Applied Optimization Techniques: CMA-ES

Seminararbeit zur Vorlesung Applied Optimization Techniques

für die
Prüfung zum Bachelor of Science

an der Fakultät für Wirtschaft
im Studiengang Wirtschaftsinformatik
in der Studienrichtung Data Science

an der
DHBW Ravensburg

Verfasser:	Ibele Luisa, Janez Isabel,
	Steinwender Hanna, Romer Judith
Dozent:	Prof. Dr. Martin Zaefferer
Abgabedatum:	17.08.2022

Eigenständigkeitserklärung

Wir versichern hiermit, dass wir die vorliegende Projektarbeit mit dem Thema

Applied Optimization Techniques: CMA-ES

selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren haben wir an den entsprechenden Stellen innerhalb des Reports gekennzeichnet. Wir versichern zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Ort, Datum	Ibele Luisa
Ort, Datum	Janez Isabel
Ort, Datum	Steinwender Hanna
Ort, Datum	Romer Judith

Inhaltsverzeichnis

Abkürzungsverzeichnis

CMA ..... Covariance Matrix Adaptation

CMA-ES ..... Covariance Matrix Adaptation Evolution Strategy

DE ..... Differential Evolution

ES ..... Evolution Strategy

MLE ..... Maximum Likelihood Estimation

RMSE ..... Root Mean Square Error

Abbildungsverzeichnis

1 Einleitung

In den letzten Jahren haben sich für die Lösung verschiedener schwieriger Optimierungsprobleme in den Bereichen Wissenschaft und Technologie evolutionäre Algorithmen als praktischen Ansatz erwiesen. Evolutionäre Algorithmen sind stochastische Suchmethoden, die die Natur der natürlichen biologischen Evolution übernehmen und es einer Population von Organismen ermöglichen, sich zum Überleben an ihre Umgebung anzupassen. Einer der größten Nutzen evolutionärer Algorithmen liegt in ihrer Flexibilität und der Tatsache, dass sie wenig beziehungsweise kein Vorwissen bezüglich Differenzierbarkeit oder Stetigkeit der Zielfunktion erfordern. Die Idee, evolutionäre Prozesse auch am Computer umzusetzen beziehungsweise nachzuahmen, stammt aus dem Bereich der Optimierung. Optimierung gilt als eine Reihe von Aktionen, die durch kontinuierliches Lernen durchgeführt werden, um unter bestimmten Umständen das bestmögliche Ergebnis für ein Problem zu erzielen. Ein Optimierungsalgorithmus versucht also aus allen möglichen Lösungen für ein Problem die bestmögliche Lösung durch einen vordefinierten Ansatz zu finden. Evolutionäre Algorithmen sind dabei in der Lage, durch das Optimieren von Lösungskandidaten, welche aus realwertigen Parametern bestehen, solche Optimierungsprobleme zu lösen. Sie folgen dabei der Grundidee “Survival of the Fittest”, um immer bessere Annäherungen an eine optimale Lösung zu erzielen.

In der vorliegenden Ausarbeitung für die Vorlesung Applied Optimization Techniques soll einer der evolutionäre Algorithmen, die *Covariance Matrix Adaption Evolution Strategies* (CMA-ES), vorgestellt und zur Lösung eines Optimierungsproblems auf die gegebene Projektaufgabe angewendet und evaluiert werden.

Im Rahmen der gegebenen Projektaufgabe gilt es, ein Problem aus dem Bereich des Machine Learnings, der *Maximum Likelihood Estimation* (MLE) für Gaußsche Prozessmodelle, zu untersuchen und mithilfe eines Optimierungsalgorithmus, hier der CMA-ES, zu lösen. Die Lösung des verwendeten Algorithmus soll schlussendlich mit der gegebenen Default-Lösung verglichen werden.

Die vorliegende Ausarbeitung ist in insgesamt fünf Kapitel gegliedert. Nach der Einleitung soll sich das zweite Kapitel um die theoretischen Grundlagen zur Lösung der Aufgabenstellung drehen. Dabei soll zum Einen der verwendete Optimierungsalgorithmus CMA-ES an sich und die Default-Lösung, das heißt die Differential Evolution, vorgestellt werden. Zum

Anderen soll aber auch auf die gegebene Problemstellung aus dem Bereich des Machine Learnings der Maximum Likelihood Estimation für Gaußsche Prozessmodelle eingegangen werden. Im dritten Kapitel, der praktischen Umsetzung, soll die gegebene Problemstellung genauer erläutert sowie die Vorgehensweise der Modelloptimierung beschrieben werden. Auch soll das Evaluierungskonzept der Lösung vorgestellt werden. Darauffolgend soll im vierten Kapitel die empirische Untersuchung abgehandelt werden. Dies bedeutet, dass sowohl der CMA-ES als auch die Differential Evolution evaluiert werden sowie auf die Hyperparameteroptimierung durch GridSearch eingegangen werden soll. Dabei soll die Güte des Algorithmus auch bei veränderter Problemstellung betrachtet werden. Im letzten Kapitel sollen dann die zentralen Erkenntnisse der Projektaufgabe zusammengefasst und ein Fazit gezogen werden.

2 Theoretische Grundlagen

In diesem Kapitel soll der für das Projekt verwendete Algorithmus CMA-ES vorgestellt werden, wobei zum Einen auf die grundlegende Funktionsweise und zum Anderen auf die Einsatzmöglichkeiten des Algorithmus genauer eingegangen werden soll. Anschließend soll der für die in der Projektaufgabe gegebene Default-Algorithmus DE sowie die Problemstellung der Maximum Likelihood Estimation für Gaußsche Prozessmodelle beschrieben werden.

2.1 Covariance Matrix Adaptation Evolution Strategy

Der CMA-ES Algorithmus, welcher von N. Hansen und A. Ostermeier entwickelt wurde, ist ein moderner derivatefreier Suchalgorithmus für schwierige Optimierungsprobleme im kontinuierlichen Bereich (vgl. Hansen, 2016). Der Algorithmus gilt als ein State-of-the-art Werkzeug für kontinuierliche, evolutionäre Optimierung und ist besonders nützlich bei Black-Box-Optimierungsproblemen. Des Weiteren ist die CMA-ES definiert als ein stochastischer und vergleichsbasierter Suchalgorithmus, der die multivariate Normalverteilung als Stichprobenverteilung von Kandidatenlösungen beibehält (vgl. Akimoto und Hansen, 2020). Dabei ist der CMA-ES Algorithmus ein Ansatz zweiter Ordnung, der in einem iterativen Verfahren eine positiv definite Matrix, genauer gesagt eine Kovarianzmatrix, also auf konvex-quadratische Funktionen, schätzt . Dies macht das Verfahren

bei nicht separierbaren und/oder schlecht konditionierten Problemen durchführbar. Der CMA-ES approximiert, beziehungsweise approximiert, keine Gradienten und setzt die Existenz dieser nicht voraus, was das Verfahren bei nicht glatten und nicht kontinuierlichen Problemen sowie bei multimodalen und/oder verrauschten Problemen durchführbar macht. Der Algorithmus erweist sich als besonders zuverlässiger und höchst wettbewerbsfähiger evolutionärer Algorithmus für die lokale sowie für die globale Optimierung (vgl. Hansen und Ostermeier, 2001).

Dabei funktioniert die CMA-ES jedoch besser bei globalen Strukturen. Bei weniger globalen Strukturen hat sie eher Schwierigkeiten und erfordert mehrere Neustarts beziehungsweise größere Populationen, um nicht in lokalen Optima zu enden. Insgesamt lässt sich sagen, dass die CMA-ES gut geeignet ist für Optimierungsprobleme, die nicht konvex, nicht trennbar, schlecht konditioniert, multimodal und mit verrauschten Auswertungen sind (vgl. Dang u. a., 2019, S. 1).

Wie der Name des Algorithmus schon sagt, setzt die CMA-ES sich aus zwei Bestandteilen zusammen: zum Einen *Covariance Matrix Adaptation* (CMA) und zum Anderen *Evolution Strategy* (ES). Die CMA, beziehungsweise zu Deutsch die Anpassung einer Kovarianzmatrix, ist ein derandomisiertes Verfahren, um eine Kovarianzmatrix einer gegebenen normalverteilten Verteilung zu aktualisieren beziehungsweise wie der Name schon sagt zu adaptieren (vgl. Cansiz, 2021). Generell lässt sich eine Kovarianzmatrix definieren als eine quadratische Matrix, also eine Matrix mit gleich vielen Zeilen und Spalten. Dabei werden paarweise Abhängigkeiten zwischen Variablen in der Verteilung werden durch eine Kovarianzmatrix dargestellt, das heißt die Werte einer Kovarianzmatrix zeigen die Verteilungsgröße und -richtung multivariater Daten im mehrdimensionalen Raum (vgl. Hansen und Ostermeier, 2001, S. 10-11). Die Anpassung der Kovarianzmatrix läuft auf das Erlernen eines Modells zweiter Ordnung der zugrunde liegenden Zielfunktion hinaus. Im Gegensatz zu den meisten klassischen Methoden werden weniger Annahmen über die zugrunde liegende Zielfunktion getroffen. Da nur ein Ranking von Lösungskandidaten ausgenutzt wird, benötigt das Verfahren weder Ableitungen noch eine Zielfunktion.

Darüber hinaus gehört die CMA-ES zu den *Evolutionsstrategien* (ES). Die CMA-ES gilt anerkanntermaßen als der beliebteste und effizienteste Evolutionsstrategie-Algorithmus. ES stellen stochastische, ableitungsfreie Methoden zur numerischen Optimierung von nichtlinearen oder nicht-konvexen kontinuierlichen Optimierungsproblemen dar. Generell

erzielen diese bei vielen Benchmarks und realen Anwendungen eine hochmoderne Leistung, wobei sie typischerweise eine Gaußsche Verteilung entwickeln, um sich dem Optimum anzunähern (vgl. Li u. a., 2020). Allgemein definieren ES eine Klasse evolutionärer Algorithmen für die Black-Box-Optimierung, welche grundlegend auf dem Prinzip der biologischen Evolution basieren. Dies bedeutet, dass sie nach den sich wiederholenden Hauptschritten Rekombination, Mutation und Selektion arbeiten. In jeder Wiederholung, genauer gesagt jeder Generation, entstehen durch die Variation der aktuellen Elternindividuen neue Individuen, welche die Lösungskandidaten darstellen. Die Lösungskandidaten werden gemäß einer Normalverteilung in $\mathbb{R}$ abgetastet. Aus diesen werden wiederum einige basierend auf ihrer Fitness oder mathematisch gesehen auf ihrem objektiven Funktionswert f(x) ausgewählt (Selektion), um Eltern in der nächsten Generation zu werden. So werden im Laufe der Generationsfolge Individuen mit immer besserer Fitness beziehungsweise immer besseren f-Werten generiert. Ziel der Rekombination ist es, einen neuen Mittelwert für die Verteilung auszuwählen. Der Schritt der Mutation hingegen läuft darauf hinaus, einen zufälligen Vektor hinzuzufügen, eine Störung mit einem Mittelwert von Null. Aufgrund der Zugehörigkeit zu den ES arbeitet auch die CMA-ES nach den Schritten der Selektion, Rekombination und Mutation. Die genaue Funktionsweise soll jedoch erst später beschrieben werden.

Generell lässt sich über die CMA-ES sagen, dass sie fast parameterlos ist, was bedeutet, dass nur ein Hyperparameter, nämlich die Populationsgröße λ, vorgeschlagen wird, um vom Benutzer angepasst zu werden. Das bedeutet, dass die CMA-ES für ihre Anwendung keine langwierige Parameterabstimmung benötigt, denn tatsächlich wird die Wahl der internen Parameter der Strategie nicht dem Benutzer überlassen. Das Finden guter (Standard-)Strategieparameter wird als Teil des Algorithmus-Designs und nicht als Teil seiner Anwendung betrachtet – das Ziel ist es, einen gut funktionierenden Algorithmus so zu haben, so wie er ist. Die Standardpopulationsgröße λ ist vergleichsweise klein, um eine schnelle Konvergenz zu ermöglichen. Neustarts mit zunehmender Populationsgröße verbessern die globale Suchleistung. Es lässt sich also sagen, dass ein wichtiger praktischer Vorteil von CMA-ES darin besteht, dass alle Hyperparameter des Algorithmus standardmäßig in Bezug auf die Problemdimension n definiert sind. Die CMA-ES ist ein Monte-Carlo-Verfahren zur Optimierung von Funktionen (vgl. Krause u. a., 2016, S. 1). Die Anwendung des CMA-ES erfordert dabei explizit einen endlich-dimensionalen Such-

raum, auf dem Lösungskandidaten leben (vgl. Dang u. a., 2019, S. 1). Dabei geht man von einem Suchszenario aus, in welchem eine Zielfunktion f minimiert werden soll:

$$f : \mathbb{R}^n \to \mathbb{R}$$

Die einzige zugängliche Information über f sind die Funktionswerte ausgewerteter Suchpunkte. Es ist nicht davon auszugehen, dass besondere Kenntnisse über die Struktur von f vorliegen, weshalb f als eine Blackbox betrachtet wird (vgl. Hansen, 2014). Das Leistungsmaß ist die Anzahl der Funktionsauswertungen, die benötigt werden, um einen bestimmten Funktionswert zu erreichen. Dabei verwenden viele evolutionäre Algorithmen für kontinuierliche Domänen eine Normalverteilung, um neue Suchpunkte abzutasten (vgl. Hansen, 2006).

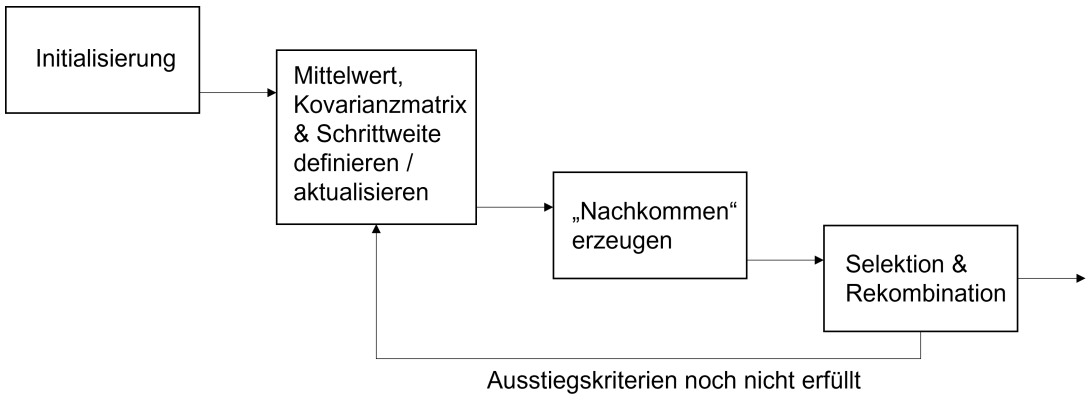


Abbildung 2.1: Die Phasen der CMA-ES (vgl. Gagganapalli, 2015)

Aufgrund seiner Zugehörigkeiten zu den Evolutionsstrategien, arbeitet der CMA-ES auch innerhalb der drei Hauptphasen Rekombination, Mutation und Selektion (siehe Abb. 2.1 p.5), durch welche er immer wieder durchiteriert.
Bei der CMA-ES beginnt die Suche nach den globalen Optima mit dem Abtasten einer multivariaten Normalverteilung um den Mittelwert M bei jeder Generation, wodurch eine Population neuer Suchpunkte generiert wird. Die zu Grunde liegende Gleichung des “Abtastens” der Suchpunkte für die Generationsnummer g = 0, 1, 2, . . . lässt sich schreiben als (vgl. Hansen, 2006):

$$x_k^{(g+1)} \sim N(m^{(g)}, (\sigma^{(g)})^2\, C^{(g)}) \text{ für } k = 1, \ldots, \lambda$$

Dabei bezeichnet $\sim$ die gleiche Verteilung auf der linken und rechten Seite.
$N(m^{(g)}, (\sigma^{(g)})^2 \sim m^{(g)} + \sigma^{(g)} N(0, C^{(g)}) \sim m^{(g)} + \sigma^{(g)} B^{(g)} D^{(g)} N(0, I)$ steht für die multivariate normale Suchverteilung und $x^{(g+1)}k \in \mathbb{R}^n$ für den k-ten Suchpunkt der Generation

$g+1.m^{(g)} \in \mathbb{R}^n$ stellt den Mittelwert der Suchverteilung der Generation g dar, $\sigma^{(g)} \in \mathbb{R}+$ die Gesamt-Standardabweichung beziehungsweise die Schrittweite bei der Generation g, $C^{(g)} \in \mathbb{R}^{nxn}$ die Kovarianzmatrix der Generation g und $\lambda \geq 2$ die Populationsgröße beziehungsweise die Stichprobengröße dar.
Um zur nächsten Generation (g+2) zu kommen, müssen die Parameter $m^{(g+1)}, C^{(g+1)}, \sigma^{(g+1)}$ berechnet werden. Dabei stellt jeder dieser Parameter einen Schritt im Algorithmus dar (vgl. Gagganapalli, 2015, S. 6).
In Folge der Selektion und Rekombination wird der Mittelwert ausgewählt. Der neue Mittelwert $m^{(g+1)}$ definiert einen gewichteten Durchschnitt von μ ausgewählten Punkten aus der Stichprobe $x_1(g+1), \ldots .., x_\lambda(g+1)$:

$$m^{(g+1)} = \sum_{i=1}^{\mu} w_i \, x_{i:\lambda}^{(g+1)} \text{ mit der Nebenbedingung:}$$

$$\sum_{i=1}^{\mu} w_i = 1 \text{ mit } w_i > 0 \text{ für } i = 1, \ldots, \mu$$

Dabei stellt $\mu \leq \lambda$ die Populationsgröße, also die Anzahl an ausgesuchten Punkten dar. $w_i = 1 \ldots \mu \in \mathbb{R}+$ steht für die positive Gewichtskoeffizienten für die Rekombination mit $w_1 \geq w_2 \geq \ldots \geq w\mu > 0$. Dabei berechnet $w_i = 1/\mu$ den Mittelwert von μ ausgewählten Punkten. Und $x_{i:\lambda}^{(g+1)}$ das i-te beste Individuum aus $x_1^{(g+1=}, \ldots, x_\lambda^{(g+1)}$ definiert.
Diese Gleichung implementiert die Selektion durch Auswählen von $\mu < \lambda$ aus λ Nachkommenspunkten. Auch gehört zur Selektion das Zuweisen von unterschiedlicher Gewichte w_i (vgl. Hansen, 2006, S. 80). Außerdem wird in dieser Gleichung die Rekombination implementiert, indem eine gewichtete Summe von μ Individuen für einen gewichteten Durchschnitt berücksichtigt werden (vgl. Hansen, 2006, S. 80). Dabei wird der evolutionäre Prozess der Selektion und Rekombination erreicht, indem der mittlere Vektor für jede Generation berechnet und dieser dann mutiert wird, um die Nachkommen der Elterngeneration zu erzeugen. Der mittlere Vektor $m^{(g+1)}$ für die Generation g stellt den gewichteten Durchschnitt der μ ausgewählten besten Individuen in der Reihenfolge des Fitnessrankings auf der Zielfunktion $x_k^{(g+1)}$ für $k = 0, 1, 2, \ldots, \mu$ dar. Allgemein kann der gewichtete Vektor ein gleichgewichteter oder ein linearer Vektor sein. Bei einem gleichgewichteten Vektor haben alle ausgewählten Stichproben den gleichen Anteil am resultierenden Mittelwert ($W_i = 1/\lambda$). Bei der Linear Weight-Vektorkonfiguration W_i hat der fitteste Punkt

einen höheren Anteil an Genen als der Punkt mit der geringeren Fitness (vgl. Gagganapalli, 2015, S. 7).

Nachdem die Phasen wie in Abbildung 2.1 durch den Algorithmus erfolgt sind, geht es darum die Kovarianzmatrix C anzupassen. Die Kovarianz-Matrix-Anpassung bestimmt die unterschiedliche Mutation für die Nachkommens-Population im Evolutionsprozess. Die Nachkommen einer Generation werden gemäß der multivariaten Normalverteilung in $\mathbb{R}^n$ abgetastet, während die Rekombination auf die Auswahl eines neuen Mittelwerts bei Generation g+1 hinausläuft, zielt die Mutation auf die Abtastung der Normalverteilung der Kovarianzmatrix multipliziert mit der Schrittgröße ab. Wie bereits erklärt, werden die Abhängigkeiten zwischen den Variablen in der Verteilung durch die Kovarianzmatrix dargestellt. Der Mittelwert der Verteilung wird nun so aktualisiert, dass die Wahrscheinlichkeit der zuvor erfolgreichen Kandidatenlösung maximiert wird, und die Kovarianzmatrix wird so adaptiert, dass die Wahrscheinlichkeit von zuvor erfolgreichen Suchschritten erhöht wird (vgl. Gagganapalli, 2015, S. 7). Das heißt, die Verteilungsparameter wie der Mittelwertvektor und die Kovarianzmatrix werden bei jeder Iteration basierend auf den Kandidatenlösungen und ihrer objektiven Wertrangfolge aktualisiert, sodass die Stichprobenverteilung in der nächsten Iteration mit größerer Wahrscheinlichkeit vielversprechende Kandidatenlösungen hervorbringen wird (vgl. Akimoto und Hansen, 2020).

Jedoch bestimmt die Anpassung der Kovarianzmatrix nicht die Gesamtskala der Verteilung, das heißt der Schrittweite des Algorithmus. Zur Steuerung der Schrittweite $\sigma_{(g}$ wird der sogenannte Evolutionspfad genutzt. Dieser kann unabhängig von der Anpassung der Kovarianzmatrix angewendet werden und wird als kumulative Schrittlängenanpassung bezeichnet. Immer wenn der Evolutionsweg kurz ist, heben sich Einzelschritte gegenseitig auf. In diesem Fall sollte die Schrittweite verringert werden. Hingegen immer wenn der Evolutionspfad lang ist und einzelne Schritte in die gleiche Richtung zeigen, sollte die Schrittweite erhöht werden, um die Anzahl der Schritte zu reduzieren. Um zu entscheiden, ob der Evolutionspfad lang oder kurz ist, wird die Pfadlänge mit der erwarteten Länge unter zufälliger Auswahl verglichen. Wenn der Auswahlpfad den Evolutionspfad länger als erwartet ausfällt, wird σ erhöht, und wenn der Auswahlpfad den Evolutionspfad kürzer als erwartet ausfällt, wird σ verringert (vgl. Gagganapalli, 2015, S. 8). Die Schrittweite $\sigma_{(g}$ bestimmt die Gesamtvarianz der Mutation bei Generation g. Die variable Eigenschaft der Schrittweite σ bei jeder Generation spielt eine entscheidende Rolle bei der Kontrolle

der vorzeitigen Konvergenz und der engen Konvergenz zu globalen Optima.
Generell lässt sich die Funktionsweise der CMA-ES also wie folgt beschreiben: der Algorithmus bildet eine parametrische Verteilung über den gegebenen Lösungsraum und tastet dabei iterativ eine Population von Lösungskandidaten aus einer parametrisierten Suchverteilung ab. Diese Lösungskandidaten werden dann durch eine Black-Box-Funktion bewertet. Die Tupel der Kandidatenbewertung bilden einen Datensatz, damit CMA-ES die Suchverteilung, d. h. ihren Mittelwert und ihre Kovarianzmatrix, aktualisieren kann.
Zu den Einsatzmöglichkeiten der CMA-ES lässt sich sagen, dass sie sich generell in den unterschiedlichsten Bereichen einsetzen lässt. In vielen Bereichen, z.B. der Robotik, wird ein Optimierungsziel oft als Kostenfunktion einer anderen parametrischen Lösungsfunktion definiert. Beispielsweise kann es sich um eine Gesamtkostenfunktion handeln, für Anwendungen im Robot Skill Learning, der Policy Search oder eine Verlustfunktion im Kontext der inversen optimalen Steuerung. Auch kann der CMA-ES für Skill Learning via Reinforcement Learning oder Inverse Reinforcement Learning eingesetzt werden (vgl. Dang u. a., 2019).

2.2 Differential Evolution

Die DE ist eine weit verbreitete Optimierungstechnik, welche die natürliche Evolution simuliert und bekannt für eine beständige Leistung in einem breiten Spektrum von Anwendungen mit nur wenigen Hyperparametern ist. Zudem zeichnet sich die DE durch einen einfachen algorithmischen Rahmen aus, welcher relativ einfach zu implementieren ist (vgl. Kumar u. a., 2022, S. 2).
DE ist wohl einer der vielseitigsten und stabilsten populationsbasierten Suchalgorithmen, der sich durch Robustheit gegenüber multimodalen Problemen auszeichnet. Darüber hinaus gibt es verschiedene Varianten des DE, wobei im Folgenden der Standard DE betrachtet werden soll (vgl. Georgioudakis und Plevris, 2020, S. 1).
Es handelt sich um keine deterministische Technik, da nicht nur eine Berechnung durchlaufen werden kann. Die Technik basiert auf der Evolution einer Vektorpopulation von realen Werten, welche die Lösungen im Suchraum darstellen. Dabei wird die Population an Lösungsvorschlägen iterativ so verändert, dass sie zu einem Optimum ihrer Funktion konvergiert. Für die Leistung eines solchen Algorithmus spielt die Wahl der Kontrollparameter eine entscheidende Rolle. Weiterhin ist die DE in 4 Schritte untergliedert, nämlich

die Initialisierung, Mutation, Kreuzung und Selektion. Bevor die Grundgesamtheit initialisiert werden kann, müssen die Ober- und Untergrenzen festgelegt werden. Sind diese festgelegt weist ein Zahlengenerator jedem Parameter in jedem Vektor zufällig einen Wert innerhalb dieses Bereiches zu. Bei der differentiellen Mutation wird die Zufallsstichprobe verwendet, um ausgewählte Vektoren zu kombinieren und einen Mutationsvektor zu erzeugen. Zusätzlich zur differentiellen Mutation wird eine einheitliche Kreuzung verwendet. Hierbei wird jeder Vektor mit einem Mutationsvektor gekreuzt. Im letzten Schritt muss selektiert werden. Dabei wird untersucht, ob der Testvektor einen Wert der Zielfunktion hat, der gleich oder kleiner als sein Zielvektor ist. Ist dies der Fall so ersetzt er den Zielvektor in der nächsten Generation. Andernfalls behält der Zielvektor seinen Platz in der Population für mindestens eine weitere Generation. Der Prozess der Mutation, Rekombination und Selektion wird so lange wiederholt, bis das Optimum gefunden ist oder ein vorgegebenes Beendigungskriterium erfüllt ist (vgl. Castillo, 2021, S. 9-11).
Der Hauptvorteil von Standard-DE ist die Tatsache, dass es nur drei Kontrollparameter gibt, die man anpassen muss. Die Leistung von DE bei einem bestimmten Optimierungsproblem hängt in hohem Maße sowohl vom Schema der Versuchsvektorgenerierung als auch von der Wahl der Kontrollparameter ab. Um gute Optimierungsergebnisse zu erzielen, muss man zunächst das Schema zur Erzeugung von Versuchsvektoren auswählen und dann die Kontrollparameter für das Optimierungsproblem anpassen. Die richtigen Werte für die Kontrollparameter zu finden, ist nicht immer einfach und kann vor allem bei schwierigen Problemen zeitaufwändig und kompliziert werden (vgl. Georgioudakis und Plevris, 2020, S. 2).

2.3 Maximum Likelihood Estimation für Gaußsche Prozessmodelle

Das Prinzip der MLE dient der Schätzung der Parameter einer angenommenen Wahrscheinlichkeitsverteilung bei bestimmten beobachteten Daten. Dazu wird die Likelihood-Funktion maximiert, sodass die beobachteten Daten unter dem angenommenen Modell am wahrscheinlichsten sind. Der Punkt, an dem die Likelihood-Funktion maximiert wird, wird als MLE bezeichnet (vgl. Myung, 2003, S. 93).
Die Maximum-Likelihood-Methode ist die gängigste Methode zur Schätzung des Parame-

ters, der eine Wahrscheinlichkeitsfunktion einer diskreten stochastischen Variablen X auf der Grundlage der Beobachtungen $x1; x2; ...; xn$, die unabhängig voneinander aus der Verteilung entnommen wurden. Die Maximum-Likelihood-Schätzung ist der Wert , welcher die Likelihood-Funktion maximiert, die definiert ist durch

$$L(\Theta) = \Pi_{i=1}^{n} P(X = x_i|\Theta) = P(X = x_1|\Theta)P(X = x_2|\Theta)...P(X = x_n|\Theta)$$

wenn X eine diskrete stochastische Variable ist und

$$L(\Theta) = \Pi_{i=1}^{n} p(x_i|\Theta) = P(x_1|\Theta)P(x_2|\Theta)...P(x_n|\Theta)$$

wenn X eine kontinuierliche stochastische Variable ist. Das heißt, die Maximum-Likelihood-Schätzung wählt den Modellparameter, der die beobachteten Daten am wahrscheinlichsten erzeugt (vgl. MIURA, 2011, S. 156).

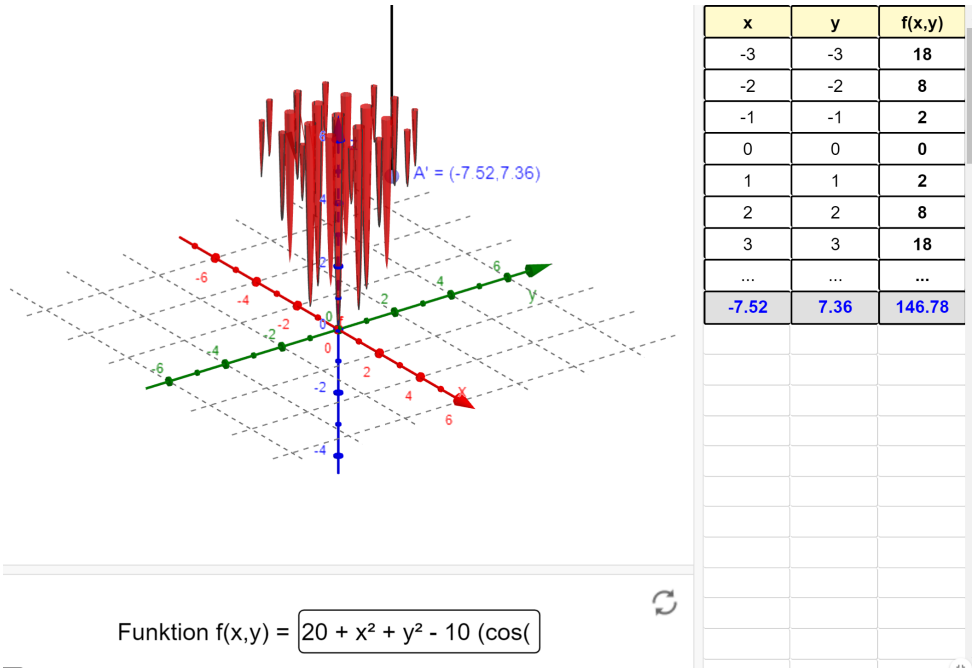
3 Praktische Umsetzung

Für die Umsetzung der Problemstellung wird ein Modell mit buildKriging, einem Gaußschen Prozessmodell, trainiert. Diesem Modell wird ein Optimierer übergeben. In dieser Arbeit wird der DE auch als Default bezeichnet, da dieser als Standardwert des Parameters eingestellt ist. Hierbei wird der Default mit dem CMA-ES verglichen. Der Fokus der Optimierung liegt auf den Thetas, die so gesetzt werden sollen, dass die MLE minimiert wird.

3.1 Problemstellung

Ziel ist es, mit vorhandenen Daten, bei denen sowohl die beiden Unbekannten x als auch die abhängige Variable y bekannt sind, ein Modell zu trainieren. Die Daten, mit denen das Modell trainiert wird, werden zu Beginn in einem ersten Schritt automatisch erstellt. Hierfür werden eine Ober- und Untergrenze für die beiden Unbekannten festgelegt und jeweils 70 zufällige Werte erzeugt. Um die y-Werte zu erhalten, werden die Unbekannten an die Funktion übergeben. Da die Funktion in der Realität häufig jedoch unbekannt und damit eine Black-Box ist, wird sie nur zur Erzeugung der y-Werte genutzt. Dadurch, dass die y-Werte mit der Funktion erzeugt werden, sind die Daten ohne Rauschen und Fehler des Modells liegen nicht am Rauschen der Daten, sondern allein bei den angewendeten Modellen. Die Funktion sieht folgendermaßen aus:

$$f(x) = 20 + x_1^2 + x_2^2 - 10(cos(2 * x_1) + cos(2 * x_2))$$



x	y	f(x,y)
-3	-3	18
-2	-2	8
-1	-1	2
0	0	0
1	1	2
2	2	8
3	3	18
...	...	...
-7.52	7.36	146.78

Abbildung 3.1: Visualisierung der Funktion

Da es sich um einen zweidimensionalen Raum handelt, kann die Funktion (siehe Abb. 4.10 p.20) auch ohne Probleme grafisch dargestellt werden (Anmerkung: Seite arbeitet nicht mit x_1 und x_2, sondern mit x und y; Formel: $20 + x^2 + y^2 - 10(cos(2 * x) + cos(2 * y))$):

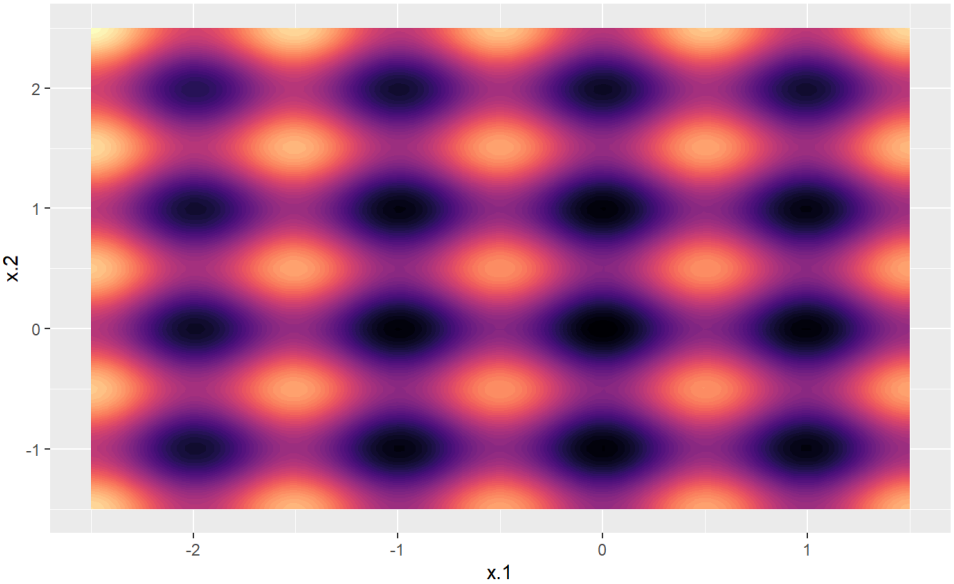


Abbildung 3.2: Visualisierung der Funktion aus der Vogelperspektive

Es fällt direkt auf: es gibt nur positive y-Werte, ein globales Minimum und zahlreiche

lokale Minima. Wichtig zu beachten ist, dass die Ober- und Untergrenzen der erzeugten Daten folgendermaßen begrenzt sind:

$$x1[-2,5;1,5]$$

$$x2[-1,5;2,5]$$

Mit Blick aus der Vogelperspektive sehen die Daten folgendermaßen visualisiert aus (siehe Abb. 3.2 p.11): Das Modell, mit dem trainiert wird, ist buildKriging von $\mathbb{R}$ (vgl. bartzbeielstein, 13.06.2020). Dieses basiert auf dem von Forrester et al. (vgl. Forrester u. a., 2008, S. 49-59) beschriebenen Code. Forrester et al. wiederrum nehmen die Ideen des Kriging von Danie G. Krige. Es ist dem Gaußschen Prozessmodell sehr ähnlich. Der Unterschied besteht darin, dass es beim Kriging einen Vektor aus Theta gibt und der Exponent p nicht fix bei zwei liegt. Bei der Funktion buildKriging wird p jedoch standardmäßig auf zwei gesetzt. Die Anzahl der Theta wird geschätzt anhand der Unbekannten - in unserem Beispiel gibt es zwei Unbekannte x und damit auch zwei Theta. Dieses Theta ist für jede Unbekannte x eine Konstante, wodurch es zu einem Gaußschen Prozessmodell wird.
Für die vorhandenen Werte werden eine Korrelationsmatrix und eine Kovarianzmatrix erzeugt. Durch die Bildung von Korrelationen wird die Annahme getroffen, dass die Funktion glatt und kontinuierlich ist. Die Höhe von Theta gibt Aufschluss über die Höhe der Korrelation, also wie viel Einfluss eine Unbekannte x auf das Zielmerkmal y hat.
Das Modell nutzt einen Gaußschen Kernel. Da die Annahme getroffen wurde, dass die Funktion bestimmte Eigenschaften wie Stetigkeit und Symmetrie aufweist, dann gilt Mercer's Theorem und das Skalarprodukt aus den Vektoren Theta und Theta transponiert kann durch den Gaußschen Kernel ersetzt werden. Das macht die Berechnung einfacher und schneller (vgl. Géron, 2020, S. 172). Um die bestmögliche Lösung zu erhalten, soll Theta optimiert werden. BuildKriging nutzt hierfür standardmäßig Differential Evolution.

3.2 Evaluierungskonzept

Für die Problemstellung wird das entwickelte Evaluierungskonzept angewandt. Dies unterteilt sich in drei Phasen (siehe Abb. 3.3 p.13).

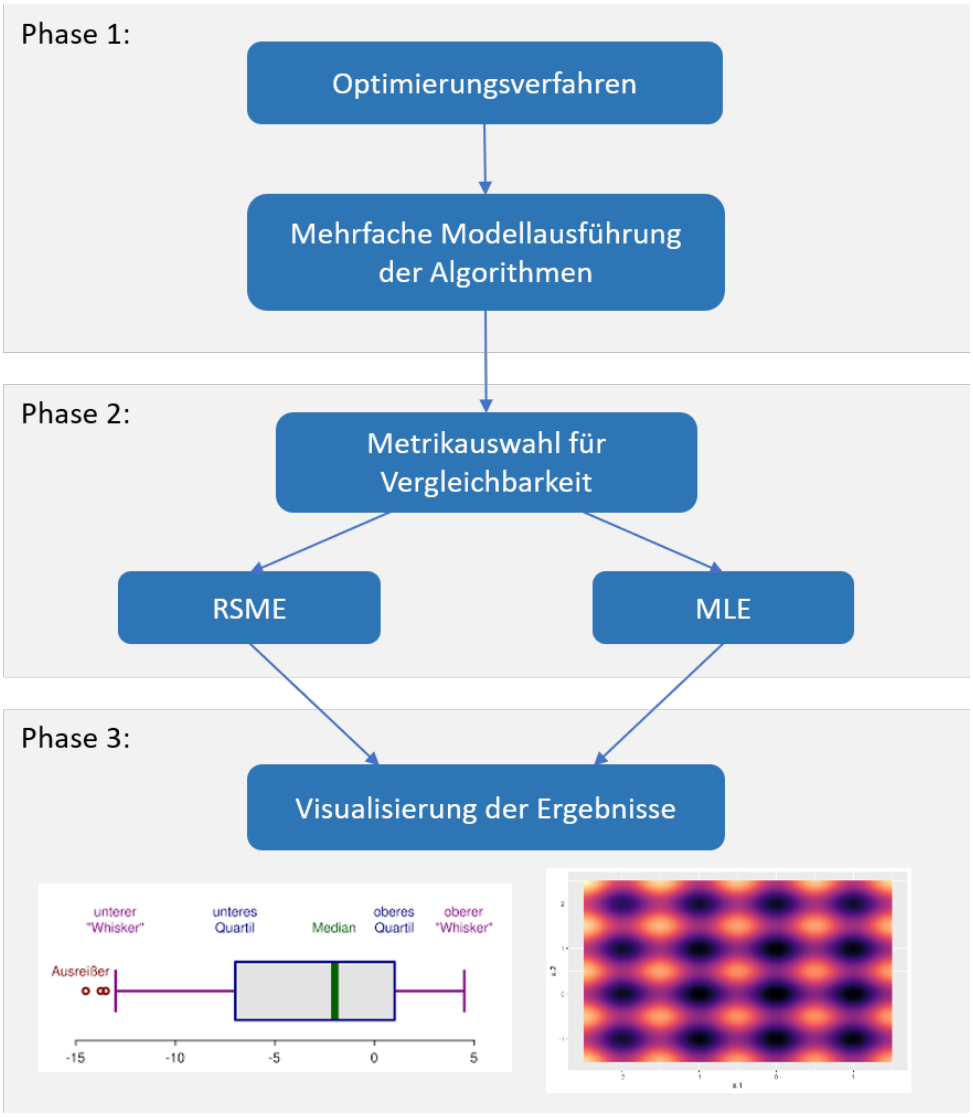


Abbildung 3.3: Evaluierungskonzept

In der ersten Phase erfolgt die Wahl der Optimierungsverfahrens. Dabei handelt es sich in der vorliegenden Aufgabenstellung um ein randomisiertes Optimierungsverfahren. Bei einem one-max-problem wäre es möglich, den fitness value für alle möglichen State Vectors, x, zu berechnen und dann den besten davon auszuwählen. Bei komplizierten Problemen ist dies jedoch nicht immer innerhalb einer angemessenen Zeitspanne möglich. Randomisierte Optimierungsalgorithmen beginnen in der Regel mit einem anfänglichen besten State Vektor, oder einer Population von mehreren Zustandsvektoren, und generieren dann nach dem Zufallsprinzip einen neuen Zustandsvektor.

Ein randomisierter Optimierungsalgorithmus ist ein Algorithmus, der versucht, die optimale Lösung für ein gegebenes Optimierungsproblem zu finden, indem er mit einem vorgegebenen State Vector beginnt und versucht, einen besseren zu finden. Dieser Vor-

gang wird so lange wiederholt, bis kein geeigneter Zustandsvektor mehr zur Auswahl steht oder nach einer bestimmten Anzahl von Versuchen kein neuer State Vector gefunden werden kann.

Anstatt immer den idealen Zustand zu erreichen, wird bei der Optimierung versucht, eine gute Lösung für ein Problem zu finden. Die Qualität der Lösung, die schließlich gefunden wird, steht in einem Spannungsverhältnis zu der Zeit, die für die Suche nach der besten Lösung aufgewendet wird. Die Methode wird jedoch letztendlich zu einem guten Gesamtergebnis führen, wenn eine beträchtliche Anzahl von Anstrengungen unternommen wird, um eine solide Lösung zu finden. Aufgrund der Wahl des randomisierten Optimierungsverfahrens ist es notwendig eine mehrfache Modellausführung der Algorithmen durchzuführen (vgl. Phd, 13.01.2019)

In der zweiten Phase befasst sich mit der Metrikauswahl für die Vergleichbarkeit der Algorithmen. In diesem Fall fiel die Wahl auf den Wurzel-Mittelwert-Fehler (RMSE) und den MLE. Eine der am häufigsten verwendeten Methoden zur Bewertung der Genauigkeit von Prognosen ist der mittlere quadratische Fehler, auch bekannt als mittlere quadratische Abweichung. Er veranschaulicht den euklidischen Abstand zwischen gemessenen wahren Werten und Prognosen.

Zu berechnen ist das Residuum (Differenz zwischen Vorhersage und Wahrheit) für jeden Datenpunkt zusammen mit seiner Norm, seinem Mittelwert und seiner Quadratwurzel, um den RMSE zu bestimmen. Da der RMSE reale Messungen an jedem projizierten Datenpunkt erfordert und verwendet, wird er häufig in Anwendungen des überwachten Lernens eingesetzt. Die Formel setzt sich wie folgt zusammen:

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\left(\frac{d_i - f_i}{\sigma_i}\right)^2}$$

Bei der Bewertung der Leistung eines Modells im Bereich des maschinellen Lernens, sei es während des Trainings, der Kreuzvalidierung oder der Überwachung nach dem Einsatz, ist es sehr vorteilhaft, eine einzige Zahl zu haben. Eine der am häufigsten verwendeten Metriken hierfür ist der Root Mean Square Error. Es ist eine geeignete Bewertungsmethode, die einfach zu verstehen ist und mit einigen der am häufigsten verwendeten statistischen Annahmen übereinstimmt (vgl. C3 AI, 28.09.2021).

Die gängigste Methode für statistische Schlussfolgerungen ist die MLE, die viele positive statistische Eigenschaften wie Konsistenz und asymptotische Normalität aufweist. Die

MLE-basierte Analyse und die GLSE-basierte Inferenz sind für das gemeinsame univariate/multivariate Regressionsmodell vollkommen vergleichbar (vgl. Pan und Fang, 2002, S.4).
In der dritten und letzten Phase geht es um die Visualisierung der Ergebnisse. Dies erfolgt mit Hilfe einer Boxplot-Darstellung und einer grafischen Visualisierung. Um Erkenntnisse über die Variabilität oder Streuung der Daten zu gewinnen, ist ein Boxplot sehr nützlich. Es handelt sich um ein Diagramm, das die Verteilung der Werte in den Daten anschaulich darstellt. Boxplots haben den Vorteil, dass sie weniger Platz beanspruchen als Histogramme oder Density Plots, was beim Vergleich von Verteilungen über mehrere Gruppen oder Datensätze von Vorteil ist.
Die Verteilung von Daten wird mithilfe von Boxplots dargestellt. Dabei handelt es sich um standardisierte Datenvisualisierungstechniken, die auf einer fünfstelligen Zusammenfassung basieren ("Minimum", erstes Quartil, Median, drittes Quartil und "Maximum") (vgl. Galarnyk, 12.09.2018).

4 Empirische Untersuchung

Im folgenden Kapitel soll der CMA-ES mit dem DE anhand des bereits vorgestellten Evaluierungskonzeptes verglichen werden. Dabei sollen in einem ersten Schritt beide Optimierer grundlegend evaluiert werden. Im zweiten Unterkapitel werden die Parameter des CMA-ES optimiert um herauszufinden, ob das Modell so verbessert werden kann. Im dritten und letzten Unterkapitel werden zum einen die Daten als auch die Funktion, aus der die Daten stammen, verändert um so zu untersuchen, welche Unterschieden zwischen den beiden Optimierern bestehen.

4.1 Evaluation der beiden Algorithmen

Die Evaluation folgt nach dem in Kapitel 2.3 vorgestellten Evaluationskonzept. Da mit weiterführenden Versuchen, bei denen die Datenmenge und die Parameter der Optimierer angepasst werden, die Modelle und Optimierer teils sehr lange benötigen, wird die Anzahl der Iterationen auf 50 beschränkt. Dadurch ergeben sich viele genug Durchläufe, um Schwankungen abzufangen, aber wenig genug, dass die Dauer der Durchführung innerhalb weniger Stunden bleibt.

Für die Ermittlung des RMSE werden 50 neue Datenpunkte innerhalb der Ober- und Untergrenze erzeugt. Die Funktionswerte an diesen Stellen können ohne Probleme berechnet werden, da uns die Funktion in diesem Fall zur Verfügung stehen. In realen Anwendungsfällen würde ein Teil der Beobachtungen als Testdaten bei Seite gelegt und später anhand der für das Training nicht genutzten Beobachtungen ein RMSE berechnet werden. Durch die Berechnung der Funktionswerte haben wir in diesem Projekt den Vorteil, dass die Daten ohne Rauschen bzw. Fehler erzeugt werden. Für die Berechnung des RMSE müssen dann nur noch die Datenpunkte an die Modelle übergeben und die Werte vorhergesagt werden.

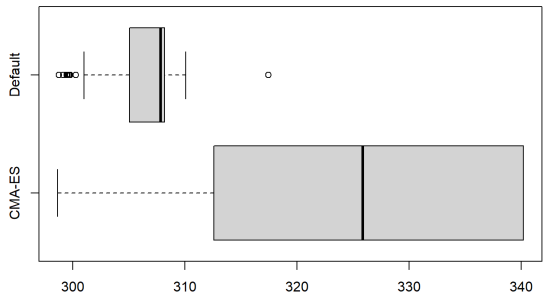


Abbildung 4.1: MLE

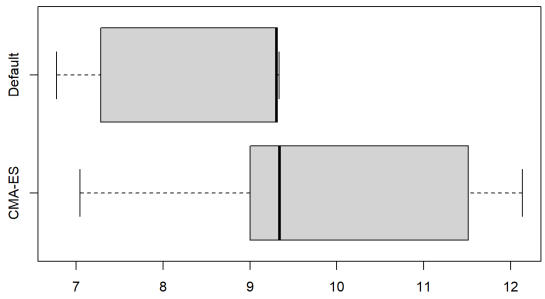


Abbildung 4.2: RMSE

Zuerst werden die Defaultlösung, der DE, und der CMA-ES ohne Anpassung von Parametern oder Änderung der Datengrundlage evaluiert. Bei der MLE in der obigen Abbildung (siehe Abb. 4.1 p.16) ist zu erkennen, dass sich die Werte beim CMA-ES deutlich unterscheiden von Durchführung zu Durchführung. Der DE hingegen ist deutlich stabiler (Begründung einfügen). Das Modell mit der niedrigsten MLE ist ein mit CMA-ES optimiertes. Insgesamt hat der Default im Durchschnitt eine deutliche bessere MLE. Für die Anwendung der Modelle ist jedoch der in zweiten Abbildung (siehe Abb. 4.2 p.16) dargestellte RMSE noch interessanter. Auch hier ist der DE stabiler und die Hälfe alle mit CMA-ES optimierten Modelle ist schlechter als der Default. Außerdem ist das Modell mit dem niedrigsten RMSE ein mit dem Default optimiertes Modell.

Aus dieser ersten Evaluation geht schon hervor, was in der Theorie bereits beschrieben wurde - es sind randomisierte Verfahren, die nicht immer das beste Ergebnis liefern.

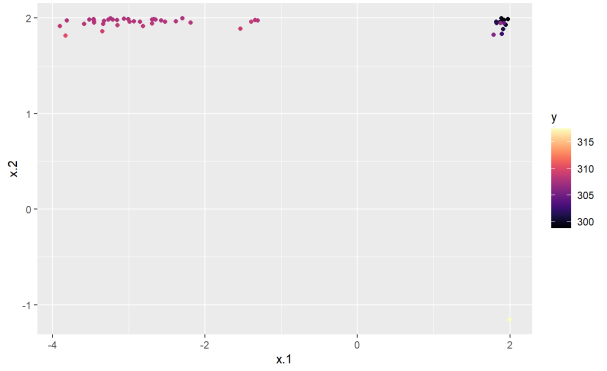


Abbildung 4.3: Default Lösung

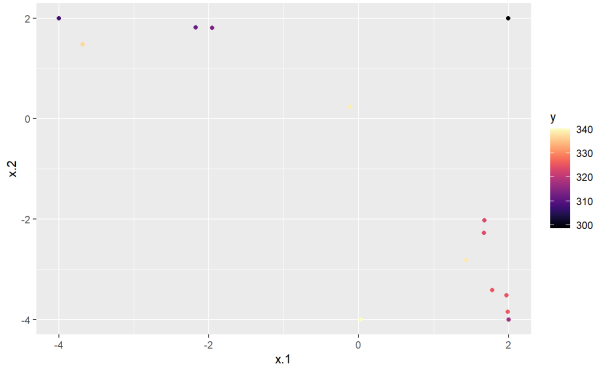


Abbildung 4.4: CMA-ES Lösung

Neben der reinen MLE im Vergleich der beiden Algorithmen, kann die MLE auch in Verbindung mit den zu optimierenden Werten, den Theta, betrachtet werden. Bei 50 Iterationen landet der CMA-ES, wie in der vorherigen Abbildung (siehe Abb. 4.4 p.17) zu erkennen, immer wieder bei den gleichen Werten, während sich die Werte beim Default (siehe Abb. 4.3 p.17), wenn auch teils nur gering, unterscheiden. Aus den beiden Grafiken geht jedoch hervor, dass für die beiden Unbekannten jeweils ein Wert von nahe zwei optimal ist. Interessant zu sehen ist, dass der Default sich bei jeder Iteration bei einem Wert von nahe zwei für x2 befindet und der CMA-ES auch Werte zwischen Minus vier und Minus zwei findet. In der Abbildung ist jedoch deutlich zu erkennen, dass hier eher Modelle mit verhältnismäßig schlechter MLE gefunden wurden.

4.2 Parameteroptimierung des Optimierers durch GridResearch

Der CMA-ES hat, wie auch andere Algorithmen, Parameter, die verändert werden können und sich auf die Ergebnisse auswirken können. Im Folgenden wird daher ein GridSearch angewendet. Dabei werden für mehrere Parameter verschiedene Werte vorgegeben. Der Algorithmus wird dann mehrfach durchgeführt, sodass jeder Werte eines jeden Parameters mit jedem anderen Wert eines jeden anderen Parameters einmal durchgeführt. Da es sich um ein randomisiertes Verfahren handelt, reicht es nicht aus, die Werte zu verändern und das Modell einmal trainieren zu lassen. Auch hier müssen mehrere Iterationen erfolgen. Um hier die Rechendauer in einem annehmbaren Bereich zu halten, werden jeweils zehn Iterationen durchgeführt. Von diesen zehn Iterationen wird der Durchschnitt der MLE berechnet.

Den größten Einfluss haben Mu und Lambda. Mu ist die Populationsgröße und Lambda die Anzahl der Kinder (vgl. bartzbeielstein, 13.06.2020). Zu beachten gilt hier, dass Lambda mindestens so groß sein muss wie Mu. Die MLE wird besser, je größer Lambda ist, wobei zu beachten ist, dass hier nur bis zu einer gewissen Wertegrenze evaluiert wird, da die Ausführung des Modells mit dem Optimierer CMA-ES langsamer wird, je höher Lambda gesetzt wird. Hinzu kommt, dass ab einem gewissen Lambda-Wert die Modelle im Durchschnitt über alle Iterationen hinweg nur noch minimal besser werden und nicht mehr signifikant besser. Anstatt den höchsten ausprobierten Lambda-Wert und damit durchschnittlich beste Modelle zu erhalten, wird stattdessen ein Lambda-Wert genommen, bei dem der MLE sehr gut ist, die Ausführung aber nicht übermäßig lange dauert.

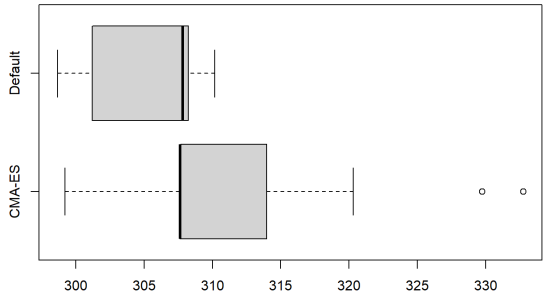


Abbildung 4.5: Evaluation MLE GridSearch

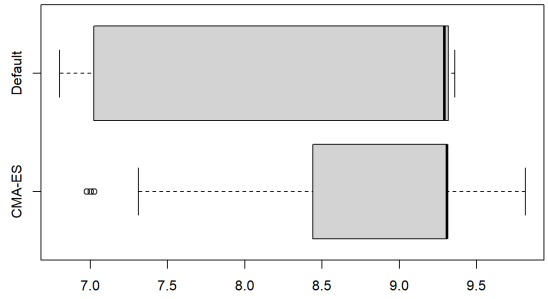


Abbildung 4.6: Evaluation RMSE GridSearch

Wie in der linken Abbildung (siehe Abb. 4.5 p.18) zu erkennen ist, ist der CMA-ES deutlich stabiler hinsichtlich der MLE geworden und die mit CMA-ES optimierten Modelle sind im Durchschnitt insgesamt besser. Der Median des CMA-ES liegt nun sogar minimal vor dem Median des DE. Die schlechteste MLE ist deutlich niedriger und in diesem Fall sind es auch nur wenige Ausreißer, die deutlich schlechter sind. Dafür gab es bei diesen Iterationen kein Modell mit CMA-ES, das genau so gut war wie ohne GridSearch. Das beste Modell stammt in diesem Fall von der Defaultlösung, wobei anzumerken ist, dass der beste CMA-ES hier nur minimal schlechter ist als ohne GridSearch. Es überwiegen daher die Vorteile, dass der CMA-ES mit Anpassung der Parameter deutlich stabilere Ergebnisse liefert.

Besonders bei Betrachtung der rechten Abbildung (siehe Abb. 4.6 p.18), die den RMSE abbildet, zeigt sich, dass sich die Parameteroptimierung lohnt. Der Median der mit CMA-

ES optimierten Modele liegt zwar immer noch bei etwas über neun, dafür wurden aber, wie oben bereits erwähnt, die schlechten Modelle eliminiert. Außerdem gibt es ein Modell mit CMA-ES, dass die sieben erreicht hat. All dies spricht dafür, die Parameter auch im weiteren Verlauf anzupassen.

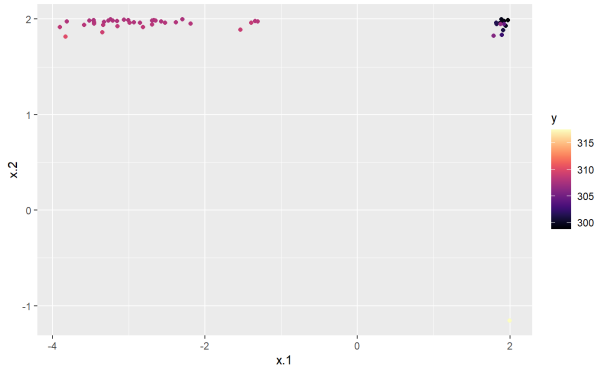


Abbildung 4.7: Evaluation Theta Default

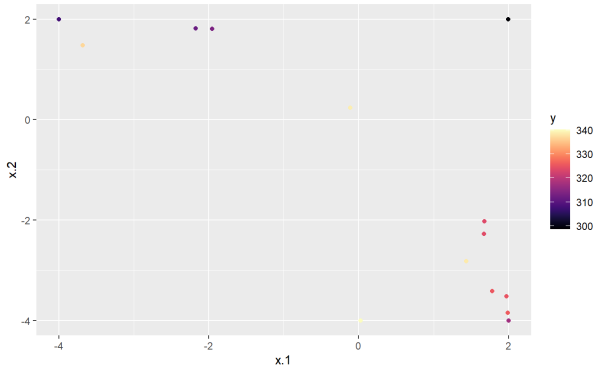


Abbildung 4.8: Evaluation Theta CMA-ES

Bei Betrachtung der Theta fällt auf, dass der CMA-ES (siehe Abb. 4.8 p.19) nun deutlich weniger Optima bei Minus vier für x2 findet und die sehr schlechten Modelle, die vermehrt im inneren Bereich waren, sind nun auch weg. Zufälligerweise ist auch bei diesen Iterationen beim Default (siehe Abb. 4.7 p.19) das schlechteste Modell weggefallen, bei dem ca. Minus eins für x2 vorhergesagt wurde.
Abschließend ist noch anzumerken, dass aufgrund der Begrenzung an Iterationen und dem Zufall eine gewisse Unsicherheit in den optimalen Werten für die Parameter vorhanden bleiben.

4.3 Änderung der Problemstellung

Die Daten und die Problemstellung können auf vielfältige Art und Weise geändert werden, um den CMA-ES mit sich selbst oder dem DE zu vergleichen. Für eine erste Idee werden die erzeugten Daten mit den Vorhersagen der Modelle über den gesamten Bereich innerhalb der Ober- und Untergrenzen verglichen. Bei näherer Betrachtung (siehe Abb. 4.9 p.20) fällt auf, dass sich die erzeugten Datenpunkte in bestimmten Bereichen häufen und in anderen Bereichen nur sehr wenige Datenpunkte vorhanden sind. Nun können wir beide Modelle einmal alle Werte vorhersagen lassen, um ein genaues Bild von den Algorithmen zu bekommen.

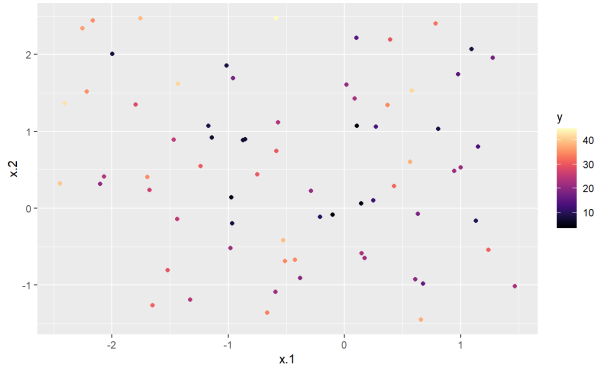


Abbildung 4.9: Verteilung der Datenpunkte über den Raum

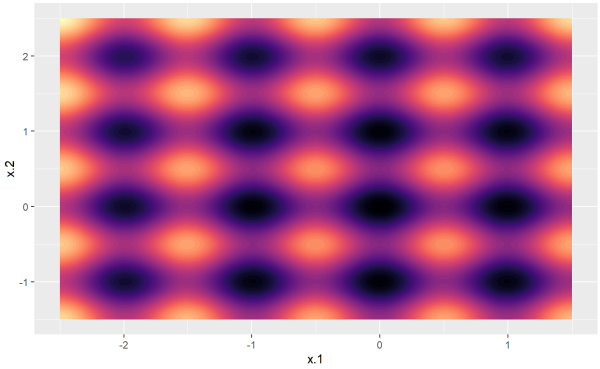


Abbildung 4.10: Visualisierung der Funktion

Für eine bessere Vergleichbarkeit wird jeweils das beste mit dem DE (siehe Abb. 4.11 p.20) bzw. dem CMA-ES (siehe Abb. 4.12 p.20) optimierte Modell betrachtet. Auf den ersten Blick fällt direkt auf, dass sich die Modelle kaum unterscheiden, das ist in diesem Fall weniger interessant. Deutlich interessanter ist der Vergleich der Vorhersagen mit den Trainingsdaten und der Visualisierung der tatsächlichen Funktion (siehe Abb. 4.10 p.20). In den Bereichen, in denen sich die Trainingsdaten häufen, ähneln die Vorhersagen der tatsächlichen Funktion. Zu erkennen ist dies besonders im linken, unteren Bereich. Auch dort, wo sich Datenpunkte mit ähnlichen Funktionswerten häufen und wenig Varianz vorhanden ist, wie im gesamten unteren Bereich, sind die Modelle schlecht. Bei Betrachtung der Funktionsweise des Gaußschen Prozessmodells erklärt sich der Grund hierfür sehr schnell. Da nur wenige Datenpunkte bzw. Datenpunkte mit ähnlichen Funktionswerten zum Training vorhanden sind, orientiert sich das Modell an eben diesen vorhanden Datenpunkten und glättet die Schätzung. Sie wird dadurch ungenauer und entspricht in diesen Bereichen nicht der tatsächlichen Funktion.

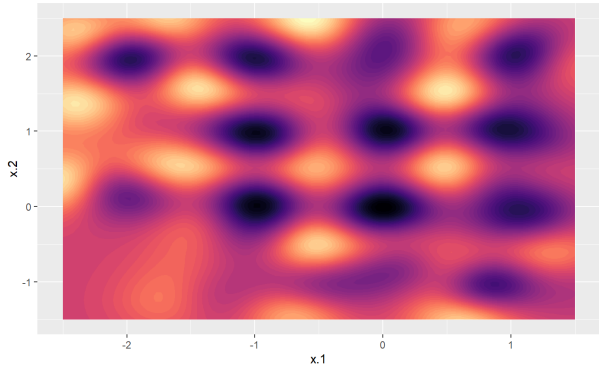


Abbildung 4.11: Visualisierung des besten Modells der Defaultlösung

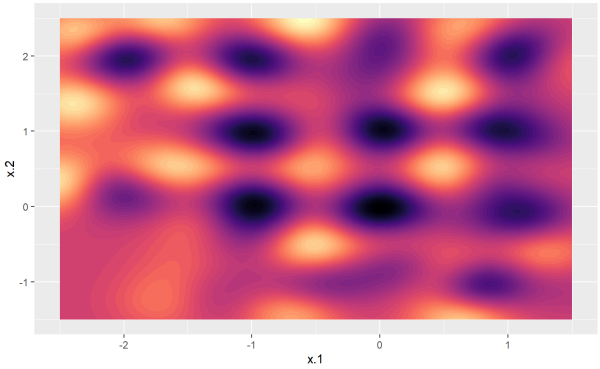


Abbildung 4.12: Visualisierung des besten Modells der CMA-ES Lösung

Um dies zu überprüfen, gibt es mehrere Möglichkeiten. Zum einen können die Trainingsdaten variiert werden. Wenn der Seed geändert oder entfernt wird und zufällig neue Daten generiert werden, werden sich die Bereiche mit gehäuften Datenpunkten ändern. Die Modelle sollten sich dann entsprechend anpassen und in den Bereichen näher an der tatsächlichen Funktion sein, in denen sich die Datenpunkte häufen. Dafür werden sie anderen Bereichen schlechter.

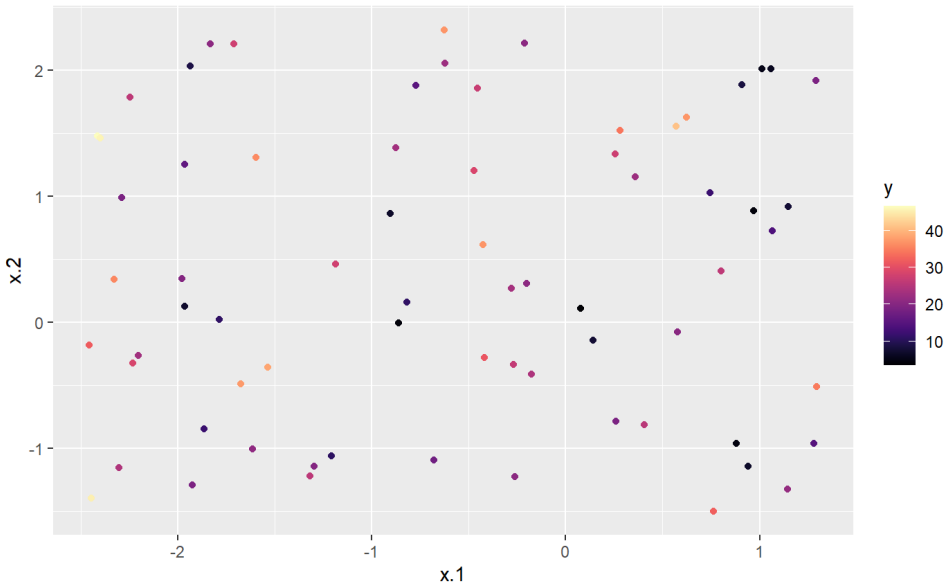


Abbildung 4.13: Daten ohne Seed

Im Folgenden werden also neue Datenpunkte ohne vorgeschriebenen Seed erzeugt (siehe Abb. 4.13 p.21). Die Verteilung der Daten über den Raum verändert sich wie in der Abbildung zu erkennen ist. Häufungen von Datenpunkten gibt es nun vor allem im linken, mittleren und rechten Bereich mit Lücken dazwischen, wobei der untere Teil des Bereichs mehr Datenpunkte aufweist, als der obere. Die Modelle sollten also in ihren Vorhersagen der tatsächlichen Funktion vor allem links, links unten, teilweise in der Mitte und rechts sehr ähnlich sehen. In den anderen Bereichen sollten die Vorhersagen deutlich von der Funktion abweichen. Um dies zu überprüfen, werden in einer Schleife die beiden besten Modelle gesucht. Sowohl der Default (siehe Abb. 4.14 p.22), als auch der CMA-ES (siehe Abb. 4.15 p.22) zeigen eben vermutetes Verhalten.

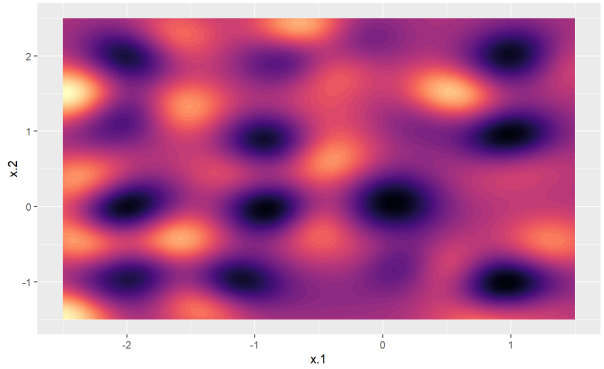


Abbildung 4.14: Visualisierung ohne Seed der Defaultlösung

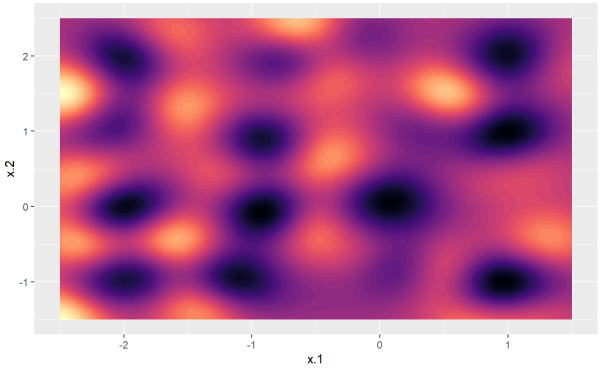


Abbildung 4.15: Visualisierung ohne Seed der CMA-ES Lösung

Anstatt die Daten zu ändern, hätte diese Hypothese auch auf andere Art überprüft werden können. Wenn bekannt ist, dass für bestimmte Ausprägungen der Merkmale nur wenige Daten zur Verfügung standen, kann davon ausgegangen werden, dass die Modelle hierfür schlechte Vorhersagen treffen. Sinnvollerweise könnten dann auch die Daten, die vorhergesagt werden sollen, angepasst werden. Für dieses Beispiel werden die ursprünglichen Daten mit einem Seed von eins genommen. Die Modelle werden wie bisher trainiert, doch für die Evaluation werden die Daten, die erzeugt werden, auf den Bereich begrenzt, in dem auch mehr Trainingsdaten vorliegen. Der Bereich für die Daten zur Evaluation wird daher eingegrenzt auf folgende Werte.

$$x1[-1,5; 1,5]$$

$$x2[-0,5; 2,5]$$

Dies grenzt die sehr schlechten Bereiche aus, ohne den Bereich zu stark einzugrenzen. Die Modelle werden nun anhand des RMSE evaluiert. Da die ursprünglichen Daten für das Training verwendet werden, wären Änderungen der MLE nur auf den Zufall zurückzuführen und nicht auf die veränderten Evaluationsdaten.

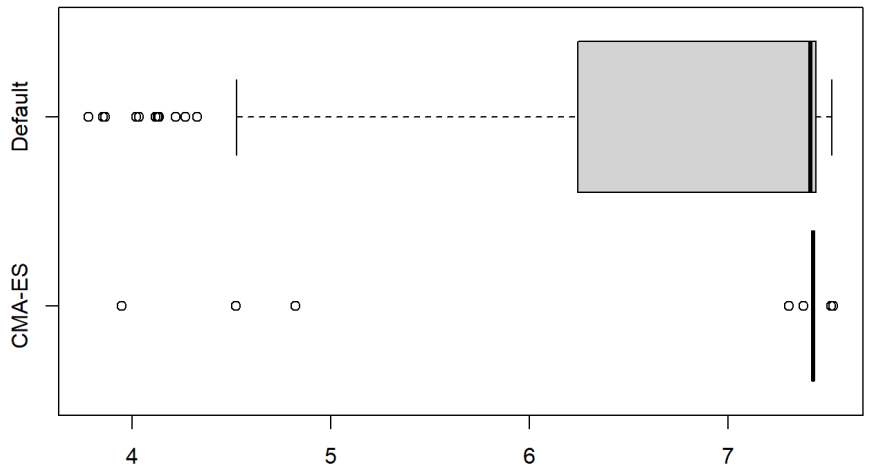


Abbildung 4.16: Evaluation begrenzt des RMSE

Beim RMSE gibt es deutliche Änderungen (siehe Abb. 4.16 p.23). Während dieser vorher im Mittel bei über neun lag und die besten Modelle auf einen RMSE von etwas besser als sieben gekommen sind, gibt es nun Modelle mit einem RMSE von weniger als vier. In der Vorhersagegenauigkeit sind die Modelle also wie erwartet in dem begrenzten Bereich deutlich besser.

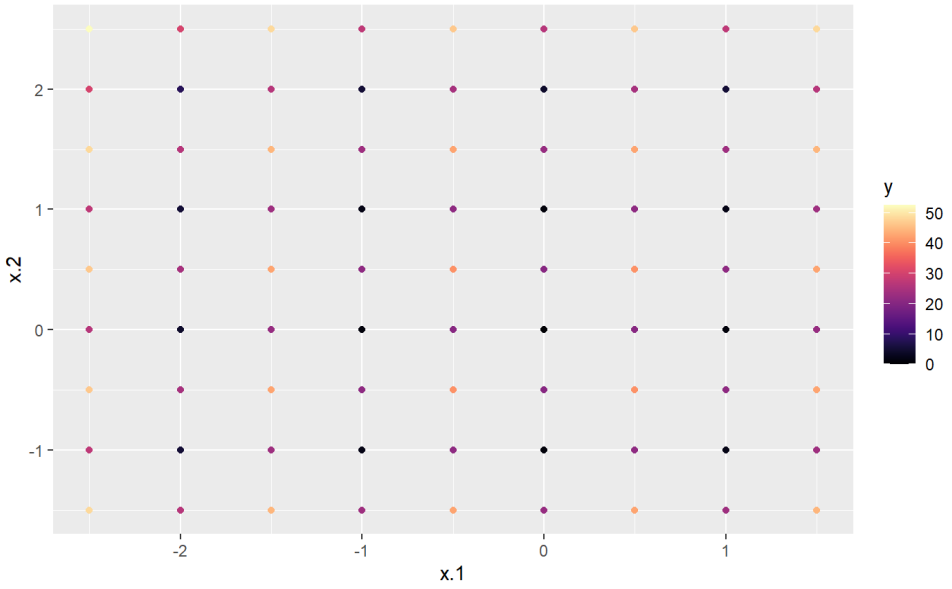


Abbildung 4.17: Gleichmäßig verteilte Datenpunkte

Nun läge die Überlegung nahe, dass die Modelle am besten sind, wenn die Daten gleichmäßig über den Raum verteilt sind. Die Trainingsdaten werden also nun so erzeugt, dass

ein Gitter von Datenpunkten gleichmäßig über den Raum verteilt erzeugt wird (siehe Abb. 4.17 p.23). In einem realen Projekt werden die Trainingsdaten niemals so vorliegen, für die empirische Untersuchung dieser Arbeit ist dieser Fall jedoch dennoch interessant zu betrachten.

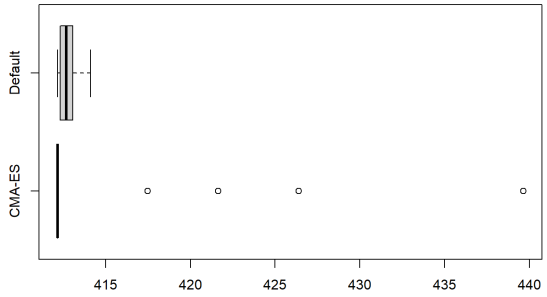


Abbildung 4.18: verteilte Datenpunkte der Evaluation der MLE

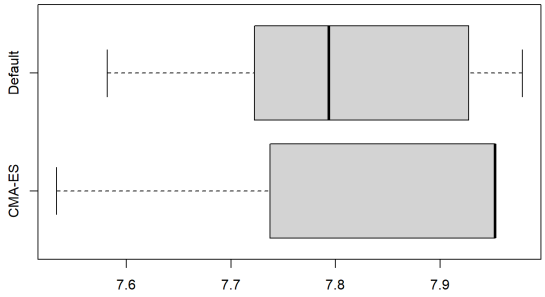


Abbildung 4.19: verteilte Datenpunkte der Evaluation der RMSE

Dem Gaußschen Prozessmodell sollte es nun möglich sein, eine beinahe perfekte Lösung zu finden. Werden die beiden Algorithmen evaluiert, ist jedoch sehr schnell zu erkennen, dass die Modelle nicht so gut sind, wie erwartet. Eine Auffälligkeit ist, dass beide Modelle eine sehr geringe Varianz der MLE (siehe Abb. 4.18 p.24) haben, lediglich beim CMA-ES gibt es ein paar schlechtere Ausreißer, in den überwiegenden Fällen ist es jedoch besser als der DE. Der RMSE (siehe Abb. 4.19 p.24) ist besser als bei zufällig im Raum verteilten Datenpunkten. Das beste Modell ist ein mit CMA-ES optimierter, der Median des Default liegt jedoch deutlich vor dem Median des CMA-ES.

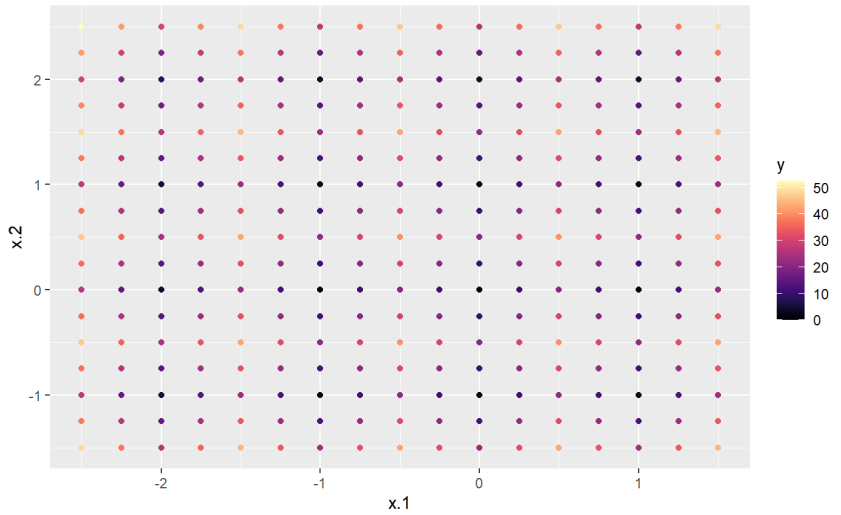


Abbildung 4.20: Gleichmäßig verteilte doppelte Anzahl der Datenpunkte

Da die Modelle nicht wie erwartet sehr gut sind, werden in einem weiteren Schritt noch mehr Datenpunkte erzeugt, ebenfalls nach dem gleichen Muster, sodass sie gleichmäßig innerhalb der Unter- und Obergrenzen über den Raum verteilt sind (siehe Abb. 4.20 p.24).

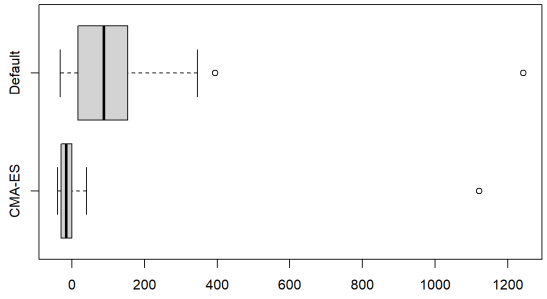


Abbildung 4.21: verteilte doppelte Anzahl der Datenpunkte der MLE

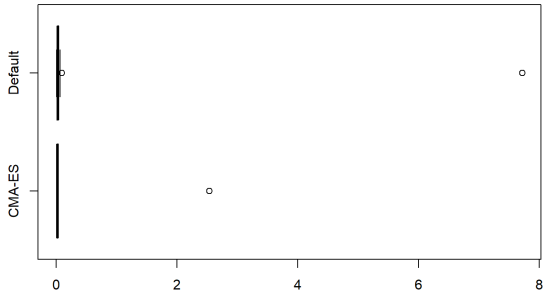


Abbildung 4.22: verteilte doppelte Anzahl der Datenpunkte der RMSE

Nun zeigt sich das vermutete Verhalten zumindest bei einigen Modellen. Sowohl der Default, als auch der CMA-ES erreichen eine MLE nahe null (siehe Abb. 4.21 p.25), wobei auch zu erwähnen ist, dass hier Modelle mit der bisher schlechtesten MLE von über 1.000 gefunden wurden. Bis auf eine Ausnahme liegen alle mit CMA-ES optimierten Modelle deutlich enger zusammen, als die mit DE optimierten Modelle. Der RMSE (siehe Abb. 4.22 p.25) liegt bei beiden Optimierungsalgorithmen bei nahe null, bis auf ein paar wenige Ausnahmen. Hier ist zu erwähnen, dass das schlechteste Modell ein mit dem Default optimiertes ist.

Die bisher vorgenommenen Änderungen waren auf die Verteilung der Daten im Raum der Trainings- und Testdaten bezogen. Im Folgenden soll nun die Menge der zufällig erzeugten Trainingsdaten verändert werden. Die bisher vorgenommenen Änderungen waren ausschließlich auf die Verteilung der Daten im Raum der Trainings- und Testdaten bezogen. Im Folgenden soll nun die Menge der vorhandenen Trainingsdaten verändert werden. Die bisherige Datenmenge wird einmal verdoppelt auf 140 und einmal halbiert auf 35. Zu erwarten wäre, dass die Vorhersage bei 140 Datenpunkten genauer wird, das Training dafür aber auch länger benötigt. 35 Datenpunkte werden vermutlich nicht ausreichen, um überzeugende Modelle zu erstellen. Interessant wird jedoch der Vergleich der beiden Optimierungsalgorithmen, vor allem, ob sich einer der beiden als besser bei höheren oder geringeren Datenmengen herausstellt.

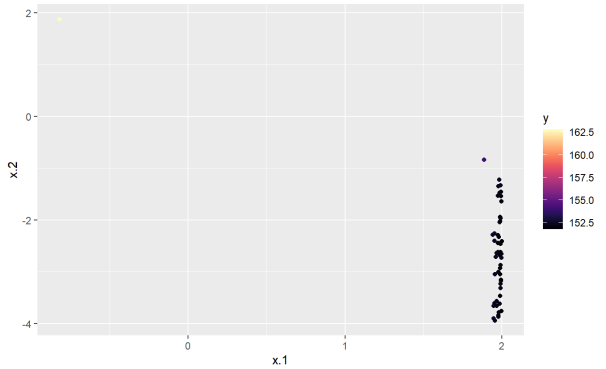


Abbildung 4.23: Evaluation der Thetas der halben Datenmenge des Default

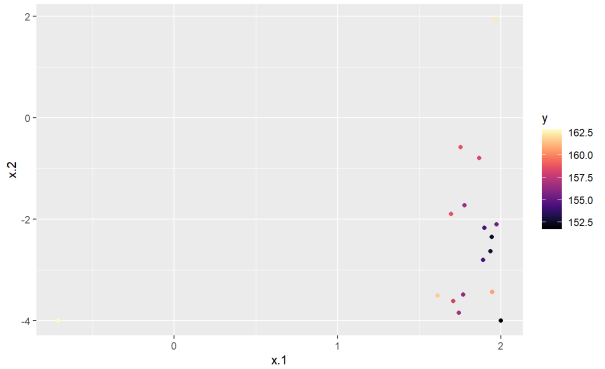


Abbildung 4.24: Evaluation der Thetas der halben Datenmenge des CMA-ES

Bei Betrachtung der Thetas fällt auf, dass sich beide Algorithmen für x1 einen Wert nahe zwei finden, lediglich einen Ausreißer gibt es bei nahe Minus 1 für x1 bei beiden Modellen. Bereits bei diesen Visualisierungen fällt auf, dass es beim CMA-ES (siehe Abb. 4.24 p.26) deutlich unterschiedliche MLE gibt, während es beim DE (siehe Abb. 4.23 p.26) lediglich der Ausreißer einen schlechteren MLE gibt. Im nächsten Schritt wird daher die MLE in den Boxplots näher betrachtet.

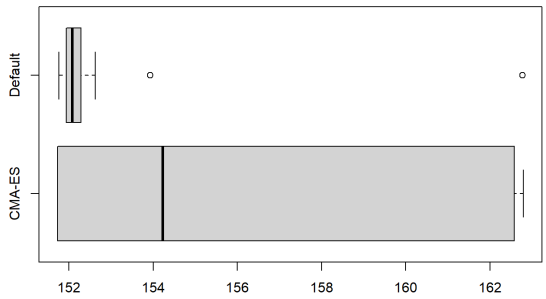


Abbildung 4.25: Evaluation der MLE der halben Datenmenge des Default

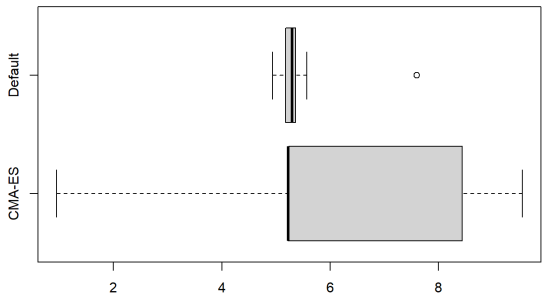


Abbildung 4.26: Evaluation des RMSE der halben Datenmenge des CMA-ES

Der Default (siehe Abb. 4.25 p.26) ist auf diesen Daten deutlich stabiler als der CMA-ES (siehe Abb. 4.26 p.26), wobei dessen Median nicht deutlich schlechter ist als der des Default. Die Parameter des CMA-ES sind bereits optimiert, der Grund für die hohe Schwankungsbreite kann also nicht wie auf den ursprünglichen Daten durch die Optimierung der Parameterwerte verbessert werden. Es überrascht daher auch nicht, dass der CMA-ES größere Varianzen beim RMSE aufweist. Um Mittel sind die beiden Modelle in etwa gleich gut, wobei hier überraschenderweise der RMSE bei ungefähr fünf liegt, die

Vorhersagen also besser waren als bei den ursprünglich 70 Datenpunkten. Außerdem ist das beste Modell hinsichtlich RMSE ein mit CMA-ES optimiertes, das bei nahe null liegt. Bei nicht optimierten Parameterwerten des CMA-ES sind sowohl die MLE als auch der RMSE deutlich schlechter. Im Folgenden wird die Datenmenge auf 140 zufällig verteilte Datenpunkte erhöht und die Modelle evaluiert.

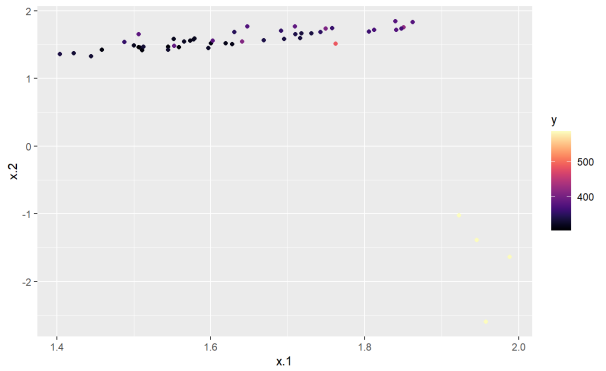


Abbildung 4.27: Default Lösung mit doppelter Datenmenge

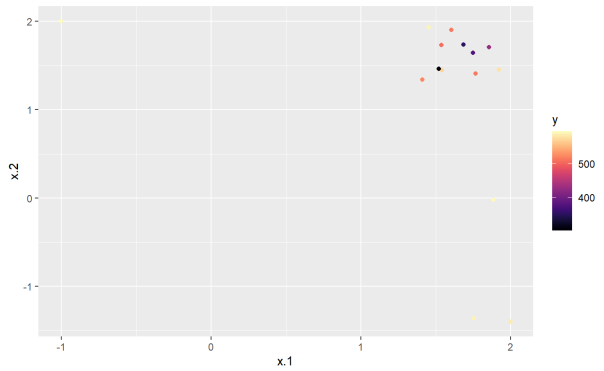


Abbildung 4.28: CMA-ES Lösung mit doppelter Datenmenge

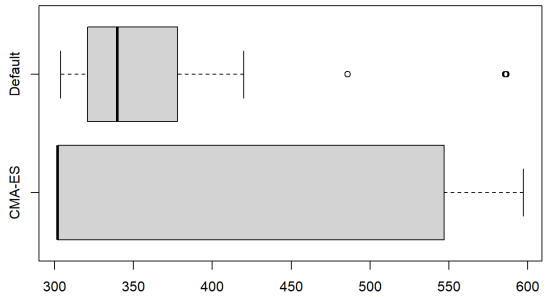


Abbildung 4.29: Doppelter Datenmenge mit der MLE

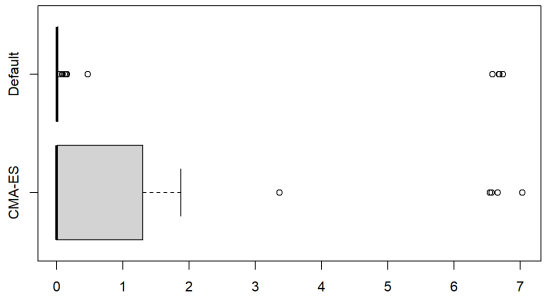


Abbildung 4.30: Doppelter Datenmenge mit dem RMSE

Wenn die Menge der Daten verdoppelt wird, ergeben sich andere Thetas als bei halber Datenmenge. Bei halber Datenmenge wurde für x1 überwiegend zwei vorhergesagt und x2 variiert. Bei der doppelten Datenmenge wird für das Theta für x2 überwiegend Werte zwischen eins und zwei vorhergesagt. Das Theta für x1 variiert beim DE wieder deutlich, beim CMA-ES (siehe Abb. 4.28 p.27) werden vor allem Werte zwischen eins und zwei gefunden. Die MLE (siehe Abb. 4.29 p.27) variiert bei doppelter Datenmenge sehr stark, wobei der Default (siehe Abb. 4.27 p.27) wie üblich deutlich stabiler ist. Im Median besser ist aber der CMA-ES, wobei sowohl das beste, als auch das schlechteste Modell ein mit

CMA-ES optimiertes ist. Erwähnenswert ist, dass die MLE von 300 bis 600 variiert, also bei manchen Modellen doppelt so hoch ist wie bei anderen.
Wie zu erwarten war, ist der RMSE (siehe Abb. 4.30 p.27) bei doppelter Datenmenge deutlich besser. Beinahe alle Modelle des Default sind nahe null und auch der Median des CMA-ES liegt bei nahe null. Es gibt einzelne Ausreißer mit deutlich schlechterem RMSE, insgesamt würden aber bereits wenige Iterationen ausreichen, um ein Modell mit sehr gutem RMSE zu erhalten. Wichtig zu erwähnen ist jedoch, dass sich die Rechendauer deutlich erhöht hat mit Erhöhung der Datenmenge.
In einem nächsten Schritt soll überprüft werden, wie sich die Optimierer verhalten, wenn die Anzahl an Dimensionen verändert wird. Anzumerken ist, dass die Anzahl der Dimensionen nicht verringert werden kann, da wie bei der aktuellen Problemstellung nur zwei haben und wenn hier eine Dimension entfernt wird, handelt es sich um ein lineares Problem. Daher wird im Folgenden die Anzahl der Dimensionen nur erhöht und nicht verringert.

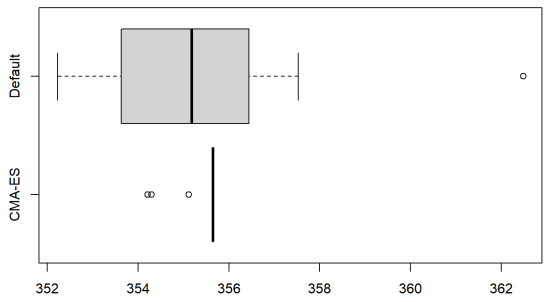


Abbildung 4.31: Drei Dimensionen MLE

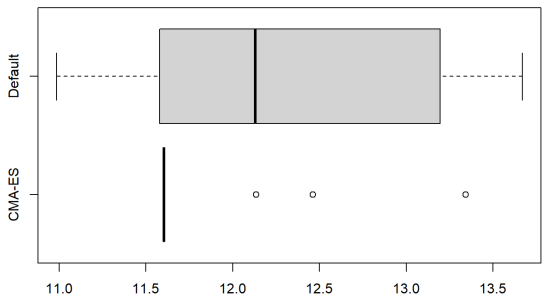
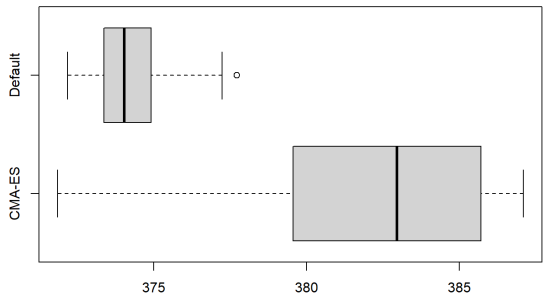


Abbildung 4.32: Drei Dimensionen RMSE

Wird die Anzahl der Dimensionen auf drei erhöht, ist das erste, was auffällt, dass der CMA-ES sehr stabil ist mit wendigen Ausreißern, sowohl bei der MLE (siehe Abb. 4.31 p.28) als auch beim RMSE (siehe Abb. 4.32 p.28). Besser ist jedoch bei den meisten Iterationen der Default, dessen Median liegt vor dem Median des CMA-ES. Das beste und schlechteste Modell hinsichtlich MLE und RMSE sind vom Default. Beim RMSE sind die meisten Modelle des CMA-ES besser, als die des DE.



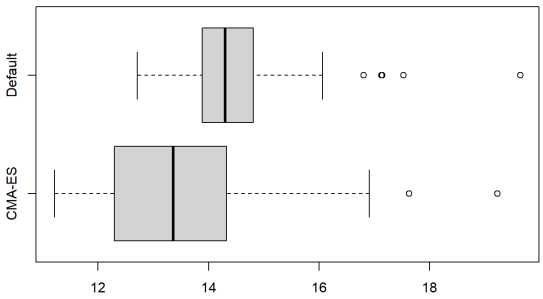
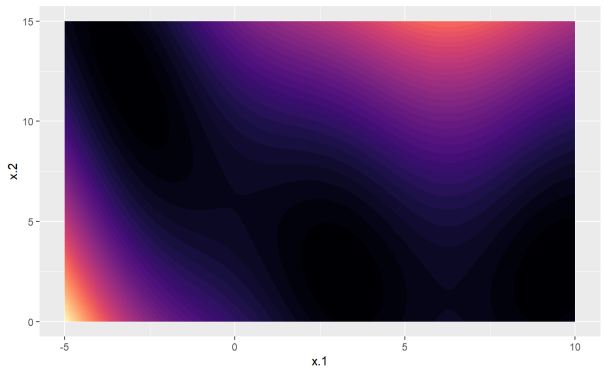


Abbildung 4.33: Vier Dimensionen MLE **Abbildung 4.34:** Vier Dimensionen RMSE

Wird die Anzahl der Dimensionen auf vier erhöht, ergibt sich direkt ein deutlich anderes Bild. Sowohl die MLE (siehe Abb. 4.33 p.29), als auch der RMSE (siehe Abb. 4.34 p.29) des Default variieren weniger als beim CMA-ES. Die beiden Modelle mit der besten und der schlechtesten MLE sind mit CMA-ES optimierte, also genau anders herum, als bei drei Dimensionen. Bei Betrachtung des RMSE fällt auf, dass auch hier der Median des CMA-ES wie bei drei Dimensionen, besser ist, als der Median des Default. In diesem Fall ist auch das beste Modell ein mit CMA-ES und das schlechteste ein mit dem Default optimiertes. Bisher wurde immer die gleiche Funktion genutzt. Im nächsten und letzten Schritt wird daher eine andere Funktion gewählt und die Modelle werden neu evaluiert. Der Funktionsterm lautet:

$$(x_2 - 5,1/(4 * \Phi^2) * (x_1^2) + 5/\Phi * x_1 - 6)^2 + 10 * (1 - 1/(8 * \Phi)) * cos(x_1) + 10$$

Zuerst müssen neue Datenpunkte erstellt werden, mit denen die Modelle trainieren können.



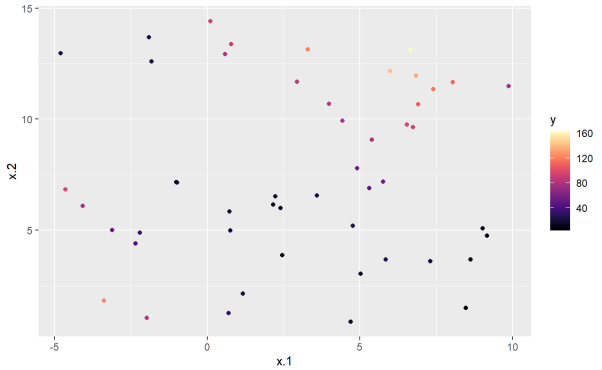


Abbildung 4.35: Visualisierung von der neuen Funktion

Abbildung 4.36: Visualisierung der Datenpunkte der neuen Funktion

Mit diesen Datenpunkten (siehe Abb. 4.36 p.29) werden die Modelle nun trainiert und für den Default und den CMA-ES wird jeweils das beste Modell innerhalb einer gewissen Anzahl an Iterationen gesucht. Die Vorhersagen über den gesamten Raum innerhalb der Unter- und Obergrenze werden dann visualisiert (siehe Abb. 4.35 p.29), um einen ersten Eindruck zu erhalten, wie nahe sie der tatsächlichen Funktion kommen.

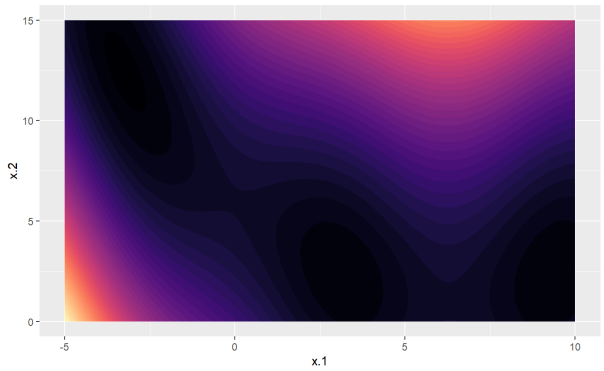


Abbildung 4.37: Visualisierung von der neuen Funktion mit dem Default

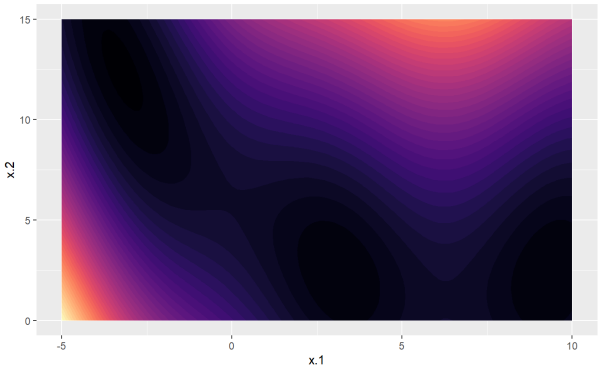


Abbildung 4.38: Visualisierung der Datenpunkte der neuen Funktion mit dem CMA-ES

Wie anhand der Visualisierungen zu erkennen ist, sehen sowohl das mit dem DE (siehe Abb. 4.37 p.30) als auch das mit dem CMA-ES (siehe Abb. 4.38 p.30) optimierte Modell der tatsächlichen Funktion sehr ähnlich. Es ist also davon auszugehen, dass auch die MLE und der RMSE deutlich besser sind als auf der ursprünglich genutzten Funktion.

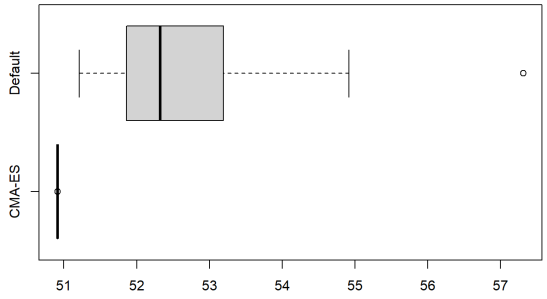


Abbildung 4.39: MLE der neuen Funktion

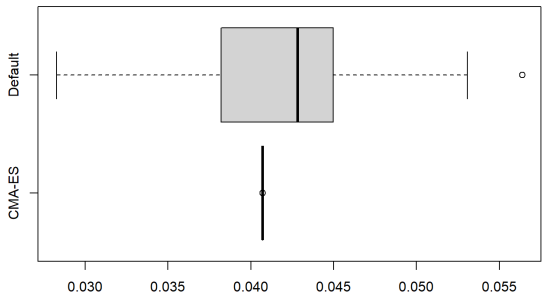


Abbildung 4.40: RMSE der neuen Funktion

Wie vermutet ist die MLE (siehe Abb. 4.39 p.30) deutlich besser. Bei der ursprünglichen Funktion lag diese bei nahe 300 und bei dieser liegt sie zwischen 50 und 60. Der CMA-ES

gelangt über alle 50 Iterationen hinweg zur gleichen MLE und ist auch besser, als jedes mit dem Default optimierte Modell. Beim RMSE (siehe Abb. 4.40 p.30) ergibt sich ein etwas anders Bild. Der CMA-ES gelangt zwar auch hier zum immer gleichen Ergebnis und der DE variiert etwas mehr, jedoch ist das beste Modell ein mit dem DE optimiertes. Dafür ist auch das schlechteste Modell eines vom Default. Wobei hier zu erwähnen ist, dass der RMSE so gering ist, dass vermutlich zu vernachlässigen ist, welcher Optimierer gewählt wird.
Bei der Funktion fällt auf, dass die Minima nahe beieinander liegen und nicht wie im vorherigen Bereich durch deutlich höhere Funktionswerte unterbrochen werden. Aufgrund der Funktionsweise des Gaußschen Prozessmodelles ist es bei dieser Funktion daher möglich, deutlich bessere Ergebnisse zu erhalten.

5 Fazit

Ziel der Arbeit war es, den DE und den CMA-ES als Optimierungsalgorithmus für die Optimierung der MLE eines Gaußschen Prozessmodells zu verwenden und die beiden Algorithmen miteinander zu vergleichen. Hierzu wurden zuerst die theoretischen Grundlagen erklärt, um die Hintergründe der Ergebnisse der empirischen Untersuchung nachvollziehen zu können. Nach der Klärung der Klärung und Beschreibung der Problemstellung wurde eine Evaluierungskonzept erarbeitet. Dieses hilft dabei, die beiden Algorithmen nachvollziehbar und korrekt miteinander zu vergleichen. Anschließend folge die Empirische Untersuchung. Dabei konnten einige interessante Informationen gewonnen werden.
Beim Vergleich der beiden Optimierer ist das erste, was auffällt, dass der DE deutlich stabiler in den Ergebnissen ist als der CMA-ES. Besonders die MLE, aber auch der RMSE variieren beim Default deutlich weniger. In diesem Zusammenhang ist es interessant zu sehen, dass der CMA-ES über die Iterationen hinweg immer wieder die gleichen Werte für die Thetas ermittelt, während die beim Default leicht, aber stetig variieren. Der CMA-ES kann jedoch über die Parameter etwas stabiler gestalten. Besonders wichtig sind dabei die Werte von Mu und Lambda. Je größer Lambda im Versuch eingestellt wurde, desto besser wurden die Modelle über die gesamten Iterationen hinweg besser. Zu beachten ist jedoch, dass die Erhöhung von Lambda die Rechendauer deutlich verlängert, also eine Entscheidung getroffen werden muss. Entweder eine deutlich längere Rechendauer und

dafür verhältnismäßig etwas bessere Modelle oder durchschnittlich gute, aber nicht immer die besten Modelle mit moderater Rechendauer. Im weiteren Verlauf der Arbeit wurden die Grenzen des Gaußschen Prozessmodells trotz Optimierer aufgezeigt. Die Modelle sind nur gut, wenn genügend Datenpunkte vorhanden sind. Bei realen Problemen muss hier überlegt werden, ob schlechte Vorhersagen in Bereichen, für die nur wenige Trainingsdaten vorhanden sind, hingenommen werden oder ob für diese Bereiche gar nicht erst Vorhersagen getroffen werden. Auch bei der gleichmäßigen Verteilung der Daten im Raum, wozu es in realen Projekten vermutlich niemals kommen wird, müssen genügend Datenpunkte vorhanden sein, um sehr gute Modelle zu erhalten.

Bei Verringerung der Datenmenge werden die Modelle nicht, wie erwartet, schlechter in den Vorhersagen. Der RMSE ist in diesem Fall sogar etwas besser. Bei Erhöhung der Datenmenge sind die Modelle, wie erwartet, ebenfalls besser. Jedoch muss auch in diesem Fall eine Entscheidung getroffen werden - je mehr Daten, desto länger die Rechendauer. Bei realen Problemen muss daher überlegt werden, wie viele Daten gesammelt und an die Modelle übergeben werden, sodass die Vorhersagen zwar gut genug sind, das Training aber nicht zu lange dauert.

Bei der Nutzung einer anderen Funktion hat sich gezeigt, dass die Modelle auf dieser deutlich bessere Ergebnisse liefern. Sowohl die MLE, als auch der RMSE sind auf der neuen Funktion deutlich besser, als auf der vorherigen.

In dieser Arbeit wurden nur der DE und der CMA-ES, zwei Evolutionsstrategen, miteinander verglichen. Für zukünftige Arbeiten wäre ein Vergleich mit weiteren Algorithmen interessant, vor allem mit Optimierten die nicht aus dem Bereich der Evolutionsstrategien stammt. Denkbar wäre auch, die Problemstellung noch einmal deutlich zu verändern und noch höher dimensionierte Daten zu verwenden. Hier würde sich die Frage stellen, ob der CMA-ES ab einer bestimmten Dimension so viel besser als der DE ist, dass sich ein weiterer Vergleich gar nicht mehr lohnt.

Literatur

[Akimoto und Hansen 2020] Akimoto, Y. ; Hansen, N.: Diagonal Acceleration for Covariance Matrix Adaptation Evolution Strategies. In: *Evolutionary Computation* 28 (2020), 09, Nr. 3, S. 405–435. – URL https://doi.org/10.1162/evco_a_00260

[bartzbeielstein 13.06.2020] bartzbeielstein: Build Kriging Model. (13.06.2020). – URL https://rdrr.io/github/bartzbeielstein/SPOT/man/buildKriging.html. – Zugriffsdatum: 11.08.2022

[C3 AI 28.09.2021] C3 AI: *Root Mean Square Error (RMSE) - C3 AI*. 28.09.2021. – URL https://c3.ai/glossary/data-science/root-mean-square-error-rmse/. – Zugriffsdatum: 10.08.2022

[Cansiz 2021] Cansiz, Sergen: Interpretation of Covariance, Covariance Matrix and Eigenvalues | Towards Data Science. In: *Towards Data Science* (2021). – URL https://towardsdatascience.com/5-things-you-should-know-about-covariance-26b12a0516f1. – Zugriffsdatum: 07.08.2022

[Castillo 2021] Castillo, Oscar: *Differential Evolution Algorithm with Type-2 Fuzzy Logic for Dynamic Parameter Adaptation with Application to Intelligent Control*. Cham : Springer International Publishing AG, 2021 (SpringerBriefs in applied sciences and technology. Computational intelligence). – URL https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=6403596. – ISBN 978-3-030-62133-9

[Dang u. a. 2019] Dang, Viet-Hung ; Vien, Ngo A. ; Chung, TaeChoong: A covariance matrix adaptation evolution strategy in reproducing kernel Hilbert space. In: *Genetic Programming and Evolvable Machines* 20 (2019), Nr. 4, S. 479–501. – ISSN 1389-2576

[Forrester u. a. 2008] Forrester, Alexander I. J. ; Sóbester, András ; Keane, Andy J.: *Engineering Design via Surrogate Modelling*. Wiley, 2008. – ISBN 9780470060681

[Gagganapalli 2015] Gagganapalli, Srikanth R.: *Implementation and evalution of CMA-ES algorithm*, North Dakota State University of Agriculture and Applied Science, Diplomarbeit, 2015

[Galarnyk 12.09.2018] Galarnyk, Michael: Understanding Boxplots - Towards Data Science. In: *Towards Data Science* (12.09.2018). – URL https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51. – Zugriffsdatum: 10.08.2022

[Georgioudakis und Plevris 2020] Georgioudakis, Manolis ; Plevris, Vagelis: A Comparative Study of Differential Evolution Variants in Constrained Structural Optimization. In: *Frontiers in Built Environment* 6 (2020)

[Géron 2020] Géron, Aurélien: *Praxiseinstieg Machine Learning mit Scikit-Learn, Keras und TensorFlow: Konzepte, Tools und Techniken für intelligente Systeme*. 2. Auflage. Heidelberg : O'Reilly, 2020. – URL `https://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=2541564`. – ISBN 9783960103394

[Hansen und Ostermeier 2001] Hansen, N. ; Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. In: *Evolutionary Computation* 9 (2001), Nr. 2, S. 159–195

[Hansen 2006] Hansen, Nikolaus: The CMA Evolution Strategy: A Comparing Review. In: Lozano, Jose A. (Hrsg.): *Towards a new evolutionary computation* Bd. 192. Berlin and Heidelberg and New York : Springer, 2006, S. 75–102. – ISBN 978-3-540-29006-3

[Hansen 2014] Hansen, Nikolaus: CMA-ES: A Function Value Free Second Order Optimization Method. (2014). – URL `https://hal.inria.fr/hal-01110313`

[Hansen 2016] Hansen, Nikolaus: *The CMA Evolution Strategy: A Tutorial*. https://arxiv.org/pdf/1604.00772. 2016

[Krause u. a. 2016] Krause, Oswin ; Arbonès, Dídac R. ; Igel, Christian: CMA-ES with Optimal Covariance Update and Storage Complexity. In: Lee, D. (Hrsg.) ; Sugiyama, M. (Hrsg.) ; Luxburg, U. (Hrsg.) ; Guyon, I. (Hrsg.) ; Garnett, R. (Hrsg.): *Advances in Neural Information Processing Systems* Bd. 29, Curran Associates, Inc., 2016. – URL `https://proceedings.neurips.cc/paper/2016/file/289dff07669d7a23de0ef88d2f7129e7-Paper.pdf`

[Kumar u. a. 2022] Kumar, B. V. (Hrsg.) ; Oliva, Diego (Hrsg.) ; Suganthan, P. N. (Hrsg.): *Springer eBook Collection*. Bd. 1009: *Differential Evolution: From Theory to Practice*. 1st ed. 2022. Singapore : Springer Singapore and Imprint Springer, 2022. – ISBN 978-981-16-8082-3

[Li u. a. 2020] Li, Zhenhua ; Lin, Xi ; Zhang, Qingfu ; Liu, Hailin: Evolution strategies for continuous optimization: A survey of the state-of-the-art. In: *Swarm and Evolutionary Computation* 56 (2020), S. 100694

[MIURA 2011] MIURA, Keiji: An Introduction to Maximum Likelihood Estimation and Information Geometry. In: *Interdisciplinary Information Sciences* 17 (2011), Nr. 3, S. 155–174. – ISSN 1340-9050

[Myung 2003] Myung, In J.: Tutorial on maximum likelihood estimation. In: *Journal of Mathematical Psychology* 47 (2003), Nr. 1, S. 90–100. – ISSN 00222496

[Pan und Fang 2002] Pan, Jian-Xin ; Fang, Kai-Tai: Maximum Likelihood Estimation. In: Pan, Jian-Xin (Hrsg.) ; Fang, Kai-Tai (Hrsg.): *Growth Curve Models and Statistical Diagnostics*. New York, NY : Springer New York, 2002 (Springer Series in Statistics), S. 77–158. – ISBN 978-1-4419-2864-1

[Phd 13.01.2019] Phd, Genevieve H.: Getting Started with Randomized Optimization in Python. In: *Towards Data Science* (13.01.2019). – URL `https://towardsdatascience.com/getting-started-with-randomized-optimization-in-python-f7df46babff0`. – Zugriffsdatum: 10.08.2022