State-Space Inference and Learning with Gaussian Processes

Garcia Cesar[1] und Sarbandi Janik[2]

***Abstract*— Das Erlernen dynamischer Systeme, mit unbekanntem Zustandsraum, ist ein aktuell noch ungelöstes Problem. Gaußprozesse werden zum Erlernen von Funktionen genutzt. In diesem Paper wird der Ansatz, von Ryan Turner, dynamische Systeme anhand von Gaußprozessen zu erlernen, erläutert. Hier bei werden zunächst die von Herrn Turner vorgestellten Konzepte vorgestellt und anschließend näher beschrieben. Außerdem behandelt dieses Paper den Versuch diese Konzepte nach zu implementieren.**

I. EINLEITUNG

Dieses Projekt wurde, im Rahmen des Faches "Stochastische Regelung und statistische Lernverfahren", erstellt. Die behandelte Thematik basiert auf dem Paper, "State-Space Inference and Learning with Gaussian Processes", von Ryan Turner und Marc Deisenroth [3].

Die Aufgabe des Projektes bestand zum Einen darin die im vorgegebenen Paper dargestellten Konzepte zu verstehen und zum Anderen darin herauszufinden in wie fern mit den vorgestellten Methoden und dem gegebenen Code weiter gearbeitet werden kann.

Inferenz wird im Zusammenhang mit dynamischen System zur Signalverarbeitung, aber auch zur Zustandsabschätzung genutzt. Das Ziel der Inferenz ist es, trotz verrauschter Signale oder fehlender Teilinformationen, zuverlässig den aktuellen Zustand abzuschätzen. Selbst wenn nicht alle Zustände des Zustandsraumes messbar sind, können diese mithilfe von Modellen prognostiziert werden.

In diesem Paper wird ein Algortithmus vorgestellt, welcher nur anhand verrauschter Messungen, ohne ein gegebenes Modell, alle Zustände abschätzt. Der Algorithmus versucht dabei abwechselnd den aktuellen Zustand abzuschätzen und ein Modell der Dynamik des Systems zu erlernen.

II. PROBLEMSTELLUNG

Der vorgeschlagene GPIL-Algorithmus zur Inferenz und zum Erlernen nichtlinearer dynamischen Systemen ist in der Lage, die dynamischen Gleichungen des Systems und der Messfunktionen zu modellieren. Darüber hinaus erfolgt dies mit nur T sequentiellen Messungen ($\mathbf{Y}$), des Systems und ohne Kenntnis des Zustandsraumes.

*Project within the course Statistical Learning and Stochastic Control, University of Stuttgart, February 6, 2019.

[1]Garcia Cesar ist ein Student in dem Fachbereich Technische Kybernetik an der Universität Stuttgart, st118962@stud.uni-stuttgart.de

[2]Sarbandi Jani ist ein Student in dem Fachbereich Technische Kybernetik an der Universität Stuttgart, st117200@stud.uni-stuttgart.de

Die Modellierung der beiden Funktionen erfolgt durch nichtparametrische Gauß'sche Prozesse. Die Gauß'schen Prozesse berücksichtigen drei spezifische Unsicherheiten: das Rauschen im System, das Rauschen in den Messungen und die fehlende Modellierung der Dynamik.

A. Systembeschreibung

Aufgrund der Eigenschaften des Algorithmus ist es erforderlich, dass das zu modellierende System die Markov-Eigenschaften erfüllt. Das bedeutet, dass sich die am System beteiligten Zustandsraumvariablen im Laufe der Zeit und nur in Abhängigkeit von ihrem letzten Zustand entwickeln. Daher betrachten wir nur die Zustands-Raum-Systeme, die durch folgende Gleichungen modelliert werden können:

$$\mathbf{x_t} = f(\mathbf{x_{t-1}}) + \epsilon, \mathbf{x_t} \in \mathbb{R}^M, \tag{1}$$

$$\mathbf{y_t} = g(\mathbf{x_t}) + \upsilon, \mathbf{y_t} \in \mathbb{R}^D. \tag{2}$$

Das Systemsrauschen $\epsilon \sim N(0, \Sigma_\epsilon)$ und das Messrausch $\upsilon \sim N(0, \Sigma_\upsilon)$ werden als Normalverteilt, mit Mittelwert Null, approximiert.

B. Gaussian Prozess Inferenz and Learning

"Ein Gaußscher Prozess ist eine Verallgemeinerung der Gaußschen Wahrscheinlichkeitsverteilung. Während eine Wahrscheinlichkeitsverteilung Zufallsvariablen beschreibt, die Skalare oder Vektoren sind (bei multivariaten Verteilungen), regelt ein stochastischer Prozess die Eigenschaften von Funktionen." [1, Kapitel 1,Seite 4]. Eins der heraus stechenden Merkmale eines Gausschen Prozesses ist der Konfidenzbereich. Die Zustandsschätzung gibt nicht nur den Erwartungswert des jeweiligen Zustandes aus, sondern berechnet eine Warscheinlichkeitsverteilung für den Wert dieses Zustandes.

Wie bereits erwähnt, lässt sich die Modellierung des Systems und der Messungsgleichung mit den sequentiellen Beobachtungen $\mathbf{Y} = [\mathbf{y_1}, \mathbf{y_2}, .., \mathbf{y_t}, .., \mathbf{y_T}]$ rekonstruieren. Da die Zustandsraumvariablen nicht vorgegeben sind, werden diese Parameter explizit spezifiziert und ein GP durch einen Pseudo-Trainingssatz parametrisiert, der als ein Satz freier Parameter für das Lernen angesehen wird. Diese Parametern beziehen sich auf die Pseudo-Trainingseingset, die für sparse GP-Approximationen verwendet werden. Diese Pseudo Trainingsset und die Markov Eingeschaft des Systems sind im Abbildung 1 dargestellt.

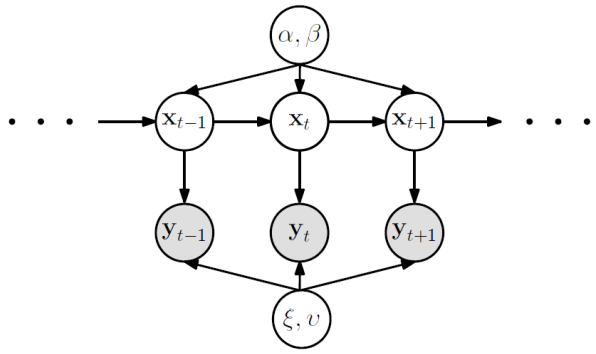

Fig. 1. Die Freie Parametern α, β, ξ, und υ in einem Markovian Schema

III. LÖSUNGSANSATZ

Bei dem vorgestellten Algorithmus handelt es sich um einen Expactation-Maximization (EM) Algorithmus. Der Algorithmus besteht aus zwei alternierenden Schritten. Der erste Schritt, die Inferenz, berechnet eine Wahrscheinlichkeitsverteilung der Zustände. Im zweiten Schritt wird anhand der vorher abgeschätzten Zustände das Modell, der Dynamik, des Systems verbessert. Aufgrund der Modellierung der Systemgleichungen in Form von Gaußprozessen, besteht die Modellverbesserung aus einer Hyperparameteroptimierung. Im folgenden werden die nötigen Hyperparemeter erläutert und die beiden Schritte des Algorithmusses genauer beschrieben. Der Ablauf des Algorithmus ist schematisch in Abbildung 2 dargestellt.

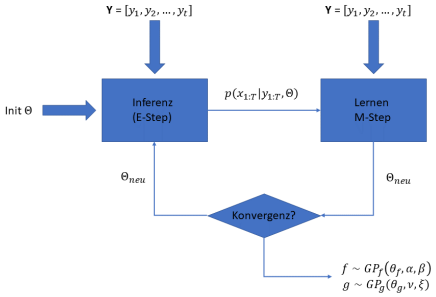

Fig. 2. Ablauf des EM-Algorithmus

A. Hyperparameter

Als Hyperparameter werden alle freien Parameter des Algorithmus bezeichnet. Diese müssen zunächst initialisiert und sollen im Laufe des Algorithmus optimiert werden. Der Hyperparametern Raum kann wie folgt zusammengefasst werden:

$$\Theta := \{\alpha, \beta, \xi, \upsilon, \theta_{\mathbf{f}}, \theta_{\mathbf{g}}, \mu_{\mathbf{0}}, \mathbf{\Sigma_0}\}$$

1) Pseudo Trainingsatz: Wird ein System durch Gaußprozesse modelliert braucht man nicht zwangsläufig alle Beobachtungen um eine gute Approximation des Systemverhaltens zu bekommen. Es reichen einige Punkte, wenn diese an den richtigen Stellen gewählt werden. Die so gewählten Punkte nennt man einen Pseudo-Trainingssatz. Normaler weiße können diese Punkte aus den Beobachtungen generiert werden. In unserem speziellen Fall sind aber weder die Eingänge noch die Ausgänge der Gleichung f bekannt. Außerdem sind zwar die Ausgänge der Gleichung g gegeben, jedoch keinerlei Information darüber, welche Eingänge zu diesen Messungen geführt haben. Aus diesem Grund wird der Pseudo-Trainingssatz als Teil des Hyperparameterraumes implementiert.

Die Systemgleichung f wird von den Pseudo-Inputs $\alpha = \{\alpha_i \in \mathbb{R}^M\}_{i=1}^N$ parametrisiert, die entsprechende Pseudo-Ziele der Funktion f sind mit $\beta = \{\beta_i \in \mathbb{R}^M\}_{i=1}^N$ bezeichnet.

Nach [3] sind die Pseudo-Inputs, die Lage im Zustandraum, an der sich die Mittelwerte der gauß'schen Basis Funktion (SE-KErnel), befinden. Während die Pseudo-Ziele die entsprechen Mittel der Werte der Funktionen der entsprechenden Lage sind.

Die Parametrisierung der Messfunktion g wird gleich zu dem Pseudo-Inputs gemacht. Für g ergeben sich die Pseudo-Inputs $\xi = \{\xi_i \in \mathbb{R}^M\}_{i=1}^N$ und die Pseudo-Outputs $\upsilon = \{\upsilon_i \in \mathbb{R}^D\}_{i=1}^N$

Die Messsungen und das Pseudo-Trainingset werden in einen Algorithmus eingegeben. Die unbekante dynamische Gelichung und Messgleichung werden durch Gauß-Prozesse modelliert

$$\begin{aligned} x_{ti} &\sim GP_f(\mathbf{x}_{t-1} \mid \alpha, \beta_i) \\ y_{tj} &\sim GP_g(\mathbf{x}_t \mid \xi, \upsilon_j) \end{aligned} \tag{3}$$

2) Hyperparameter der Gaußprozesse: Die Funktionen f und g werden jeweils durch einen Gaußprozess mit einem Squared-Exponential-Kernel (SE-Kernel) approximiert. Beim verwenden eines SE-Kernels müssen drei Hyperparameter gewählt werden. Die Längenskalierung beschreibt die voraussichtliche Schwankung der approximierten Funktion, die Signalvarianz gibt an auf welchen Wertebereich abgebildet wird, die Auswertungsvarianz enthält Auskunft über die Messunsicherheiten. Pro Gaußprozess existiert jeweils ein 3-Tupel dieser Parameter. Diese werden im weiteren mit θ_f und θ_g bezeichnet.

3) Anfangszustand: Der Inferenz-Schritt des Algorithmus benötigt eine Initialisierung des nicht messbaren Zustandes x_1. Der Anfangszustand wird aus einer Normalverteilung mit Mittelwert μ_0 und Varianz Σ_0 gezogen. Die Variablen μ_0 und Σ_0 sind Teil des Parameterraumes.

B. Inferenz (E-Step)

Der E-Step des EM-Algorithmus soll, wie in Fig 2 dargestellt, anhand der Messung und den momenta-

nen Hyperparemetern, die Wahrscheinlichkeitsverteilungen $p(x_{1:T}|y_{1:T}, \Theta)$ berechnen. Der E-Step ist in zwei Abschnitte unterteilt, den Forward- und den Backward-Sweep.

1) Forward: Innerhalb des Forward-Sweeps finden zwei Updates statt. Das Timeupdate und das Filterupdate. Ziel des Forward-Sweeps ist es die Wahrscheinlichkeitsverteilung $p(x_t|y_{1:t})$ für alle $t <= T$ zu berechnen. Der Ablauf des Forward-Sweeps ist in Algorithmus 1 als Pseudo-Code dargestellt.

a) Time Update: Der Zeit-Update entspricht der einschritt prädiktiv Verteilung der latenten Zustandsraumvariablen $\mathrm{p}(\mathbf{x}_t|\mathbf{y}_{1:t-1})$ unter Verwendung von $\mathrm{p}(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$ als Prior. Das Satz von Bayes besagt:

$$p(x_t \mid y_{1:t-1}) = \int p(x_t \mid x_{t-1}) p(x_{t-1} \mid y_{1:t-1}) dx_{t-1} \quad (4)$$

Diese Verteilung kann, durch Auswertung der GP'S, analytisch berechnet und mit einer Gaussverteilung approximiert werden. Die Wahrscheinlichkeitsverteilung $p(x_t|x_{t-1})$ ergibt sich aus dem Gaußprozess GP_f. Dies berechnet sich wie im Appendix A erläutert, mit x_{t-1} als unsicherem Input. Im ersten Schritt ist der Prior durch die Hyperparameter: $p(x_0) = N(\mu_0, \Sigma_0)$, gegeben. In den weiteren Schritten wird das Ergebnis des letzten Filterupdates als Prior genutzt. Beide Wahrscheinlichkeitsverteilungen sind Normalverteilt. Das Integral kann gelöst werden. Das Ergebnis wird mittels Exact-Moment-Matching, wie in [2], als Gaußverteilung approximiert und wie folgt dargestellt: $p(x_t|y_{1:t-1}) \sim N(\mu_t^p, \mathbf{C}_t^p)$.

b) Filter-Update: Das Filterupdate nutzt die durch das Timeupdate erhaltenen Mittelwerte und (Ko-)Varianzen, um die a postori Warscheinlichkeiten $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ zu berechnen. Diese berechnen sich durch folgenen Gleichungen:

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = N(\mu_t^e, \mathbf{C}_t^e) \quad (5)$$
$$\mu_t^e = \mu_t^p + \mathbf{C}_{xy}\mathbf{C}_{yy}^{-1}(\mathbf{y}_t - \mu_y) \quad (6)$$
$$\mathbf{C}_t^e = \mathbf{C}_t^p - \mathbf{C}_{xy}\mathbf{C}_{yy}^{-1}\mathbf{C}_{yx} \quad (7)$$

Die Kovarianzen C_{yy}, C_{xy}, C_{yx} werden anhand der Grußprozesse berechnet. Eine genaue Beschreibung wie diese Kovarianzen berechnet werden befindet sich in [2].

Algorithm 1 Forward Algorithm

1: $p(x_1) \leftarrow N(\mu_0, \Sigma_0)$ Prior für $p(x_2)$
2: **procedure** TIME UND FILTER UPDATE
3: **for** <t = 2 : T> **do**
4: Hier wird Time Update gemacht
5: $p(x_t|y_{1:t-1}) \leftarrow N(\mu_t^p, C_t^p)$
6: Hier wird Filter Update gemacht
7: $p(x_t|y_{1:t}) \leftarrow N(\mu_t^e, C_t^e)$
8: Nächste Iteration $Prio \leftarrow p(x_t|y_{1:t})$ für $N(\mu_{t+1}^p, C_{t+1}^p)$

2) Backward: Der Backward-Sweep liefert dem M-Step, die für die Optimierung nötigen, Variablen. Um die zu optimierende Kostenfunktion aufzustellen werden folgende Variablen benötigt: Der Mittelwert und die Varianz für jedes x_t und das aktuell vorhergesagte Verhältnis zwischen x_t und x_{t+1}, die Cross-Kovarinz. Hierfür wird für jedes x_t die Wahrscheinlichkeitsverteilung

$$\gamma(x_t) = p(x_t|y_{1:T}) = N(\mu_t^s, C_t^s) \quad (8)$$

berechnet.

Dieser Schritt wird als Backwardsweep bezeichnet, da $\gamma(x_{t-1})$ über folgendes von $\gamma(x_t)$ abhängiges Integral berechnet wird:

$$\gamma(x_{t-1}) = \int p(x_{t-1}|x_t, y_{1:T}) \cdot \gamma(x_t) dx_t \quad (9)$$

Zur Berechnung des Integrals wird zunächst $\gamma(x_T) = N(\mu_T^e, C_t^e)$ gesetzt. Außerdem werden die Wahrscheinlichkeiten:

$$p(\mathbf{x}_{t-1}, \mathbf{x}_t|\mathbf{y}_{1:t-1}) = N\left(\begin{bmatrix}\mu_{t-1}^e \\ \mu_t^p\end{bmatrix}, \begin{bmatrix}\mathbf{C}_{t-1}^e & \mathbf{C}_{ep} \\ \mathbf{C}_{ep}^T & \mathbf{C}_t^p\end{bmatrix}\right) \quad (10)$$

benötigt. Die Werte für μ_{t-1}^e und C_{t-1}^e sind die Ergebnisse des Forward-Sweeps, μ_t^p und C_t^p stammen aus dem Time-Update und wurden auch bereits im Forward-Sweep berechnet. Die Matrix C_{ep} ist die Kovarianz $Cov[x_{t+1}, x_t|y_{1:t}]$. Die Berechnung dieser Kovarianz kann in [2] nachgelesen werden. Anhand dieser Wahrscheinlichkeiten lässt sich die Warscheinlichkeit $p(x_{t-1}|x_t, y_{1:t-1})$, mit $J_t = C_{ep} \cdot (C_t^p)^{-1}$ wie folgt ausdrücken:

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y}_{1:t-1}) = N(\mathbf{J}_{t-1}\mathbf{x}_t + \mu_{t-1}^e - \mathbf{J}_{t-1}\mu_t^p, \mathbf{C}_{t-1}^e - \mathbf{J}_{t-1}\mathbf{C}_{ep}^T) \quad (11)$$

Anschließend kann das Integral 9 gelöst und die Ausgänge des E-Steps bestimmt werden.

$$p(\mathbf{x}_{t-1}|\mathbf{y}_{1:T}) = \gamma(\mathbf{x}_{t-1}) = N(\mathbf{x}_{t-1}|\mu_{t-1}^s, \mathbf{C}_{t-1}^s) \quad (12)$$
$$\mu_{t-1}^s = \mu_{t-1}^e + \mathbf{J}_{t-1}(\mu_t^s - \mu_t^p) \quad (13)$$
$$\mathbf{C}_{t-1}^s = \mathbf{C}_{t-1}^e + \mathbf{J}_{t-1}(\mathbf{C}_t^s - \mathbf{C}_t^p)\mathbf{J}_{t-1}^T \quad (14)$$
$$R_t^s = \mathbf{C}_t^s \mathbf{J}_{t-1}^T \quad (15)$$

Die drei Ausgänge des E-Steps beschreiben die Cov$[x_t, x_{t-1}]$, die Mittelwerte der Normalverteilungen $p(x_t|y_{1:T})$ und die entsprechenden Varianzen. Diese sind in Form von R_t^s, μ_t^s und C_t^s gegeben.

C. Learning (M-Step)

Das Erlernen der Dynamik des Systems geschieht im M-Step. Die Funktionen f und g sind durch Gaußprozesse GP_f und GP_g approximiert. Die Gaußprozesse sind durch die

Hyperparameter charakterisiert. Erlernen der Systemdynamik bedeutet in diesem Fall, Optimieren der Hyperparameter. Es wird die Log-Maximimum-Likelihood Methode verwendet. Die optimierten Werte für Θ berechnen sich zu:

$$\theta_{max} = \underset{\Theta}{argmax} \quad Q \tag{16}$$

Mit

$$\begin{aligned} Q = \mathbb{E}_x[log(p(X,Y|\Theta))] = \mathbb{E}_x[log(p(x_1,\Theta))] + \\ + \mathbb{E}_x[\sum_{t=2}^{T} log(p(x_t|x_{t-1})) + \sum_{t=1}^{T} log(p(y_t|x_t))] \end{aligned} \tag{17}$$

In Gleichung 17 kann man sehen das die Kostenfunktion aus dem Einfluss des Transitionsmodells und dem Einfluss des Messmodels besteht.

Es werden die Hyperparameter, des Modells, welches die größte Wahrscheinlichkeit besitzt, die geschätzten Zuständen X unter Berücksichtigung der beobachteten Messungen Y zu approximieren, berechnet. Die neuen Hyperparameter werden an den E-Step übergeben und der EM-Algorithmus beginnt von vorne. Die beiden Schritte werden bis zu einem Abbruchkriterium wiederholt.

IV. REALISIERUNG

Im Laufe dieses Projekts wurden wir beauftragt den Algorithmus zu testen. Nachdem wir die Konzepte des latenten Zustandraumes und der latenten Funktionen im Rahmen dieses Papiers gelesen und verstanden hatten, haben wir versucht, den Code, der auf der Webseite der Autoren gegeben ist, zu verwenden. Dieser Versuch stieß auf ein Problem mit der Implementierung des Maximierungsalgorithmus, wir konnten das Schema "Expectation-Maximization" nicht finden. Stattdessen wurde ein anderer weg, die optimalen Hyperparameter zu erhalten, implementiert.

Anschließend haben wir versucht, den Algorithmus der Erwartungsmaximierung selbst zu implementieren. Dies war auch deshalb nicht erfolgreich, weil wir damals die Mathematik hinter dem Gaußschen Prozess der unbekannten Inputs und Funktionen nicht komplett geklärt hatten. Grund dafür war die Notwendigkeit, drei verschiedene Paper und Teile einer Dissertation durch zu arbeiten, um die richtigen Gleichungen zu finden.

Unabhängig von den Komplikationen stellten wir fest, dass der in [3] gegebene Code die richtige Vorhersage für die latenten Variablen und Funktionen unter dem (GP-ADF) Assumed Density Filtering with Gaussian Process (aus [2]) implementiert hat. Aufgrund des Zeitmangels nach dem Versuch der ersten beiden Ansätze, entschieden wir uns dazu, den ursprünglichen Code zu akzeptieren, auch wenn nicht exakt das Schema verwendet wird, welches wir zu analysieren versuchten. Der wesentliche Unterschied zwischen dem vorgestellten und dem tatsächlich genutztem Code, ist die Kostenfunktion. Diese Berücksichtigt im genutzten Code nicht explizit den Einfluss des Transitionsmodells.

Wie folgt präsentieren wir die Ergebnisse, die wir erreicht haben, indem wir nur zwei Parameter geändert und eine spezifische Anfangswerte der Hyperparameter implementiert haben. Dieser Anfangswert ist nach [4] von extremer Bedeutung für die Konvergenz des Erwartungsmaximierungsschemas und wurde in der Optimierung implementiert. Die zu variierenden Parameter waren die Anzahl der Pseudo-Trainingspunkte und die Tiefe des Optimierungsalgorithmus, um möglichst einige Konvergenzwerte zu finden.

V. AUSWERTUNG

Im folgenden werden die Simulationsergebnisse präsentiert und anhand dieser einige Probleme des Algorithmus veranschaulicht. Die Simulationen wurden mit dem auf der Website von Herrn Turner veröffentlichten GPIL-Algorirhmus berechnet.

Um den Algorithmus zu testen, haben wir zunächst versucht das System zu erlernen, welches auch von Ryan Turner in [3], verwendet wird. Die Systemgleichungen sind in Gleichungen 18.

$$\begin{aligned} x_{t+1} &= 3 * sin(3 * x_t) + \epsilon \\ y_t &= x_t + \nu \\ \epsilon &\sim N(0, 0.1^2) \\ \nu &\sim N(0, 0.1^2) \end{aligned} \tag{18}$$

Die wichtigsten Einstellungen, die im Programm festgelegt werden müssen, sind die Anzahl der Iterationen, die Anzahl der Pseudo-Targets und die Anzahl der Vergleichspunkte. Dies sind auch die Parameter,die ,die Dauer des Durchlaufs bestimmen. Es ist uns nicht gelungen eine Kombination dieser Werte zu finden die in akzeptabler Zeit (¡10 Stunden) zur Konvergenz der richtigen Funktionen führt.

Das beste Ergebnis haben wir mit 10 Pseudo-Targets, 500 Vergleichspunkten und 2000 Iterationen erreicht. Unser Rechner hat circa 9 Stunden zur Berechnung dieser Ergebnisse gebraucht. Die Ergebnisse werden im folgenden dargestellt.

A. Ergebnisse

Die Resultierenden GP's der Simulation sind 3 und 4. Die grauen Bereiche um die Funktion herum sind die Konfidenzbereiche, die orangenen Punkte die Pseudo-Targets.

Anhand dieser Abbildung kann man bereits einige Schlüsse ziehen. Zum einen wurde die richtige Frequenz des Sinus gefunden. Außerdem ist die Funktion g annähernd linear approximiert worden. Die Amplitude des Sinus ist jedoch zu gering und die Steigung der Funktion g ist ebenfalls zu

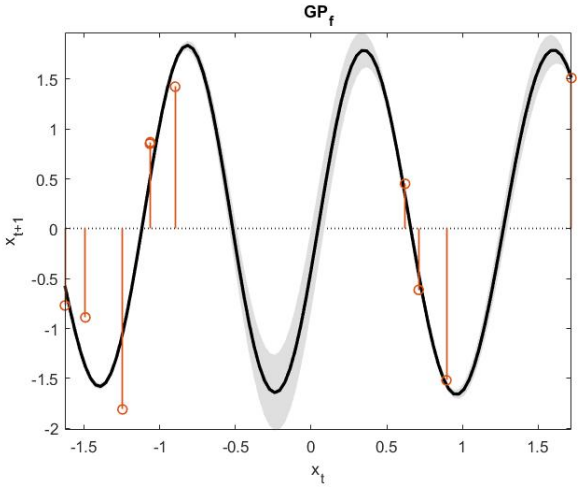

Fig. 3. Gp_f nach 2000 Iterationen

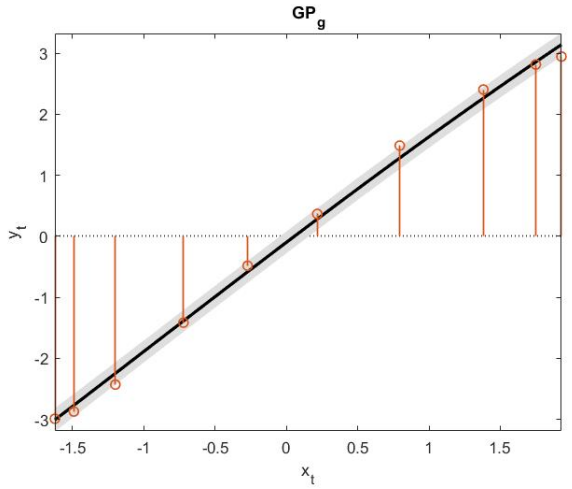

Fig. 4. GP_g nach 2000 Iterationen

gering. Diese Abweichungen lassen sich durch die Kombination der beiden Funktionen erklären. Zur Erinnerung: Der Algortithmus versucht in erster Linie die gemessenen Beobachtungen zu erklären. Abbildung 5 zeigt die (an den Stellen t=40-50) als Eingang des Algorithmus genutzten Messwerte (Ytrain) im Vergleich zu den mit unseren beiden GP's erzeugten Mittelwerten und Konfidenzbereichen für Y in den Zeitschritten 40-50. Die Messwerte liegen innerhalb des Konfidenzbereichs, der mit unseren GP's erzeugten Verteilung von y. Die beiden GP's sind also eine mögliche Erklärung der Messungen.

Unser eigentliches Ziel ist es jedoch nicht nur eine mögliche Erklärung der Messungen zu bekommen, sondern die Dynamik des Systems möglichst genau zu bestimmen. Um dies zu analysieren sind in Abbildung **??** die geschätzten Werte für x_t im Vergleich zu den tatsächlichen Werten des Systems für das gleiche Zeitintervall plottiert.

Es ist zusehen, dass die geschätzten und realen Werte recht weit aus einander liegen. Die realen Werte liegen zusätzlich

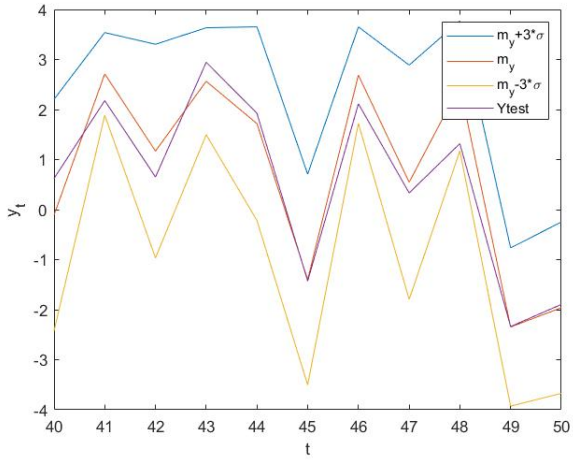

Fig. 5. m_y für t=[40,..,50]

außerhalb des Konfidenzbereichs.

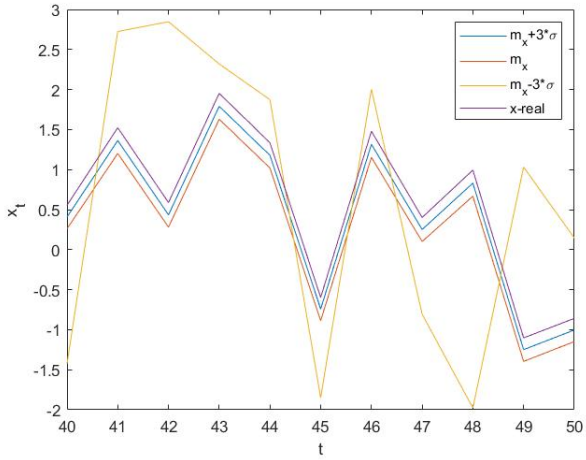

Fig. 6. m_x für t=[40,..,50]

Bei Betrachtung von Abbildung 3 fällt auf, dass die Pseudo-Targets nicht auf der Funktion liegen. Dies ist ein klares Anzeichen dafür, dass der Algorithmus, nach 2000 Iterationen noch nicht konvergiert ist. Eine Möglichkeit um bessere Ergebnisse zu erzielen besteht sicherlich darin den Algorithmus länger laufen zulassen.

VI. Bewertung

APPENDIX

A. Gaußprozesse mit unsicheres Input

Hier wird der Gaußprozess der Messgleichung 3 veranschaulicht. In [2] wird folgendes Modell abgeleitet.

Das Gaußprozess zieht aus den verrauschten Beobachtungen $[\mathbf{Y}]$ die latente Funktion g der Gleichung 3. Ein

Gaußprozess wird vollständig definiert durch einen Mittelwert $m(.)$ und eine Covarianzmatrix $k(.,.)$, dem Kernel. Die Modellierung in diesem Paper verwendet den "Square Exponential Kernel" (SE) Kernel. (**??**) und ergibt folgenden posteriori Verteilung von der Wert der Funktion $h_\star = h(x_\star)$ für gegebenes $x_\star$

$$p(g(x_\star) \mid x_\star) = \sim GP(m_g(x_\star), \sigma_g^2) \tag{19}$$
$$m_g(x_\star) = \mathbf{k}_\star^T(\mathbf{K} + \Sigma_\nu \mathbf{I})^{-1}\upsilon \tag{20}$$
$$\sigma_g^2 = k_{\star\star} - \mathbf{k}_\star^T(\mathbf{K} + \Sigma_\nu \mathbf{I})^{-1}\mathbf{k}_\star \tag{21}$$

mit $\mathbf{k}_\star = k(\xi, x_\star)$, $k_{\star\star} = k(x_\star, x_\star)$ und wo $\mathbf{K} = K_{ij} = k(\xi_i, \xi_j)$. Außerdem $\xi = [\xi_1, ..., \xi_N]$ sind die pseudo-training Inputs und $\upsilon = [\upsilon_1, ..., \upsilon_N]$ sind die pseudo-training ziele von der Funktion g

B. Vorhersage für unsichere Inputs

Wie vorher erwähnt muss im Zeit-update die Gleichung (4) gelöst werden. In diesem Schema ist es wichtig zu beachten dass der test-Input $x_\star$ unsicher ist, deswegen lässt sich aus einer normaler Verteilung berechnet $x_\star \sim N(\mu, \Sigma)$.

Hier wird die $p(g(x_\star) \mid \mu, \Sigma)$ bei einer Gaussian aproximiert $N(\mu_\star, \sigma_\star^2)$

$$y_t = \sim GP_g(\mathbf{x}_t \mid \xi, \upsilon) = p(g(x_\star) \mid \mu_\star, \sigma_\star) \tag{22}$$
$$\mu_\star = \gamma^T l \tag{23}$$
$$\sigma_\star{}^2 = \gamma^T \mathbf{L}\gamma + \sigma_h^2 - sp(\gamma\mathbf{L}) - \mu_\star \tag{24}$$

mit $\gamma = (\mathbf{K} + \Sigma_\nu \mathbf{I})^{-1}$ und sp(.) ist die Spur und $\mathbf{L}$ kann in Siete 4 von [2] gelesen werden.

ACKNOWLEDGMENT

REFERENCES

[1] C. E. Rasmussen und C. K. I. Williams, *Gaussian Processes for Machine Learning*, the MIT Press, 2006.

[2] Deisenroth, M. P., Huber, M. F., and Hanebeck, U. D. *Analytic moment-based Gaussian process filtering*, Proceedings of the 26th International Conference on Machine Learning, 2009.

[3] Ryan Turner, Marc Deisenroth *State-Space Inference and Learning with Gaussian Processes*, AIStats, 2010.

[4] Ryan Darby Turner *Gaussian processes for state space models and change point detection*, PhD thesis, University of Cambridge, Department of Engineering, Cambridge, UK, 2011.