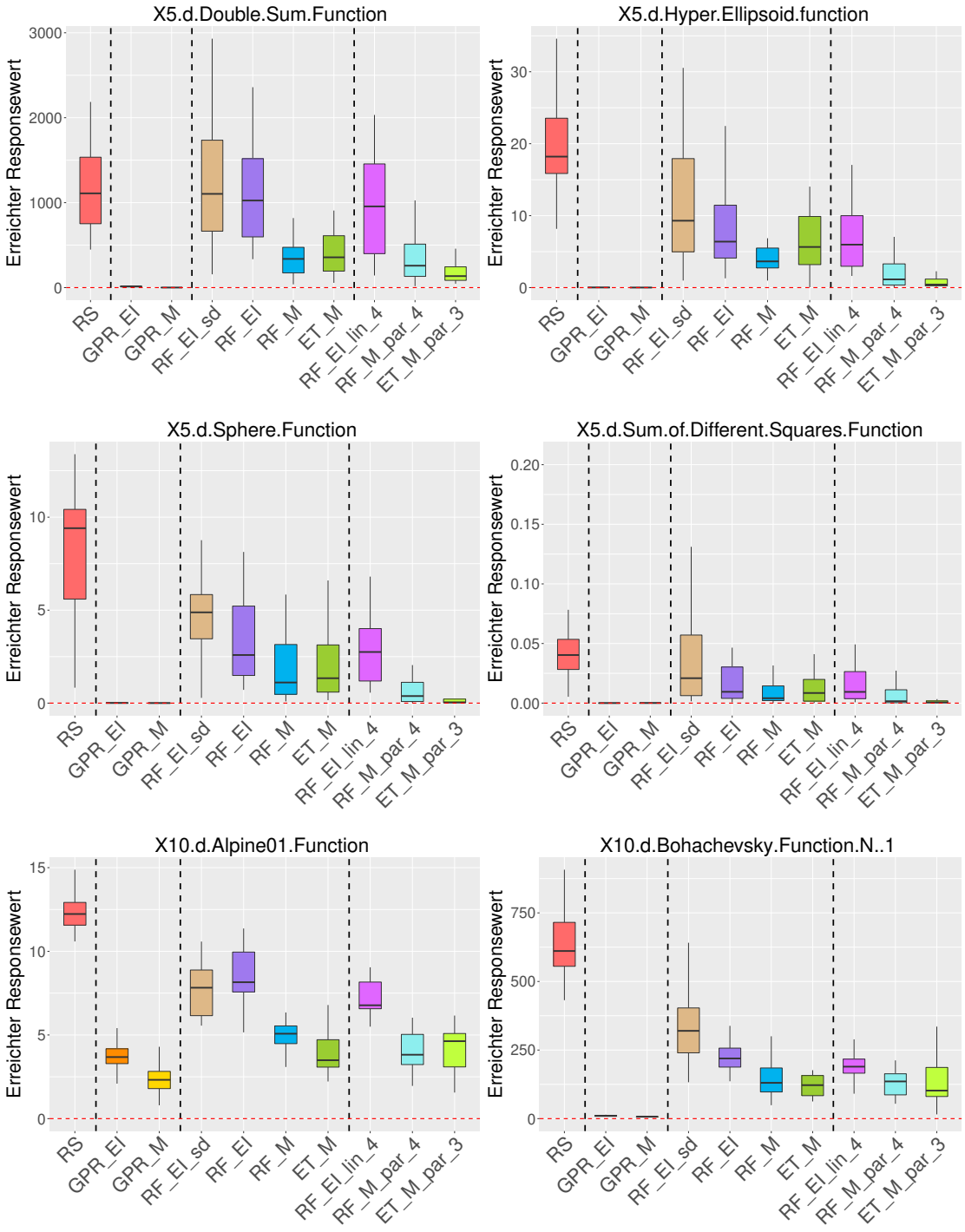

Abbildung 33 Ergebnisse fünf- und zehndimensionaler synthetischer Funktionen mit stetigem Parameterraum. Die y-Achse gibt den besten erreichten Responsewert an, wobei bei jeder Methode die einzelnen Replikationen zu sehen sind. Die rote gestrichelte Linie zeigt das globale Optimum.

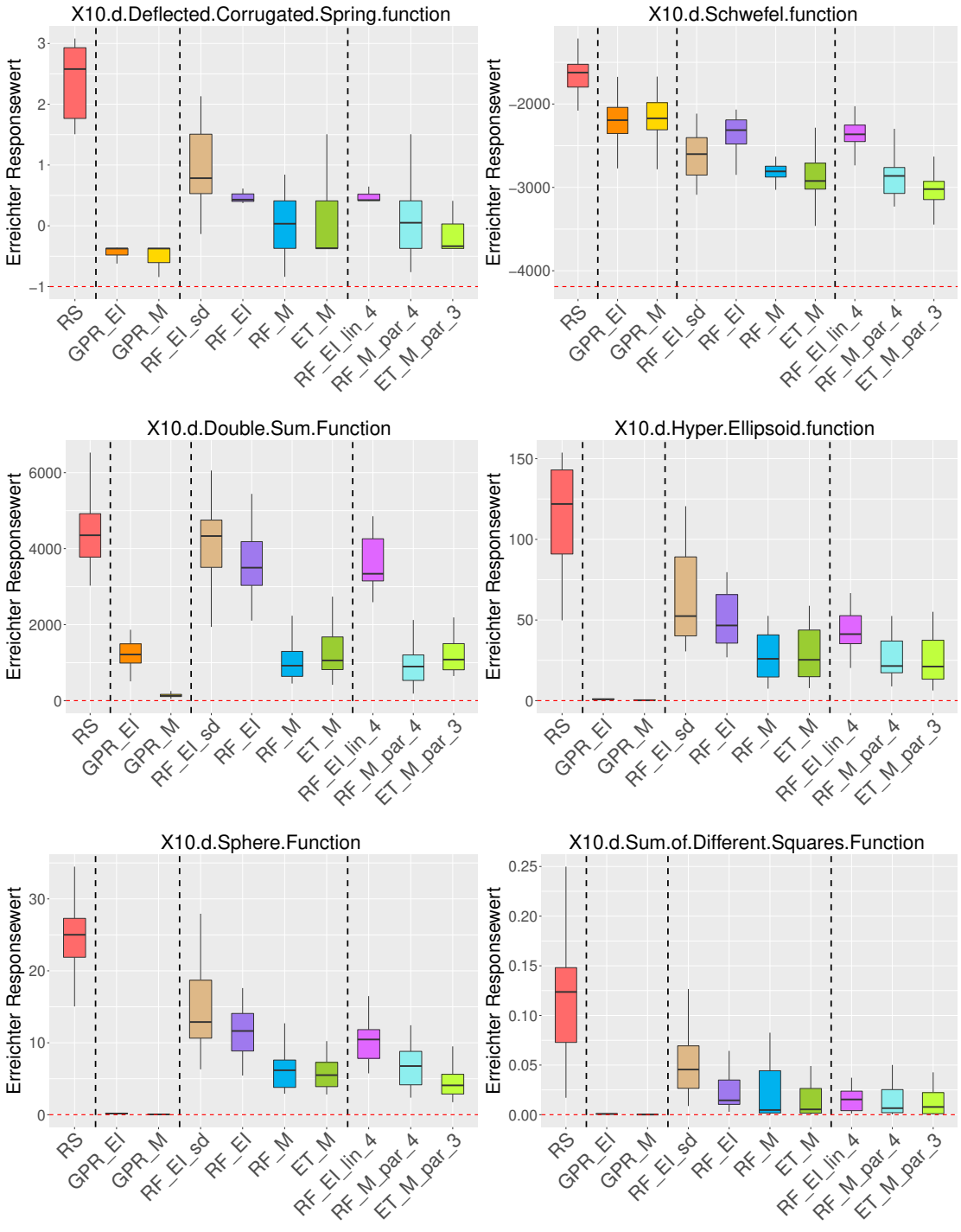

Abbildung 34 Ergebnisse zehndimensionaler synthetischer Funktionen mit stetigem Parameterraum. Die y-Achse gibt den besten erreichten Responsewert an, wobei bei jeder Methode die einzelnen Replikationen zu sehen sind. Die rote gestrichelte Linie zeigt das globale Optimum.

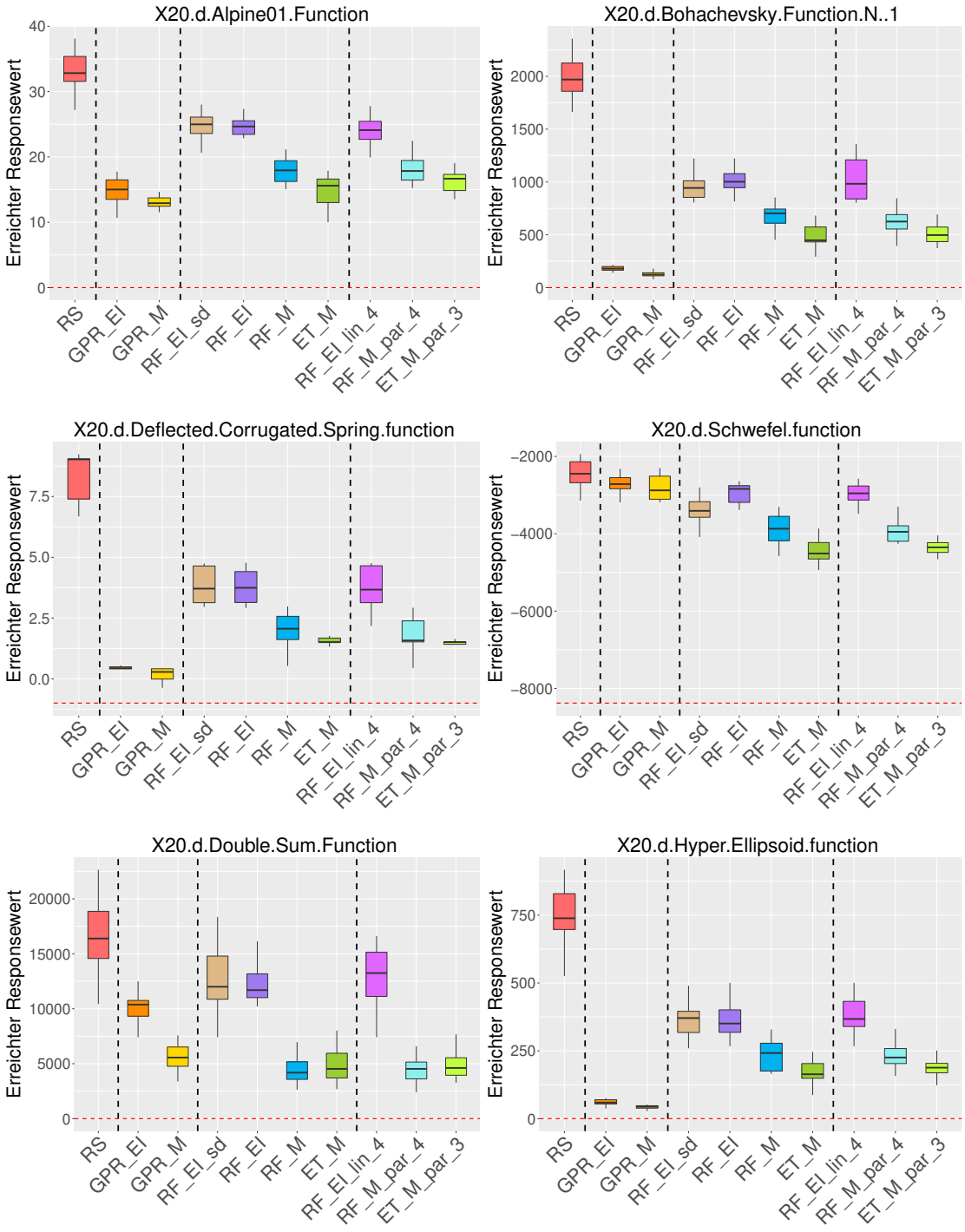

Abbildung 35 Ergebnisse zwanzigdimensionaler synthetischer Funktionen mit stetigem Parameterraum. Die y-Achse gibt den besten erreichten Responsewert an, wobei bei jeder Methode die einzelnen Replikationen zu sehen sind. Die rote gestrichelte Linie zeigt das globale Optimum.

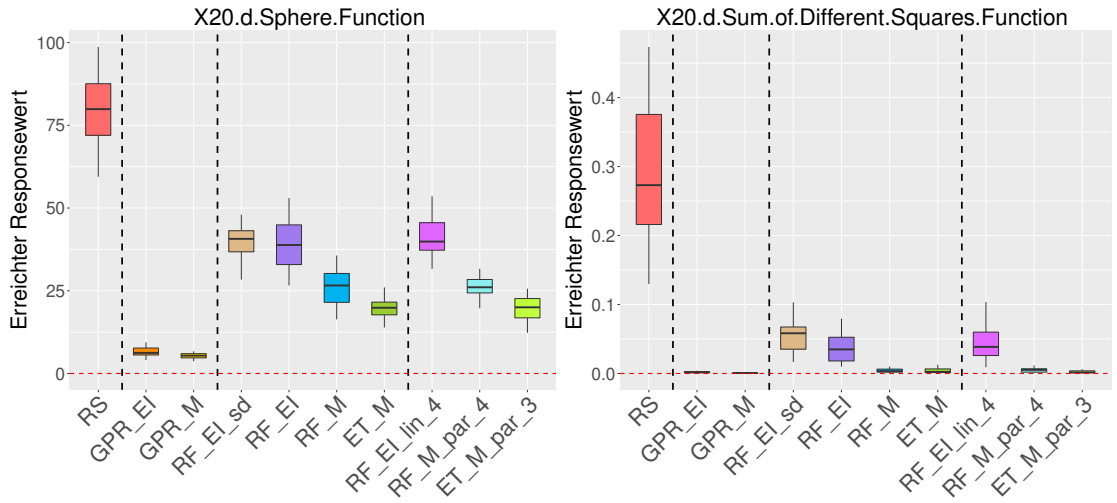

Abbildung 36 Ergebnisse zwanzigdimensionaler synthetischer Funktionen mit stetigem Parameterraum. Die y-Achse gibt den besten erreichten Responsewert an, wobei bei jeder Methode die einzelnen Replikationen zu sehen sind.

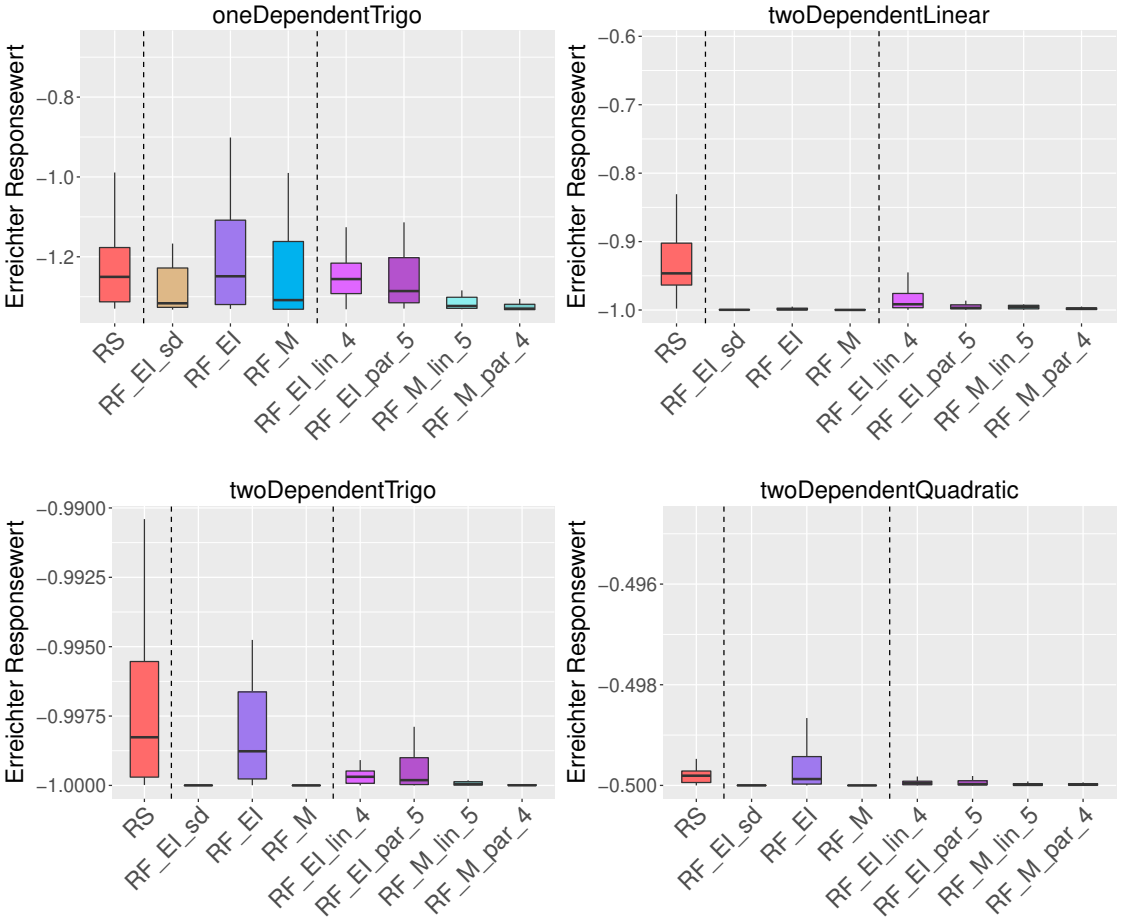

Abbildung 37 Ergebnisse synthetische Funktionen mit gemischtem Parameterraum (zweidimensional). Die y-Achse gibt den besten erreichten Responsewert an, wobei bei jeder Methode die einzelnen Replikationen zu sehen sind.

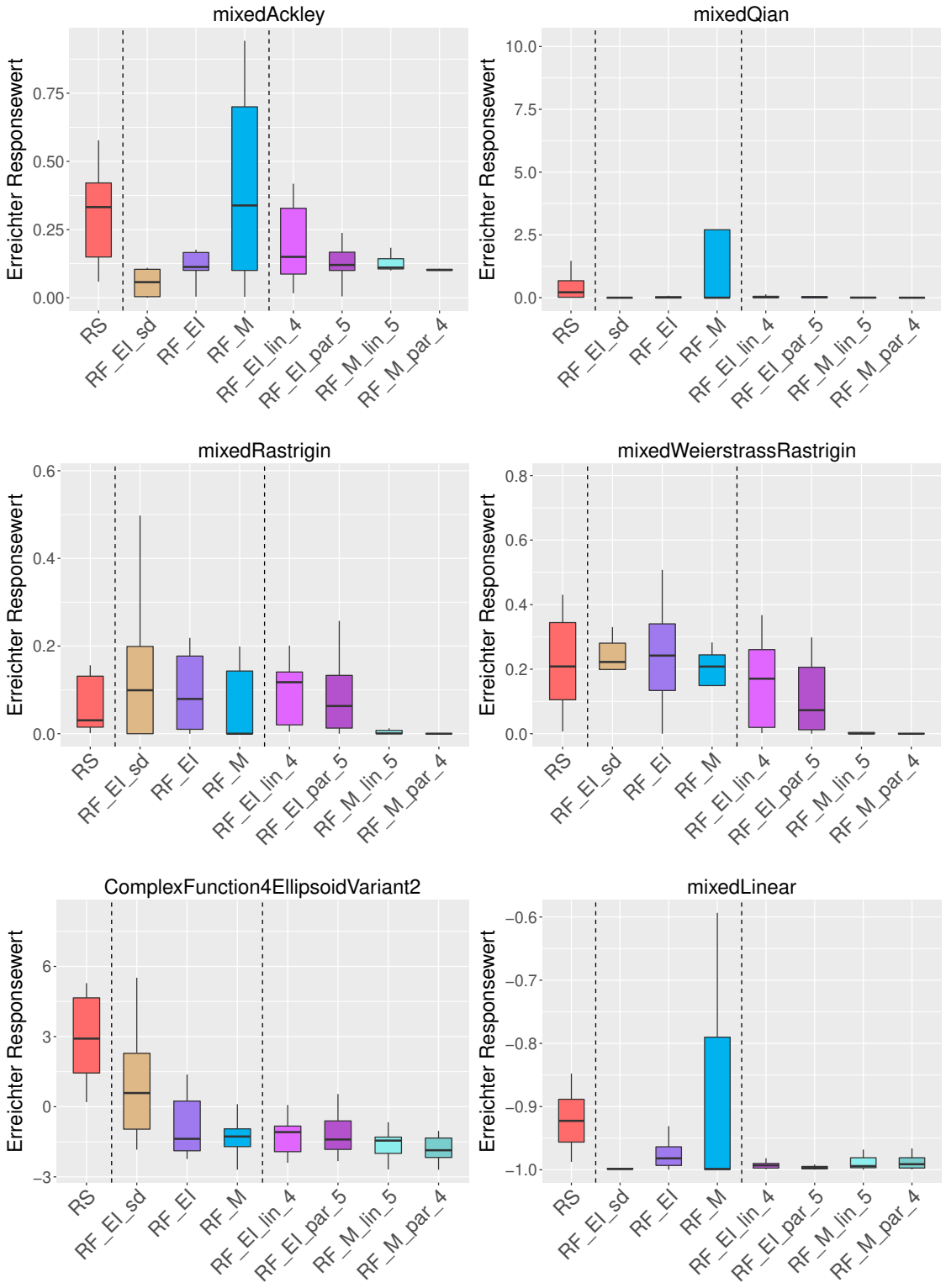

Abbildung 38 Ergebnisse synthetischer Funktionen mit gemischtem Parameterraum. Die y-Achse gibt den besten erreichten Responsewert an, wobei bei jeder Methode die einzelnen Replikationen zu sehen sind.

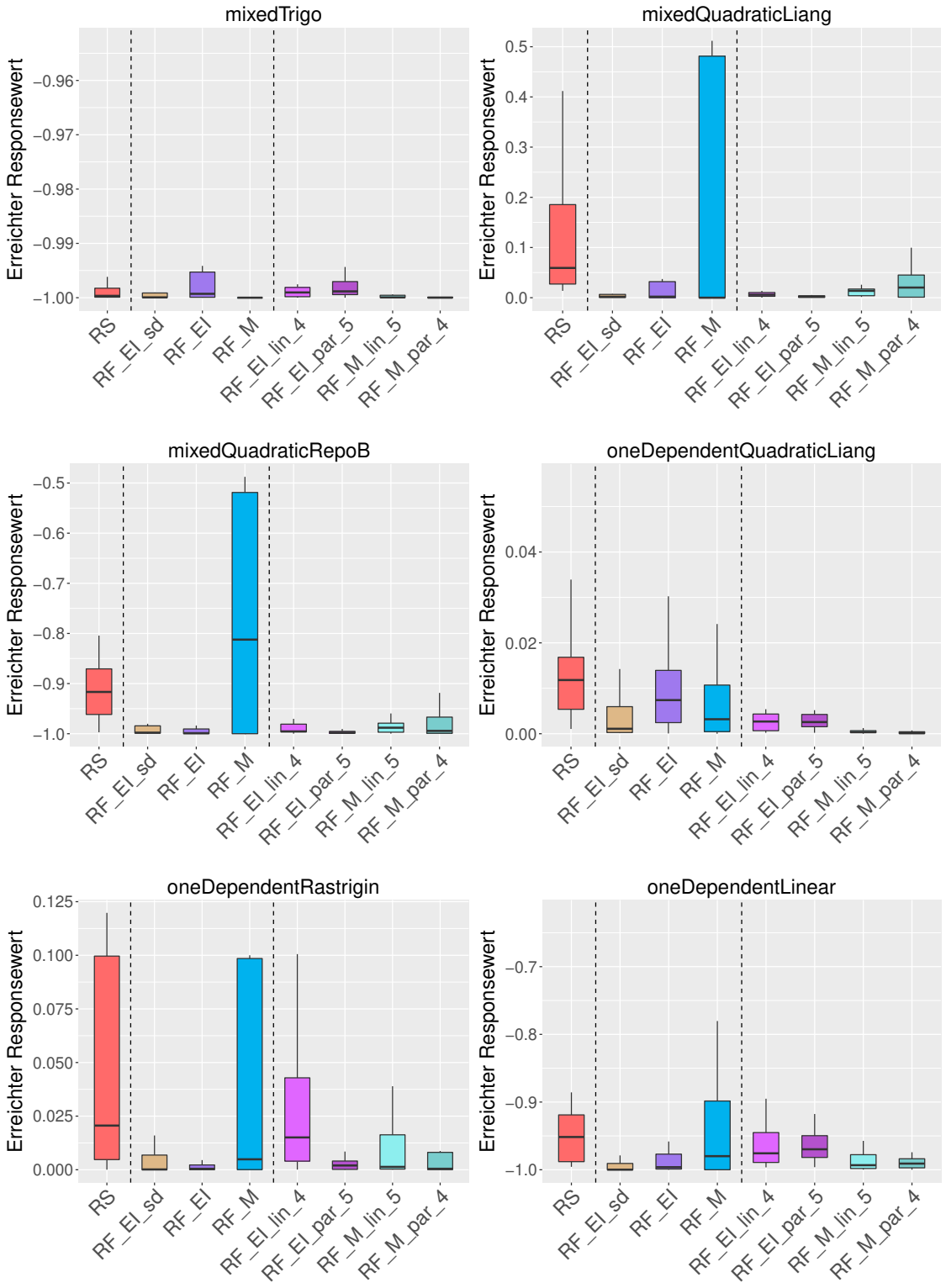

Abbildung 39 Ergebnisse synthetischer Funktionen mit gemischtem Parameterraum. Die y-Achse gibt den besten erreichten Responsewert an, wobei bei jeder Methode die einzelnen Replikationen zu sehen sind.

A.4 Ergebnisse der äußeren Kreuzvalidierung realer Datenbeispiele

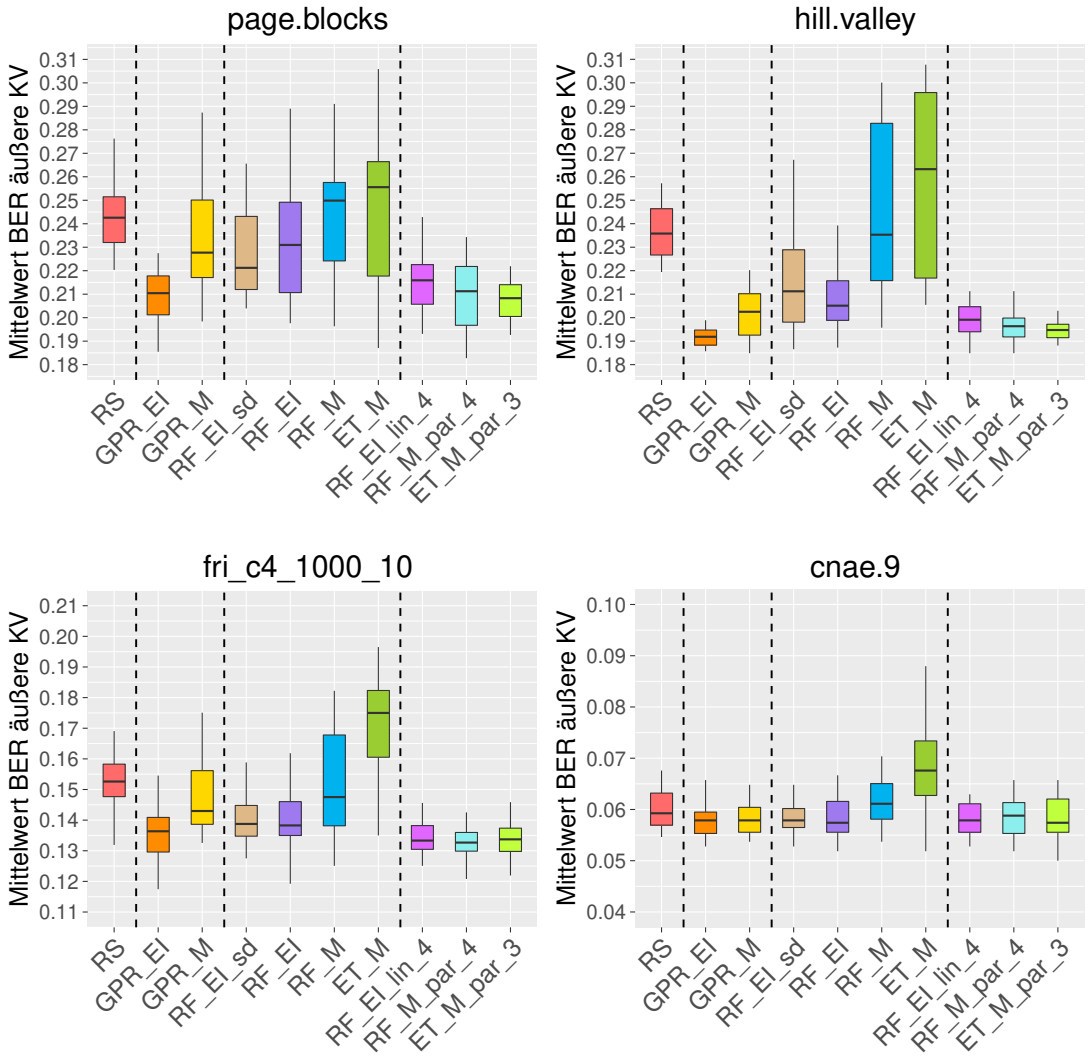

Abbildung 40 Ergebnisse der zweidimensionalen Hyperparameter-Optimierung. Die y-Achse zeigt den Mittelwert der BER aus der äußeren Kreuzvalidierung (KV). RS = Random Search, GPR = Gauss-Prozess-Regression, RF = Random Forest, EI = Expected Improvement, M = Mittelwert, lin = lineare Abnahme des Mindestabstands, par = parabelförmige Abnahme des Mindestabstands, Zahl = entspricht dem Teiler des Startwerts, sd = EI mit einfacher Form der Varianzberechnung bei RF.

Literatur

[1] Brochu E, Cora VM, De Freitas N. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. arXiv preprint arXiv. 2010; 1012.2599: 1-49.

[2] Shahriari B, Swersky K, Wang Z, Adams RP, De Freitas N. Taking the human out of the loop: A review of bayesian optimization. Proceedings of the IEEE. 2016; 104.1: 148-175.

[3] Feurer M, Springenberg JT, Hutter F. Initializing Bayesian Hyperparameter Optimization via Meta-Learning. AAAI. 2015: 1128-1135.

[4] Bischl B, Lang M, Bossek J, Horn D, Richter J, Thomas J. mlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions. Noch nicht Veröffentlicht. 2016.

[5] Friedman J, Hastie T, Tibshirani R. The elements of statistical learning. Vol. 1. Berlin: Springer series in statistics; 2001.

[6] Wiener N. Extrapolation, interpolation, and smoothing of stationary time series. Vol.2. Cambridge:MIT press; 1949.

[7] Kolmogorov AN. Interpolation und Extraplation von Stationaren Zulfulligen Folgen. Bull. Acad. Nauk. USSR, Ser. Math. 1941; 50: 3.

[8] Rasmussen CE, Williams CKI. Gaussian Processes for Machine Learning. The MIT Press; 2006.

[9] Jones DR. A taxonomy of global optimization methods based on response surfaces. Journal of global optimization. 2001; 21.4: 345-383.

[10] Martinez-Cantin R, De Freitas N, Brochu E, Castellanos J, Doucet A. A Bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot. Autonomous Robots. 2009; 27.2: 93-103.

[11] Brochu E, Brochu T, De Freitas N. A Bayesian interactive optimization approach to procedural animation design. Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Eurographics Association. 2010: 103-112.

[12] Oshiro TM, Perez PS, Baranauskas JA. How many trees in a random forest?. International Workshop on Machine Learning and Data Mining in Pattern Recognition. Springer Berlin Heidelberg. 2012: 154-168.

[13] Preuss M, Wagner T, Ginsbourger D. High-dimensional model-based optimization based on noisy evaluations of computer games. Learning and Intelligent Optimization. Springer Berlin Heidelberg. 2012: 145-159.

[14] Geurts P, Ernst D, Wehenkel L. Extremely randomized trees. Machine learning, 2006; 63.1: 3-42.

[15] Wager S, Hastie T, Efron B. Confidence Intervals for Random Forests - The Jackknife and the Infinitesimal Jackknife. Journal of Machine Learning Research. 2014; 15.1: 1625-1651.

[16] Mayer V. Random Forests und Unsicherheitsschätzung [Masterseminar Bischl B.]. LMU München. 2016.

[17] Sexton J, Laake P. Standard errors for bagged and random forest estimators. Journal of Computational Statistics & Data Analysis. 2009; 53.3: 801-811.

[18] Jones DR, Schonlau M, Welch WJ. Efficient global optimization of expensive black-box functions. Journal of Global optimization. 1998; 13.4: 455-492.

[19] Bergstra JS, et al. Algorithms for hyper-parameter optimization. Advances in Neural Information Processing Systems, 2011: 2546-2554.

[20] Hutter F, Hoos HH, Leyton-Brown K. Sequential model-based optimization for general algorithm configuration (extended version). Technical Report TR-2010-10. University of British Columbia. Computer Science. 2010.

[21] Eggensperger K, Feurer M, Hutter F, Bergstra J, Snoek J, Hoos H, Leyton-Brown K. Towards an empirical foundation for assessing bayesian optimization of hyperparameters. NIPS workshop on Bayesian Optimization in Theory and Practice. 2013: 1-5.

[22] Bergstra J, Bengio Y. Random search for hyper-parameter optimization. Journal of Machine Learning Research. 2012; 13.Feb: 281-305.

[23] Friedrichs F, Igel C. Evolutionary tuning of multiple SVM parameters. Neurocomputing. 2005; 64: 107-117.

[24] Loshchilov I, Hutter F. CMA-ES for Hyperparameter Optimization of Deep Neural Networks. arXiv. 2016; 1604.07269: 1-8.

[25] Birattari M, et al. F-Race and iterated F-Race: An overview. Experimental methods for the analysis of optimization algorithms. Springer Berlin Heidelberg. 2010: 311-336.

[26] Birattari M, Stützle T, Paquete L, Varrentrapp K. A Racing Algorithm for Configuring Metaheuristics. GECCO. 2002; Vol. 2: 11-18.

[27] Ansótegui C, Sellmann M, Tierney K. A gender-based genetic algorithm for the automatic configuration of algorithms. International Conference on Principles and Practice of Constraint Programming. Springer Berlin Heidelberg. 2009: 142-157.

[28] Hutter F, Hoos HH, Stützle T. Automatic algorithm configuration based on local search. AAAI. 2007; Vol. 7: 1152-1157.

[29] Hutter F, Hoos HH, Leyton-Brown K, Stützle T. ParamILS: an automatic algorithm configuration framework. Journal of Artificial Intelligence Research. 2009; 36.1: 267-306.

[30] Pedregosa F. Hyperparameter optimization with approximate gradient. arXiv preprint arXiv. 2016; 1602.02355.

[31] Maclaurin D, Duvenaud D, Adams RP. Gradient-based hyperparameter optimization through reversible learning. Proceedings of the 32nd International Conference on Machine Learning. 2015.

[32] Snoek J, Larochelle H, Adams RP. Practical bayesian optimization of machine learning algorithms. Advances in neural information processing systems. 2012: 2951-2959.

[33] Hutter F. Scalable and Flexible Bayesian Optimization for Algorithm Configuration [Vortrag]. NIPS 2015 Workshop: Bayesian Optimization: Scalability and Flexibility. 2015. Verfügbar auf: https://www.youtube.com/watch?v=4DZUNumzTr0

[34] Team Clusteranalyse. Abstands- und Ähnlichkeitsmaße [Internet]. TUM;2009. [Stand Dez 2016]. Verfügbar auf: http://www-m9.ma.tum.de/material/felix-klein/clustering/Allgemeines/Abstandsmasse.php

[35] Gower JC. A general coefficient of similarity and some of its properties. Biometrics. 1971: 857-871.

[36] Kaufman L, Rousseeuw PJ. Finding groups in data: an introduction to cluster analysis. Vol. 344. Hoboken: John Wiley & Sons; 2009.

[37] Etzkorn B. Data Normalization and Standardization [Internet]. [Stand Jan 2017]. Verfügbar auf: http://www.benetzkorn.com/wp-content/uploads/2011/11/Data-Normalization-and-Standardization.pdf

[38] R Core Team. R: A language and environment for statistical computing [Software]. R Foundation for Statistical Computing, Vienna, Austria. 2016. Verfügbar auf: https://www.R-project.org/.

[39] Simm J, De Abril I, Sugiyama M. Tree-Based Ensemble Multi-Task Learning Method for Classification and Regression [R Paket]. 2014. Verfügbar auf: http://cran.r-project.org/package=extraTrees.

[40] Bischl B, Lang M, Kotthoff L, Schiffner J, Richter J, Studerus E, Jones Z, Casalicchio G. mlr: Machine Learning in R [R Paket]. JML. 2016; 17(170): 1-5 Verfügbar auf: https://cran.r-project.org/package=mlr

[41] Bischl B, Bossek J, Horn D, Lang M. mlrMBO: Model-Based Optimization for mlr [R Paket]. Verfügbar auf: https://github.com/mlr-org/mlrMBO

[42] Roustant O, Ginsbourger D, Deville Y. DiceKriging, DiceOptim: Two R Packages for the Analysis of Computer Experiments by Kriging-Based Metamodeling and Optimization [R Paket]. Journal of Statistical Software. 2012. Verfügbar auf: http://www.jstatsoft.org/v51/i01/.

[43] Liaw A, Wiener M. Classification and Regression by randomForest [R Paket]. R News. 2002. Verfügbar auf: https://cran.r-project.org/package=randomForest

[44] Meyer D, Dimitriadou E, Hornik K, Weingessel A, Leisch F. e1071: Misc Functions of the Department of Statistics [R Paket], Probability Theory Group (Formerly: E1071). TU Wien. 2015. Verfügbar auf: https://cran.r-project.org/package=e1071

[45] Meyer D, Buchta C. proxy: Distance and Similarity Measures [R Paket]. 2015. Verfügbar auf: https://cran.r-project.org/package=proxy

[46] Lang M, Surmann D. batchtools: Tools for Computation on Batch Systems [R Paket]. Verfügbar auf: https://cran.r-project.org/package=batchtools

[47] Bossek J. smoof: Single and Multi-Objective Optimization Test Functions [R Paket]. 2016. Verfügbar auf: https://cran.r-project.org/package=smoof

[48] Vanschoren J, Van Rijn JN, Bischl B, Torgo L. OpenML: networked science in machine learning. SIGKDD Explorations. 2013; 15.2: 49-60. Verfügbar auf: http://www.openml.org/

[49] Schork KU. Lokale Kriging-Verfahren zur Modellierung und Optimierung gemischter Parameterräume mit Abhängigkeitsstrukturen [Bachelorarbeit]. TU Dortmund. 2014.

[50] Chen YW, Lin CJ. Combining SVMs with various feature selection strategies. Springer Berlin Heidelberg. 2006; 207: 315-324.

[51] Stasinopoulos DM, Rigby RA. Generalized additive models for location scale and shape (GAMLSS) in R. Journal of Statistical Software. 2007; 23.7: 1-46.

[52] Mayr A, Fenske N, Hofner B, Kneib T, Schmid M. Generalized additive models for location, scale and shape for high dimensional data—a flexible approach based on boosting. Journal of the Royal Statistical Society: Series C (Applied Statistics). 2012; 61.3: 403-427.

[53] Forrester A, Sobester A, Keane A. Engineering design via surrogate modelling: a practical guide. John Wiley & Sons. 2008.

Eidesstattliche Erklärung

Ich versichere hiermit, dass ich die vorliegende Masterarbeit mit dem Thema

Alternative Infill-Kriterien für Random Forest in Sequential Model-Based Optimization

selbstständig verfasst und keine anderen Hilfsmittel als die angegebenen Quellen benutzt habe. Die Stellen, die anderen Werken dem Wortlaut oder dem Sinne nach entnommen sind, habe ich in jedem einzelnen Falle durch Angaben der Quelle oder Verweise kenntlich gemacht.

Die Arbeit wurde bisher keinem anderen Prüfungsamt in gleicher oder vergleichbarer Form vorgelegt und nicht veröffentlicht.

München, den