Forschungsbeitrag zur Künstlichen Intelligenz

Julia Vinogradska absolvierte ein Studium der Mathematik mit Nebenfach Informatik an der Universität Stuttgart. In ihrer Masterarbeit „Automorphismen von Graphgruppen“ befasste sie sich mit partiell kommutativer Gruppen, die unter anderem bei der Analyse von nebenläufigen Prozessen Verwendung finden. Seit 2014 ist Julia Vinogradska in der Zentralen Forschung der Robert Bosch GmbH als Industriedoktorandin tätig. In ihrer Promotion beschäftigt sie sich mit der Analyse von Machine Learning Verfahren, speziell Reinforcement Learning Methoden, die automatisiert die Regelung von unbekannten Systemen lernen. Sie entwickelte unter anderem automatisierte Verfahren, die Stabilitätsgarantien für gelernte Regler und Dynamiken berechnen. Nach Abschluss Ihrer Promotion wird Julia Vinogradska weiterhin in der Zentralen Forschung der Robert Bosch GmbH, im kürzlich gegründeten Bosch Center for Artificial Intelligence (BCAI) daran arbeiten, Machine Learning Verfahren zu analysieren und ihre Funktionalität mit theoretischen Garantien abzusichern.

Julia Vinogradska

Lernende Regler und Stabilitätsgarantien

„Computer languages of the future will be more concerned with goals and less concerned with procedures specified by the programmer“ – was Marvin Minsky, ein Pionier der künstlichen Intelligenz, Ende der 60er Jahre für Programmiersprachen vorhersagte gilt heutzutage nicht nur für die Art und Weise Computern bestimmte Anweisungen zu übermitteln, sondern auch für die grundlegende Herangehensweise an Probleme. In den letzten Jahren erlebt die künstliche Intelligenz eine Renaissance, beflügelt durch einige Durchbrüche im Bereich des maschinellen Lernens (engl. *machine learning*) und die deutlich gestiegene Rechenkapazität, die zur Verfügung steht. Es ist nun möglich komplexe Aufgabenstellungen nicht zu lösen, indem man speziell auf die Problemstellung zugeschnittene Vorgehensweisen entwickelt und diese in allen Einzelheiten ausgestaltet, um sie dann auf einem Rechner auszuführen, sondern den Computer selbst lernen zu lassen welche Vorgehensweise für das vorliegende Problem optimal ist und welche Informationen dafür benötigt werden.

Dieser technische Fortschritt eröffnet neue Möglichkeiten, die „klassischen“, wissensbasierten Ingenieurswissenschaften mit datengetriebenen Ansätzen der künstlichen Intelligenz zusammenzubringen, um komplexere Probleme effizient lösen zu können. Ein Feld, das von dieser Kombination von maschinellem Lernen und klassischen Methoden immens profitiert, ist das Gebiet der Regelungstechnik. In der Regelungstechnik geht es darum, technische Systeme durch gezieltes Eingreifen mittels einer oder mehrerer Stellgrößen so zu beeinflussen, dass sie ein bestimmtes, vorgegebenes Verhalten aufweisen. So möchte man z.B. für eine Klimaanlage ein Zimmer auf einer gewünschten Temperatur halten, indem man die Menge und Temperatur der zuströmenden Luft variiert, oder bei einem Tempomaten das Fahrzeug auf einer vom Fahrer gewählten Geschwindigkeit halten, indem man die Stellung des Gaspedals abhängig von aktueller Geschwindigkeit wählt.

Wie kommt man nun von der Problemstellung zu einem funktionierenden Regler? Im Wesentlichen lässt sich das Vorgehen in drei Schritte unterteilen: 1. Herleitung eines Modells, das das Systemverhalten beschreibt, 2. Synthese eines Reglers unter Zuhilfenahme des Modells und 3. Nachweis der Stabilität, also der korrekten Funktion des Reglers

Kurzzusammenfassung:

In den letzten Jahren wurden in der künstlichen Intelligenz einige Durchbrüche erreicht. Dieser technische Fortschritt ermöglicht es, regelungstechnische Fragestellungen mit neuen Ansätzen anzugehen. Von der Systemidentifikation über die Reglersynthese bis zum Nachweis der Stabilität kann die künstliche Intelligenz dazu beitragen, die Probleme möglichst effizient und kostengünstig zu lösen und dabei den manuellen Aufwand zu minimieren.

unter der Annahme, dass das echte System sich wie das Modell verhält. Doch was hat das alles mit künstlicher Intelligenz zu tun? Zunächst einmal sind die drei oben beschriebenen Schritte klassische Problemstellungen aus den Ingenieurswissenschaften und man könnte meinen, dass sie heutzutage nahezu komplett gelöst wären und wir auf einen reichhaltigen Fundus an Modellen, Vorgehensweisen für Reglersynthese und den Nachweis der Stabilität zurückgreifen könnten. Allerdings ist jeder der drei Schritte für sich genommen ein schweres Problem das meist nicht mit Standardmethoden hinreichend gut gelöst werden kann. Hierbei kann nun künstliche Intelligenz helfen, jeden der Schritte möglichst effizient und kostengünstig zu lösen.

Um genau nachzuvollziehen, wie künstliche Intelligenz uns dabei helfen kann Systeme zu regeln, wollen wir anhand eines Beispiels aus der Praxis den Weg von Problemstellung bis zum produktreifen Regler nachvollziehen. Die Herausforderungen bei der Regelung von komplexen Systemen werden gut am Beispiel des Verbrennungsmotors erkennbar. Ein moderner Verbrennungsmotor hat eine Vielzahl an mechanischen Komponenten wie z.B. Ventilen und Klappen, die geregelt werden müssen um einen optimalen Betrieb zu gewährleisten. Dabei haben wir eine Vielzahl an Anforderungen an das Verhalten des Motors: wenig Treibstoff verbrauchen, dabei aber so viel Leistung wie möglich erbringen, möglichst wenig Schadstoffe ausstoßen, leise und vibrationsarm zu laufen und vieles mehr. Gleichzeitig finden in dem Motor komplexe Vorgänge statt, die nur sehr schwer zu beschreiben oder überhaupt zu messen sind. Gehen wir nun die einzelnen Entwicklungsschritte für den Regler durch.

Wollen wir unseren Motor betreiben, müssen wir zunächst das Verhalten des Systems verstehen und darstellen. In anderen Worten: wir brauchen ein Modell, das den aktuellen Systemzustand und die Stellgröße auf den im nächsten Zeitschritt folgenden Zustand abbildet. Man kann nun Verbrennungsmotoren und alle ihre Komponenten ausgiebig studieren und aus bekannten Wirkzusammenhängen und Naturgesetzen Modelle für ihr Verhalten herleiten. Dafür braucht man Experten, die sich mit den Eigenschaften der einzelnen Komponenten auskennen und muss diese vielen einzelnen Informationen zu einem großen Gesamtmodell miteinander kombinieren. Um für einen speziellen Motor ein Modell herzuleiten, reicht allgemeines Expertenwissen allerdings nicht aus. Die verwendeten Materialen, die Bauart des Motors und auch des dazugehörigen Fahrzeugs beeinflussen alle das Verhalten des Systems. Um diese Einflüsse zu berücksichtigen, gibt es in den Gleichungen des Modells Parameter, deren Werte auf das spezifische System angepasst werden müssen. Dazu müssen die Experten

das System in verschiedenen Modi betreiben und das Verhalten des Motors vermessen. Aus diesen Messungen schätzen die Experten dann passende Parameter für den vorliegenden Typ des Motors. Dies erfordert einen sehr hohen Zeitaufwand und auch Messungen an solch einem System sind mit hohen Kosten verbunden. Hinzu kommt, dass die ermittelten Parameter trotz aller Messungen inhärent Schätzungen sind, die niemals das echte Verhalten des Systems absolut korrekt abbilden können. Eine weitere Schwierigkeit ist, dass die Modellbildung an sich eine große Herausforderung darstellt und man nicht davon ausgehen kann, dass man zu jedem Zeitpunkt genügend Experten mit gleich hohem Kenntnisstand zur Verfügung hat.

An dieser Stelle kommt künstliche Intelligenz ins Spiel. Statt ein Modell auf Basis von physikalischem Wissen und Expertise herzuleiten und die Parameter des Modells mit Hilfe von Messwerten zu schätzen, kann man direkt aus den Messungen ein Modell für das Verhalten des Systems inferieren. Dabei wollen wir nicht eine gewisse Form von Modell vorgeben und die Parameter anpassen, das entspricht ja gerade dem klassischen Ansatz, Systemwissen in die Herleitung des Modells zu stecken und sich für eine bestimmte parametrische Form zu entscheiden. Stattdessen wollen wir einen möglichst flexiblen Ansatz wählen und nur sehr allgemeine Annahmen, wie z.B. Glattheit des Modells, treffen. Außerdem wollen wir nicht

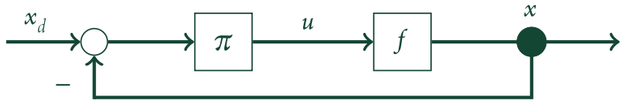


Abbildung 1:
Regelkreis mit Modell f, Systemzustand x, Zielzustand x_d, Regler π und Stellgröße u. [1]

nur eine Schätzung des Systemverhaltens ermitteln und uns anschließend komplett auf diese Schätzung verlassen und so tun, als ob die Schätzungen unseres Modells das System perfekt beschreiben. Unter der Annahme der Glattheit haben wir nämlich die zusätzliche Information, dass das System sich nicht willkürlich ändert. Wenn wir also Messungen haben und unser Modell darauf trainieren, an den Messstellen das beobachtete Verhalten zu reproduzieren, dann wissen wir zusätzlich, dass das System sich in der Nähe der Messungen ähnlich verhält. Unser Modell kann also in der Nähe der Messpunkte viel zuverlässigere Aussagen machen als bei Zuständen, die sehr weit weg sind von den Stellen, an denen wir das echte Systemverhalten beobachtet haben. Diese Information wollen wir in unserem Modell abbilden und statt reinen Schätzungen noch Aussagen über die Unsicherheit der Schätzung des Modells treffen. Mathematisch lässt sich das durch stochastische Prozesse, in unserem Anwendungsfall Gauß-Prozesse darstellen. Ein Gauß-Prozess gibt uns eine Verteilung über Funktionen, die die Messungen plausibel erklären würden. Wir beginnen mit einer Verteilung, unter der alle Funktionen wahr-

scheinlich sind, die unsere ursprünglichen Annahmen wie z.B. Glattheit erfüllen. Dann konditionieren wir diese Verteilung über Funktionen auf die Messungen, die wir vom echten System haben. Das kann man sich so vorstellen, dass wir alle Funktionen sehr unwahrscheinlich machen, die nicht so gut zu unseren Messungen passen und diejenigen Funktionen aus unserer Verteilung sehr hohe Wahrscheinlichkeit bekommen, die unsere Messdaten sehr gut erklären.

Unser Modell hat einige besondere Eigenschaften. Zunächst mal bekommen wir zu jedem Paar von Systemzustand und Stellgröße nicht nur eine Schätzung des nächsten Zustandes, sondern auch noch eine Aussage über die Unsicherheit dieser Schätzung. Haben an einer bestimmten Stelle alle Funktionen, die laut unserem Modell sehr wahrscheinlich sind, ähnliche Funktionswerte, dann ist unsere Schätzung sehr sicher. Unterscheiden sich die Werte aller plausiblen Funktionen, die die Messungen erklären würden sehr stark, dann können wir durch unsere Messungen nicht sehr viel über die Stelle sagen und die Schätzung unseres Modells ist sehr unsicher. Konkret liefert unser Gauß-Prozess zu jedem Eingang von Systemzustand und Regelsignal einen normalverteilten Ausgang für den Systemzustand im nächsten Zeitschritt, wobei der Erwartungswert dieser Verteilung als unsere Schätzung fungiert und die Varianz der Verteilung aussagt, wie sicher oder unsicher die Schätzung an dieser Stelle ist. Eine weitere Besonderheit unseres Modells ist, dass es umso reichhaltiger und komplexer wird, je mehr Messungen wir haben. Das steht im Gegensatz zu klassischen parametrischen Modellen, wo man sich vorher auf die Komplexität und damit auch die Aussagekraft des Modells festlegt, indem man die Anzahl der Parameter wählt. Unser Modell kann sich flexibel auf verschiedenste Dynamiken anpassen, ohne dass wir den Ansatz ändern müssen. Daher können wir an dieser Stelle erheblich Arbeitsaufwand sparen, da wir keine Experten brauchen, die eine geeignete parametrische Form für jedes einzelne System herleiten. Gleichzeitig profitieren wir von einer erhöhten Modellgüte, da wir explizit die Unsicherheit unserer Schätzung abbilden.

Nun haben wir also mit Hilfe von künstlicher Intelligenz ein Modell für unseren Motor erstellt und dabei den manuellen Aufwand deutlich reduziert. Was können wir mit so einem Gauß-Prozess-Modell anfangen? Wie können wir die Vorteile wie z.B. die zusätzliche Information über die Unsicherheit der Modellschätzungen bei der Reglersynthese nutzen? Können wir auch hier mit Hilfe von künstlicher Intelligenz den Arbeitsaufwand verringern? Das Teilgebiet „verstärkendes Lernen“ (engl. *reinforcement learning*) der künstlichen Intelligenz befasst sich mit Problemen, die wir auch bei der Synthese eines Reglers lösen müssen. Es geht um sequenzielle Entscheidungsprobleme, die man trotz unbekanntem oder unsicherem (also stochastischem) Systemverhalten optimal zu lösen versucht. Solche Probleme lassen sich formal als Markov-Entscheidungsprozesse (engl. *Markov decision process [MDP]*) beschreiben. Ein wesentlicher Teil eines Markov-Entscheidungsprozesses ist eine Belohnung (engl. *reward function*), die in jedem Zeitschritt ausgeschüttet wird und den aktuellen Zustand bewertet. Ziel in einem Markov-Entscheidungsproblem ist es, eine Entscheidungsstrategie (engl. *policy*) zu finden, die die erwartete Summe an Belohnungen über alle Zeitschritte maximiert. In anderen Worten: wir suchen einen Regler, der unser System möglichst optimal betreibt.

Es gibt viele Ansätze, solche Markov-Entscheidungsprobleme anzugehen. Konzentrieren wir uns hier auf einen Ansatz, der die Vorteile des Gauß-Prozess-Modells ausnutzen kann und sich somit besonders gut für unsere Motorregelung eignet. Die Idee ist, eine parametrische Form für den Regler zu wählen. Diese Form kann allerdings sehr flexibel sein, z.B. ein RBF-Netz oder die Fourier-Entwicklung einer beliebigen Funktion. Unsere einzigen Bedingungen sind, dass die Funktion sich durch einen endlichen Vektor θ parametrieren lässt und ihre Werte differenzierbar vom Zustand und den Parametern θ abhängen. Unser Ziel ist es, möglichst gute Parameter zu finden. Möglichst gut bedeutet dabei, dass dieser Parametervektor θ einen Regler beschreibt, mit dem wir beim Betrieb besonders viel Belohnung bekommen. Wie kann man nun so einen guten Regler mit Hilfe von künstlicher Intelligenz lernen?

„Dank künstlicher Intelligenz können Computer die zur Problemlösung nötigen Informationen selbst extrahieren.“

Julia Vinogradska

Nehmen wir an, wir hätten einen Parametervektor θ. Wie finden wir raus, ob dieser Parametersatz „gut“ ist, oder nicht? Mithilfe unseres Modells können wir den durch θ parametrierten Regler ausprobieren. Dazu wählen wir einen Startzustand und berechnen das Regelsignal für diesen Zustand. Diese beiden Größen nutzen wir als Eingang in unser Modell, um den nächsten Systemzustand zu schätzen. Für diesen Zustand können wir nun wieder das Regelsignal berechnen und mit unserem Modell einen weiteren Schritt machen. Zusammengefasst können wir also für einen gegebenen Parametervektor θ das Systemverhalten mit Hilfe unseres Modells simulieren.

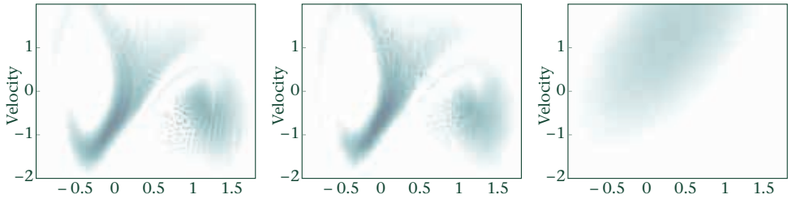
Wir können also den Systemzustand bei Anwendung unseres Reglers schätzen und folglich auch abschätzen, wie viel Belohnung wir während des Betriebs aufsammeln können. Betrachten wir einmal genauer, wie das funktioniert. Unser Systemmodell ist wie oben beschrieben durch einen Gauß-Prozess gegeben. Das bedeutet, dass wir nicht eine einzelne Funktion haben, die das Verhalten des Systems beschreibt, sondern eine Verteilung über plausible Funktionen. Die Vorteile

dieses Ansatzes wollen wir für unsere Reglersynthese ausnutzen, da die Information, ob unsere Schätzung des Verhaltens sicher oder eher unsicher ist wichtig ist, um die möglichen Belohnungen, die wir mit dem Regler bekommen können, korrekt zu schätzen. Um dieses Potential auszunutzen, müssen wir also die gesamte Verteilung über mögliche Funktionen, die unsere Messdaten erklären, berücksichtigen. Zu einem gegebenen Zustand und Regelsignal berechnet unser Modell also nicht einen Folgezustand, sondern eine Normalverteilung über Folgezustände. Mit dieser Normalverteilung müssen wir fortan rechnen. Die Schwierigkeit liegt nun darin, dass wir im nächsten Zeitschritt nicht mehr den genauen Zustand kennen, sondern eben nur eine Verteilung über Systemzustände. Für jeden einzelnen Zustand können wir das Regelsignal berechnen und mit unserem Modell den Folgezustand schätzen. Haben wir aber eine Zustandsverteilung, müssen wir für alle möglichen Zustände in dieser Verteilung das Regelsignal berechnen und über alle mit Hilfe des Modells berechneten Verteilungen von Folgezuständen mitteln. Genauer gesagt müssen wir den Erwartungswert der Modellvorhersage bezüglich der aktuellen Zustandsverteilung berechnen. Leider ist dies für die meisten interessanten Funktionen nicht in geschlossener Form möglich. Wir müssen daher die Verteilung des Folgezustands approximieren. Dafür gibt es verschiedene Methoden. Eine Möglichkeit ist, den Erwartungswert und die Varianz dieser Verteilung zu berechnen. Dies ist exakt möglich, wenn der aktuelle Zustand einer Normalverteilung folgt. Wir könnten also die Verteilung des Folgezustands durch eine Normalverteilung approximieren, deren erste zwei Momente wir ja exakt berechnen können. Diese Methode ist als *Moment Matching* bekannt. Allerdings ist der Folgezustand nicht normalverteilt, für den nächsten Zeitschritt wird diese Approximation also nicht den korrekten Erwartungswert und Varianz liefern. Wir verfolgen daher einen anderen Ansatz.

„Gelernte Modelle machen ihre inhärente Unsicherheit explizit sicht- und quantifizierbar.“

Julia Vinogradska

Da wir an einem Erwartungswert interessiert sind, müssen wir also ein bestimmtes Integral approximieren. Eine weit verbreitete Methode, um Integrale zu approximieren ist numerische Quadratur. Numerische Quadratur schätzt den Wert eines Integrals durch eine gewichtete Summe von Funktionsauswertungen. Die Stellen, an denen die Funktion ausgewertet werden soll und die entsprechenden Gewichte sind dabei durch die Quadraturregel gegeben. Diese Regeln sind so konstruiert, dass sie mit einer minimalen Anzahl an Funktionsauswertungen alle Polynome bis zu einem gewissen Höchstgrad exakt integrieren. Dadurch liefern solche Quadraturregeln sehr gute Schätzungen für die Integrale glatter Funktionen, da diese



(a) Our Numercical Quadrature (b) Monte Carlo Sampling (c) Moment Matching

beliebig genau durch Polynome angenähert werden können. Diese Vorgehensweise hat den zusätzlichen Vorteil, dass der Näherungsfehler sich nach oben hin abschätzen lässt. Dieser ist nämlich höchstens so groß wie der Näherungsfehler des Interpolationspolynoms und kann beispielsweise über die Taylor-Entwicklung der Funktion abgeschätzt werden.

Verwenden wir numerische Quadratur, um die Verteilung des Folgezustands zu approximieren, ergibt sich für unseren Gauß-Prozess eine Approximation als gewichtete Summe von Gauß-Glocken. Wenn man das Systemverhalten für mehrere Zeitschritte simuliert, hat man also zu jedem Zeitpunkt eine gewichtete Summe von Gauß-Glocken als Zustandsverteilung. Da sich unser Modell während der Simulation nicht verändert, Modellvorhersagen an derselben Stelle also immer gleich bleiben, ändern sich in unserer Näherung die Gauß-Glocken in dieser Summe nicht. Stattdessen ändern sich von Zeitschritt zu Zeitschritt nur die Gewichte der Gauß-Glocken. Insgesamt bekommen wir also eine simple, leicht interpretierbare Näherung für die Zustandsverteilung. Die Gewichte für unsere Gauß-Glocken können außerdem durch eine Matrix-Vektor-Multiplikation sehr effizient berechnet werden. Solche Rechenoperationen können sehr gut parallelisiert werden und z.B. auf einer Grafikkarte sehr schnell berechnet werden.

Mit numerischer Quadratur können wir also effizient die Verteilung des Systemzustands mit hoher Genauigkeit approximieren. Damit können wir also unser probabilistisches Gauß-Prozess-Modell des Motors optimal nutzen, um das Verhalten des Systems inklusive der Unsicherheit des Modells zu simulieren.

Abbildung 2:
Zustandsverteilungen mit Gauß-Prozess-Modell nach einigen Zeitschritten. Abb. (a) zeigt die Approximation durch numerische Quadratur, Abb. (b) 100 000 Samples aus der wahren Zustandsverteilung, und Abb. (c) Approximation durch Moment Matching. [2]

„Stabilitätsaussagen sind von fundamentaler Bedeutung für den Einsatz von künstlicher Intelligenz in der Regelungstechnik.“

Julia Vinogradska

Wie lernt der Computer nun, welcher Parametervektor Θ all unsere Anforderungen an den Motor optimal erfüllt? Zunächst müssen wir eine Belohnungsfunktion definieren. Diese soll unsere Anforderungen an das System widerspiegeln. So können wir z.B. definieren, dass niedriger Kraftstoffverbrauch, wenig Schadstoffausstoß und niedrige Lautstärke eine hohe Belohnung ausschütten. An dieser Stelle können wir das Problem dem Computer überlassen, der lernt, wie der Regler unsere Stellgrößen manipulieren muss, um möglichst viel Belohnung einzuholen. Dazu beginnt der Computer erstmal mit einem zufälligen Parametervektor Θ. Für diesen Parametervektor simulieren wir mit unserem Gauß-Prozess mit Hilfe von numerischer Quadratur das Systemverhalten, berechnen also die Verteilung des Zustands für alle Zeitschritte bis zu unserem Zeithorizont. Da die Zustandsverteilungen bekannt sind, können wir nun den Erwartungswert der ausgeschütteten Belohnung in jedem Zeitschritt berechnen und somit die erwartete Belohnung abschätzen, die wir mit diesem Regler einsammeln können. Wir können also dem Parametervektor Θ eindeutig eine erwartete Belohnung zuordnen und so die Güte dieses Parametersatzes bewerten. Wie kommen wir nun von dem ursprünglichen Vektor Θ zu einem optimalen Parametervektor? Dazu müssen wir bedenken, dass unser Modell und unser Regler glatt sind, also differenzierbar von Θ abhängen. Daher können wir nicht nur die Güte eines Parametervektors bewerten, sondern auch noch bestimmen, wie sich die kumulierte Belohnung ändert, wenn wir Θ ändern. Mit einem Gradientenaufstieg können wir also ausgehend Θ von einen optimalen Parametervektor Θ_* bestimmen. Damit haben wir also einen optimalen Regler gelernt und Schritt zwei auf dem Weg zum produktreifen Regler abgeschlossen.

In der Praxis trennt man die Schritte des Lernens von Modell und Regler nicht so strikt, wie wir es eben beschrieben haben. Stattdessen iteriert man Schritte von Verbesserung des Modells und Lernen eines optimalen Reglers gegeben das Modell. Man beginnt also mit einem beliebigen Parametersatz. Diesen wendet man an und misst dabei das Systemverhalten. Mit den so gewonnenen Daten trainiert man das Modell. Auf Basis dieses Modells lernt man einen optimalen Parametervektor Θ_*. Diesen kann man nun auf dem echten System anwenden und dabei wieder das Verhalten messen, das Modell verbessern und für das verbesserte Modell einen optimalen Regler lernen. Ein Durchgang von Regler Anwenden, Modell Verbessern und neuen Regler Lernen heißt *Episode*. Man berechnet also immer weitere Episoden, bis der Regler konvergiert. In der Praxis reicht meist eine einstellige Zahl von Episoden aus. Dieses Vorgehen hat den großen

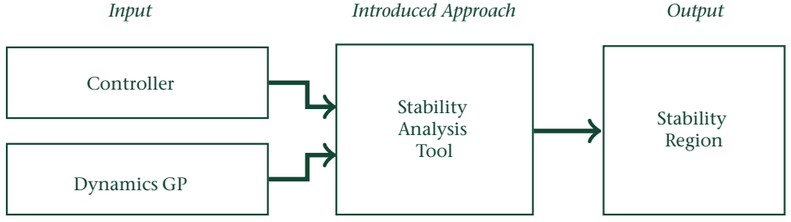


Abbildung 3:
Automatisches Verfahren zum Nachweis der Stabilität. Eingänge sind ein Gauß-Prozess-Modell und ein (gelernter) Regler, der Ausgang ein Bereich des Zustandsraums, in dem das System mit hoher Wahrscheinlichkeit asymptotisch stabil ist. [1]

Vorteil, dass man nur Messungen des Systemverhaltens in interessanten Bereichen generiert und dadurch mit vergleichsweise wenigen Messungen zum Ziel kommt.

Mit Hilfe von künstlicher Intelligenz haben wir also ein Modell für das Verhalten unseres Systems und einen Regler, der alle unsere Anforderungen an das Systemverhalten möglichst gut umsetzt, lernen können. Bis auf die Vorgabe von allgemeinen Eigenschaften des Systems wie z.B. Glattheit, die Wahl einer parametrischen Form des Reglers und die Definition von Anforderungen an das Systemverhalten in Form einer Belohnungsfunktion ist alles automatisch und ohne manuellen Aufwand vonstatten gegangen.

Kommen wir nun zum letzten Schritt: dem Nachweis der Stabilität des Systems. Dieser Schritt ist von fundamentaler Bedeutung für unser Vorhaben. Den Regler, den uns der Computer auf dem Silbertablett serviert, können wir in unsere Motorsteuerung einbauen und so viel wir wollen im Betrieb testen. Allerdings werden wir niemals in der Lage sein alle theoretisch möglichen Systemzustände zu testen, denn davon gibt es (überabzählbar) unendlich viele. Gerade bei solch sicherheitskritischen Anwendungen wie einem Verbrennungsmotor wollen wir aber auf jeden Fall sicher sein, dass unser Regler in jeder Situation korrekt funktioniert. Da wir unser Modell und Regler automatisch mit Hilfe von künstlicher Intelligenz gelernt haben, wollen wir auch bei dem Stabilitätsnachweis möglichst automatische Methoden nutzen, um diesen Vorteil voll ausnutzen zu können.

Dabei ist zu bedenken, dass unser Gauß-Prozess-Modell eine Verteilung über Funktionen definiert, die das beobachtete Systemverhalten erklären. Daher muss unser

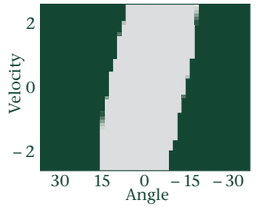


Abbildung 4:
Beispiel eines automatisch berechneten Stabilitätsnachweises. Die Farbe codiert die Wahrscheinlichkeit, dass der Regler das System von dem Startpunkt aus stabilisiert. Grau bedeutet Wahrscheinlichkeit 1, grün Wahrscheinlichkeit 0. [1]

Stabilitätsnachweis auch mit dieser Verteilung umgehen können. Wir können also nicht erwarten, dass wir eine deterministische Aussage bekommen werden, wenn wir das Systemverhalten selbst probabilistisch beschreiben. Stattdessen zielen wir auf eine Aussage über die Wahrscheinlichkeit, dass unser Regler korrekt funktioniert. Genauer gesagt möchten wir eine Mindestwahrscheinlichkeit vorgeben, mit der unser Regler das System im gewünschten Zustand stabilisiert. Wir suchen nun alle Startzustände, für die der Regler uns mit der vorgegebenen Mindestwahrscheinlichkeit zum Zielzustand bringt.

Um dieses Ziel zu erreichen, rufen wir uns nochmal ins Gedächtnis, dass die Systemzustände zu jedem Zeitpunkt bei unserem Gauß-Prozess Modell nicht exakt bekannt sind, sondern wegen der Modellunsicherheit durch eine Verteilung gegeben sind. Wir haben bereits gesehen, dass man diese Zustandsverteilungen nicht analytisch in geschlossener Form berechnen kann und stattdessen gut mit numerischer Quadratur approximieren kann. Allerdings machen wir bei dieser Approximation einen Fehler, den wir unbedingt berücksichtigen müssen, wenn wir belastbare Garantien geben wollen. Glücklicherweise können wir obere Schranken für den Quadraturfehler berechnen. Wie kommen wir nun von dem Gauß-Prozess und dem gelernten Regler zu einer Stabilitätsaussage? Zunächst einmal können wir für eine gegebene Verteilung des Startzustands die Verteilung des Systemzustands zu jedem Zeitpunkt mit numerischer Quadratur schätzen. Bei jeder dieser Schätzungen machen wir einen Fehler, den wir nach oben hin begrenzen können. Doch wie pflanzt sich dieser Fehler fort, wenn wir weitere Zeitschritte vorwärts rechnen? Wie wir gesehen haben, ergibt unsere Approximation mit numerischer Quadratur eine Näherung der Zustandsverteilung durch eine gewichtete Summe von Gauß-Glocken. Die Gauß-Glocken ändern sich dabei nicht über die Zeit, sondern nur die dazugehörigen Gewichte. Die Gewichte zu einem gewissen Zeitpunkt können dabei berechnet werden, indem man die Gewichte des vorherigen Zeitschritts mit einer bestimmten, festen Matrix multipliziert. Daher können wir leicht abschätzen, wie sich ein Approximationsfehler über mehrere Zeitschritte ausbreitet. Wenn wir also die Zustandsverteilung bis zum Zeithorizont berechnen, können wir nicht nur die Wahrscheinlichkeit, dass der Regler uns zum Zielzustand bringt schätzen, sondern auch den Fehler, den wir bei dieser Schätzung machen, nach oben hin beschränken. Daher können wir eine gewisse Mindestwahrscheinlichkeit garantieren, mit der der Regler tatsächlich korrekt funktionieren muss. Doch ist diese Abschätzung nicht sehr konservativ? Immerhin nehmen wir bei jedem Zeitschritt den größtmöglichen Fehler an, was doch sehr unwahrscheinlich ist. Diese Beobachtung ist korrekt, in der Praxis funktioniert der Regler meist besser, als unser Funktionsnachweis annimmt. Allerdings ist es im Allgemeinen nur durch solche Abschätzungen, die vom schlechtesten Fall ausgehen, möglich belastbare Stabilitätsgarantien zu geben. Bei unserem Vorgehen haben wir allerdings noch eine weitere Stellschraube: wir können die Genauigkeit der Quadratur beliebig erhöhen, wenn wir mehr Funktionsauswertungen erlauben, also mehr Rechenaufwand in Kauf nehmen. Tatsächlich können wir sogar eine bestimmte Toleranz vorgeben, mit der wir allerhöchstens mit unserer Schätzung von der echten Zustandsverteilung bis zum Zeithorizont abweichen wollen. Damit können wir zwischen benötigten Rechenressourcen und Konservativität unserer Stabilitätsgarantie abwägen.

Nun ist unser Regler produktreif! Wir haben mit Hilfe von künstlicher Intelligenz ein Modell eines komplexen Systems gelernt, das zusätzlich die Sicherheit der eigenen Schätzungen explizit abbildet. Durch Interaktionen mit dem physikalischen System konnten wir einen Regler lernen, der das System unseren Anforderungen entsprechend betreibt und dabei die Modellunsicherheit berücksichtigt. Schließlich konnten wir für einen Bereich im Zustandsraum die korrekte Funktionalität des Reglers nachweisen, ebenfalls unter Berücksichtigung der Modellunsicherheit. Der gesamte Prozess von Modell zu Stabilitätsgarantie erfordert dabei kein manuelles Eingreifen und läuft komplett automatisiert ab. Die Ergebnisse sind effizient und kostengünstig erzielt worden und dabei jederzeit nachvollziehbar und reproduzierbar. Ein unglaublicher Fortschritt! Marvin Minsky hat Recht behalten – die Details einzelner Anwendungen können wir Dank den Durchbrüchen in künstlicher Intelligenz dem Computer überlassen und uns stattdessen bei den Problemen der Zukunft den Zielen zuwenden.

Literaturverzeichnis

[1] J. Vinogradska, B. Bischoff, D. Nguyen-Tuong, A. Romer, H. Schmidt, J. Peters, “Stability of Controllers for Gaussian Process Forward Models”, International Conference on Machine Learning (ICML), 2016.

[2] J. Vinogradska, B. Bischoff, D. Nguyen-Tuong, J. Peters, “Stability of Controllers for Gaussian Process Dynamics”, (accepted), Journal of Machine Learning Research (JMLR), 2017.