technische universität
dortmund

Bachelorarbeit

Parameterschätzung mit Gütegarantie durch Bandit Models für die Regelung im Industrie 4.0 Kontext

Pierre Haritz

6. November 2017

Gutachter:
Prof. Dr. Katharina Morik
M.Sc. Sebastian Buschjäger

Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl für künstliche Intelligenz
http://www-ai.cs.tu-dortmund.de

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation und Hintergrund

Im Rahmen der Industrie 4.0 wächst der Bedarf an Automatisierung in allen Bereichen der Industrie. Vor allem dezentrale Entscheidungen, die von den eingesetzten Maschinen getroffen werden sollen, gewinnen immer mehr an Bedeutung.
Die Regelungstechnik befasst sich mit der Aufgabe, einen sich zeitlich verändernden Prozess von außen so zu beeinflussen, dass dieser Prozess in einer vorgegebenen Weise abläuft. Dabei wird insbesondere bei anspruchsvollen Regelungen auf qualifizierte Fachleute statt auf heuristische Verfahren zurückgegriffen.
Da dies vor allem bei komplexen Reglern zeitintensiv (also auch teuer) und in vereinzelten Fällen sogar unmöglich ist, ist der Wunsch nach Automatisierung groß. Gleichzeitig stellen Entscheidungen, die mit Unsicherheit über das Ergebnis getroffen werden sollen, ein großes Problem im maschinellen Lernen dar.
Der Begriff des Reinforcement Learning gliedert sich in die maschinellen Lernverfahren ein. Das Verfahren ist weder überwacht noch unüberwacht, da die korrekten Eingabe/Ausgabe-Paare dem Modell nicht zur Verfügung stehen und Fehler nicht explizit korrigiert werden, sondern aus Feedback gelernt wird. Der Fokus dabei ist auf der Balance zwischen *exploration*, also die Auswirkung von Aktionen, deren Ausgang unbekannt ist zu erforschen und der *exploitation*, dem Anwenden der aktuell besten bekannten Strategie.
Die *Bandit models*, als Teil des Reinforcement Learnings, haben ihren Ursprung im Begriff des “mehrarmigen Banditen". Sie stellen ein Problem aus der Wahrscheinlichkeitstheorie dar, in der ein Spieler in einer Spielhalle eine festgelegte Anzahl an einarmigen Banditen zur Verfügung hat und entscheiden muss, an welchen Maschinen er spielt, wie oft er das tut und in welcher Reihenfolge. Das Ziel des Spielers ist es das Maximum an Belohnungen zu erspielen, wobei die Belohnung jeder Maschine aus einer maschinenspezifischen Wahrscheinlichkeitsverteilung kommt. Dabei kann das Modell sowohl eine effiziente als auch sichere (Güteschranken werden eingehalten) *exploration* ermöglichen, um den Ertrag auf

lange Sicht zu optimieren. Auch auf *concept drifts* (Ändern der Systemeigenschaften im laufenden Prozess) kann das Modell reagieren.
Mit den Bandit Models eröffnet sich also eine vielversprechende Möglichkeit, um Kontrolltheorie und maschinelles Lernen miteinander zu verknüpfen.

1.2 Aufbau der Arbeit

Bestehende Ansätze und verwandte Arbeiten werden in Kapitel 2 vorgestellt. Kapitel 3 bietet einen Überblick über die Grundlagen der Regelungstechnik, während Kapitel 4 einen Einblick ins maschinelle Lernen gibt.
In Kapitel 5 wird dann ein Verfahren für das Lernen einer Regelstrecke und der anschließenden Regelung vorgestellt, welches im 6. Kapitel dann evaluiert wird. Fazit und anknüpfende Forschung sind Inhalt von Kapitel 7.

Kapitel 2

Verwandte Arbeiten

Immer noch ist die Verknüpfung von maschinellem Lernen und dem Regelung in der Robotik ein vielversprechendes Forschungsgebiet. Schon 1961 beschreibt Bellman [8] das *Two-armed Bandit Problem* im Zusammenhang mit *Adaptive Control Processes.*
Sutton et al. stellen in [28] Reinforcement Learning mit Direct Adaptive Optimal Control gleich. Dort beschreiben sie eine Klasse von Regelungsproblemen (*optimal control problems*), für die sie Reinforcement Learning als eine rechnerisch einfache und direkte Herangehensweise für adaptive optimale Regelung von nicht-linearen Systemen sehen.
Lewis und Vrabie beobachten das Auftreten von optimalem Verhalten in der Natur in [17] und ziehen daraus Schlüsse für das Design von Systemen, die optimales Verhalten lernen und ausüben sollen. In [18] gehen sie dann auf Regler für dynamische Systeme ein, die Eigenschaften von *adaptive control* und *optimal control* vereinen.
In [14] geben Kober et al. einen Überblick über Reinforcement Learning in der Robotik. Sie stellen in Aussicht, dass *Contextual Bandits* einen attraktiven Forschungsbereich für die Regelung von Robotern darstellen.
Ebenso wird das Lernen im Kontext der Regelungstechnik mit Neuronalen Netzen seit mehr als 35 Jahren erforscht[5],[7].

Kapitel 3

Regelungstechnik

In diesem Kapitel werden Grundlagen der Regelungstechnik und des maschinellen Lernens vorgestellt, die für das Verständnis für die folgenden Kapitel notwendig sind.

3.1 Regelungstechnik

Die Regelungstechnik, als Ingenieurwissenschaft, befasst sich mit der Aufgabe, einen sich zeitlich verändernden Prozess von außen so zu beeinflussen, dass dieser Prozess in einer vorgegebenen Weise abläuft.
Beginnend mit dem Drehzahlregler von J. Watt im Jahr 1788 über Optimierung von Reglern durch dynamische Programmierung durch R. Bellman im Jahre 1956 ist die Regelungstechnik im Zeitalter der Digitalisierung angekommen.[10]
Moderne Anwendungsgebiete der Regelungstechnik reichen von der Temperaturregelung von Raumluft über Geschwindigkeitsregelung von Fahrzeugen bis hin zu komplexen Aufgaben, wie der Kursregelung von Autopiloten in der Schiff-, Luft- und Raumfahrt, oder auch der Regelung von Industrie-Robotern.

3.1.1 Aufgabe

Zunächst definieren wir wichtige Grundbegriffe der Regelungstechnik:

3.1.1 Definition. System (Steuerungs- und Regelungstechnik)[10]
Ein System ist ein physikalisches oder auch technisches Gebilde, das das Eingangssignal in ein Ausgangssignal umformt.
Wenn sich Eingangs- und Ausgangsgrößen zeitlich ändern, werden sie oft als zeitliche Funktionen dargestellt. In diesem Falle bezeichnet man das System auch als *dynamisch*.

3.1.2 Definition. Regelungsaufgabe[19]
Gegeben ist ein dynamisches System (Regelstrecke) mit einer von außen beeinflussbaren Größe (Stellgröße) und einer messbaren Größe (Regelgröße). Weiterhin ist ein Regelungsziel vorgegeben, welches sich aus der konkreten Aufgabe ableitet. Gleichzeitig soll die Wirkung der äußeren Störung unterdrückt werden.
Gesucht ist eine Regeleinrichtung (kurz:Regler), die unter Nutzung der gemessenen Werte die Stellgröße so vorgibt, dass das geregelte System das Regelungsziel erfüllt.

3.1.2 Regelkreis

In der Regelungstechnik beschreibt man das Gebilde durch einen Regelkreis (Abbildung 3.1).

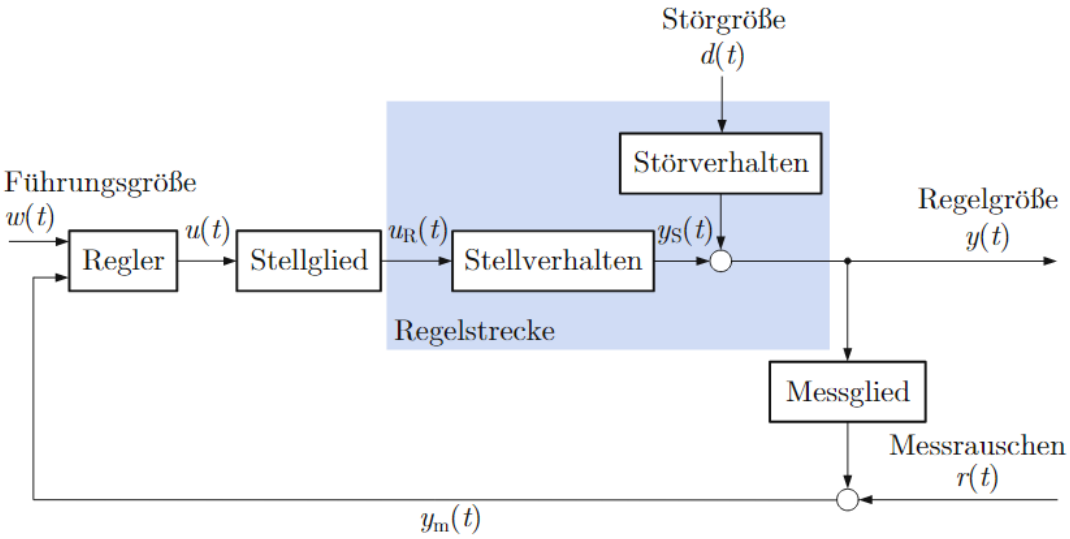

Abbildung 3.1: Erweiterter Regelkreis.[19]

w(t): Führungsgröße, auch Soll-Wert. Diesen Wert soll die Regelung erreichen.
Regler: wandelt den Fehler $e(t) = w(t) - y_m(t)$ in eine Stellgröße $u(t)$ um
u(t): Stellgröße. Aus den regelungsspezifischen Parametern berechnet, beeinflusst dieser Wert direkt das zu regelnde System (die Regelstrecke).
Regelstrecke: Bestehend aus dem Stellverhalten (beeinflusst durch den Regler) und Störverhalten (beeinflusst durch externe Störfaktoren, dargestellt durch Störgröße $d(t)$ ist die Regelstrecke das zu regelnde Glied im Regelkreis.
y(t): Regelgröße (Ausgabe), auch Ist-Wert. Dieser Wert ist das tatsächliche Wert, der durch den Regler erreicht wurde. Dieser Wert wird als Feedback $y_m(t)$, in der Realität durch Messrauschen $r(t)$ verfälscht, zurückgeführt und als Fehler $e(t) = w(t) - y_m(t)$ in

den Regler weitergeleitet. Ziel ist es den Fehler zu minimieren. Im folgenden wird für die Regelgröße die Notation $w_{ist}(t)$ verwendet.

3.1.3 Optimale Regelung

Die optimale Regelung ist ein Prinzip der Regelungstechnik, um eine optimale Ansteuerung für ein gegebenes System zu finden[20].
Um die Güte der Ansteuerung zu bewerten, wird bei der optimalen Regelung ein Gütemaß J herangezogen. Für einen Startwert w_0 und eine Steuerungsfunktion $u(t)$ definiert sich das Gütemaß im Zeiintervall $0 \leq t \leq t_e$ durch:

$$J_e(s_0, u(t)) = w_{ist}^T(t_e) S w_{ist}(t_e) + \int_{t_0}^{t_1} (w_{ist}^T(t) Q_{w_{ist}} w_{ist}(t) + u^T(t) R u(t)) dt \tag{3.1}$$

Dabei beschreibt der linke Teil der Gleichung die Geschwindigkeit, mit der die Regelstrecke von s_0 aus $s = 0$ erreicht und der rechte Teil den Aufwand für die Umsteuerung. Das Gütefunktional J wird insbesondere durch die Gewichtsmatrizen S, Q_y, R festgelegt.

Ziel ist es eine Steuerungsfunktion $u^*(t)$ zu finden, die J minimiert, also die Funktion $u(t)$, für die die Summe beider Teile minimal ist:

$$\min_{u(t)} J_e(s_0, u(t)) = J_e(s_0, u^*(t)) \tag{3.2}$$

3.1.4 Adaptive Regelung

Ein adaptiver Regler ist ein Regler mit veränderbaren Regelparametern und einem Mechanismus zum Einstellen der Parameter. Dabei sind adaptive Regler besonders als selbsteinstellende Regler geeignet, bei dem die Parameter geschätzt werden.[6]

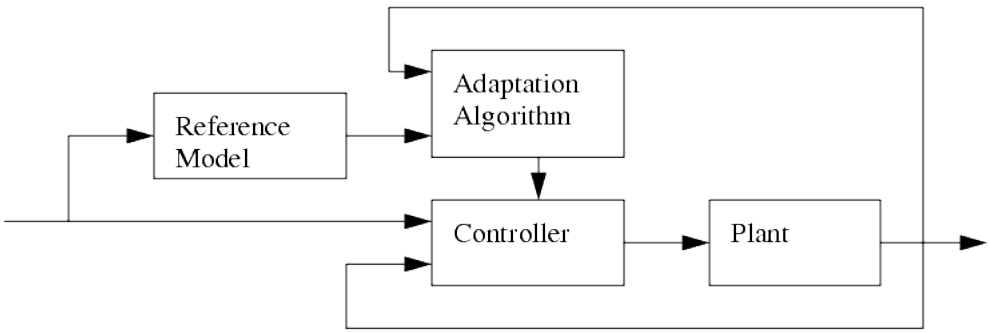

Abbildung 3.2: Geregelte Adaption mit Referenzmodell (MRAC).[1]

Abbildung 3.2 zeigt ein Prinzip der adaptiven Regelung. Hierbei wird der Fehler nun aus der Differenz zwischen der Führungsgröße des Reglers(*Controller*) und der Führungsgröße des Referenzmodells berechnet. Der Regler aktualisiert dann seine Parameter mit den von dem Korrekturmechanismus erhaltenen.

Kapitel 4

Maschinelles Lernen

Grundsätzlich beschäftigt sich das Gebiet des maschinellen Lernens damit, aus Daten zu lernen und aus dem Gelernten Vorhersagen über ähnliche Daten zu machen.
Die generische Aufgabe lässt sich folgendermaßen beschreiben[22]:
Eine Menge von Merkmalen $\mathcal{X}$, beschreibt eine Menge von Objekten $\vec{x} = (x_1, ..., x_n)$, wobei n die Dimension von X ist.
Ein Lernverfahren findet eine Funktion $f_\theta(x)$ mit $f_\theta : \mathcal{X} \mapsto \mathcal{Y}$, die den Objekten einen quantitativen oder qualitativen Ausgabewert zuordnet.
Als Modell bezeichnet man dann die gelernte Funktion.
Beim maschinellen Lernen wird zwischen überwachtem Lernen (supervised learning), unüberwachtem Lernen (unsupervised learning) und bestärkendem Lernen (reinforcement learning) unterschieden.

4.1 Supervised Learning

Als *supervised learning* bezeichnet man die maschinelle Lernaufgabe, die das Ziel hat, eine Funktion aus gelabelten Trainingsdaten abzuleiten.
Gegeben eine Menge $\mathcal{D} = \{(\vec{x_1}, y_1), \ldots, (\vec{x_n}, y_n)\}$ mit Merkmals-Vektor $\vec{x_i}$ und Label y_i für das i-te Beispiel,
finde eine Funktion $f_\theta : \mathcal{X} \mapsto \mathcal{Y}$ für $\vec{x}_1, \ldots, \vec{x}_n \in \mathcal{X}$ und $y_i, \ldots, y_n \in \mathcal{Y}$, sodass für ein neues $\vec{x}$ dann $y = \hat{f}_\theta(\vec{x})$ möglichst genau ist.
Dieses Verfahren nennt man auch Klassifikation.

Die Regression gliedert sich in die überwachten Lernverfahren ein.

4.1.1 Definition. Regression[12]
Gegeben unbekannte Parameter β, unabhängige Variablen $\mathbf{X}$ und abhängige Variable Y, finde eine Funktion für $Y \approx f(\mathbf{X}, \beta)$.

4.1.1 Logistic Regression

Die logistische Regression ist ein Regressionsmodell, in der die abhängige Variable kategorisch (binär) ist.
Dabei wird der Zusammenhang zwischen der kategorischen Variable und anderen unabhängigen Variablen durch Schätzung der Wahrscheinlichkeiten gemessen. Die dabei verwendete Funktion ist die *logistische Funktion* für $t \in \mathbb{R}$

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}} \tag{4.1}$$

Die Funktion eignet sich gut, um Wahrscheinlichkeiten auszudrücken, da $\sigma(t) \in (0, 1)$ gilt. Für den Fall, dass t eine lineare Funktion mit n abhängigen Variablen $x_1, \ldots, x_n = \vec{x}$ ist, ist die logistische Funktion mit $t = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n = \beta^T \vec{x}$ dann:

$$F(x) = \frac{1}{1 + e^{-\beta^T \vec{x}}} \tag{4.2}$$

[15]

4.1.2 Conditional Random Field

Conditional Random Fields (CRFs) zählen zu den statistischen Modellen und werden häufig im maschinellen Lernen für Probleme verwendet, bei denen ein Objekt vorhergesagt werden soll.

4.1.2 Definition. Conditional Random Field[16]
Seien $\mathcal{X}$ Beobachtungen und Y Zufallsvariablen. Sei $G = (V, E)$ ein Graph mit $Y = (Y_v)_{v \in V}$. Dann ist $(\mathcal{X}, Y)$ ein Conditional Random Field, falls gilt $p(Y_v | X, Y'_w, w \neq v) = p(Y_v | \mathcal{X}, Y_w, w \sim v) \forall v, w \in V$. Dabei bezeichne $v \sim w$ benachbarte Knoten $v, w \in V$.

4.2 Reinforcement Learning

Sutton und Barto definieren das Reinforcement Learning in [27] folgendermaßen:
Reinforcement Learning ist das Projizieren von Situationen auf Aktionen und gleichzeitiges Maximieren einer numerischen Belohnung.
Mathematisch betrachtet werden Reinforcement Learning-Settings oft als Markov-Entscheidungsproblem modelliert.

4.2.1 Definition. Markov-Entscheidungsproblem
Ein Markov-Entscheidungsproblem (auch *MDP*, aus engl. *Markov Decision Process*) ist ein Tupel (S, A, P, R, γ). Dabei ist S eine endliche Menge von Zuständen, A eine endliche Menge von Aktionen, $P_a(s, s')$ die Wahrscheinlichkeit, dass Aktion a in Zustand s zum

Zeitpunkt t zu Zustand s' zum Zeitpunkt $t+1$ führt, $R_a(s, s')$ die Belohnung, die nach mit dem Übergang von s zu s' durch Aktion a assoziiert ist und γ ein Gewicht für Belohnungen mit $\gamma \in (0, 1)$.

Das Ziel eines Markov-Entscheidungsproblems ist es eine Policy $\pi(s)$ zu finden, mit der Entscheidungen über Aktionen getroffen werden können. Dabei gilt es für $a_t = \pi(s_t)$ die Summe der möglichen Belohnungen

$$\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1}) \tag{4.3}$$

zu maximieren. Da $\gamma \in (0, 1)$ gilt, werden Aktionen also weniger gewichtet, je weiter sie in der Zukunft liegen.

4.2.2 Definition. Erwartungswert[25]
Gegeben sei eine Zufallsvariable X im Wahrscheinlichkeitsraum (Ω, Σ, P) mit Ergebnismenge Ω, Ereignismenge Σ und Wahrscheinlichkeitsmaß P. Dann definiert sich der Erwartungswert für X durch $\mathbb{E}[X] = \int_\Omega X(\omega) dP(\omega)$.

Dabei wird eine Werte-Funktion, die einen Zustand s bewertet, benötigt. Diese definiert sich durch:

$$V_\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_{a_t} \mid s_0 = s\right] \tag{4.4}$$

<mehr hier!>

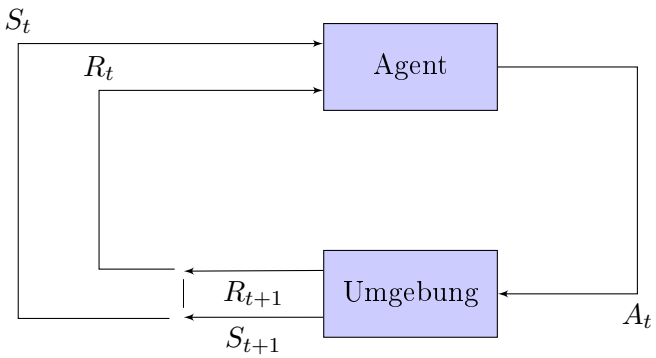

Abbildung 4.1: Feedback-Schleife im Reinforcement Learning. [27]

Abbildung 4.1 skizziert die Feedback-Schleife im Reinforcement Learning. Als Agent bezeichnet man den Akteur, also die Instanz, die eine Aktion auswählt. Das Ziel des Agenten ist es, die Summe an Belohnungen aus den Aktionen, die mit Hilfe einer internen Policy getroffen werden, zu maximieren.
Die Umgebung generiert nach Aktion A_t eine Belohnung (*reward*) R_{t+1} und gibt diese

zusammen mit dem neuen Zustand S_{t+1} an den Agenten zurück. Dieser nutzt dann das Feedback, um seine Policy anzupassen.
Dabei ist dieser neue Zustand nur von der Aktion abhängig und nicht von der Belohnung.

Im Reinforcement Learning wird dem lernenden Agenten also nicht gesagt, welche Aktionen er ausführen soll, wie oft üblich im maschinellen Lernen, sondern er muss selbst durch Ausprobieren lernen, was gut und was schlecht ist. Das bedeutet also, dass die Wahl der besten Aktion nicht immer die beste Belohnung auf lange Sicht gibt, sondern auch eine aktive Exploration der anderen Möglichkeiten notwendig ist, um Lernfortschritte zu machen.

Allgemein ist die Balance zwischen *exploration* und *exploitation* wichtiger Bestandteil des Reinforcement Learnings. Dabei wird die beste bekannte Strategie verwendet (*exploitation*), aber auch versucht, die Umgebung weiter nach einer Strategie mit besserer zukünftiger Belohnung zu durchsuchen (*exploration*). Dabei kommt es vor, dass durch Wählen einer zu Zeitpunkt t suboptimalen Aktion ein Zustand erreicht wird, aus dem eine Aktion mit besserer Belohnung gewählt werden kann, als die, die im vorherigen Zustand zum Zeitpunkt t optimal war.

4.3 Bandit Models

Das Treffen von Entscheidungen unter unsicheren Bedingungen ist eine Herausforderung im maschinellen Lernen.

Die Belohnungen, die von der Umgebung bei Ausführung einer Aktion zurückgegeben werden, sind jedoch nicht immer bekannt. Im Gegensatz zum Reinforcement Learning offenbart sich also die Belohnung erst nach Ausführen der Aktion. Dieses Problem bezeichnet man als mehrarmigen Banditen, oder auch n-armigen Banditen. Grundsätzlich kann man die *Bandit Models* also als eine Teilmenge der RL-Probleme auffassen.

Das Modell des n-armigen Banditen lässt sich als Markov-Entscheidungsproblem mit unbeschränktem Zeithorizont definieren[9].
Sei $t = 0, 1, \ldots$. Dann soll zu jedem t eine Auswahl einer Aktion aus n Verteilungen(Armen) $a_t \in 1, \ldots, n$ getroffen werden. Für $a_t = k$ ist R_t^k eine zufällige Belohnung und X_t^k die zugehörige Zufallsvariable. Für Zustandsvariable s_t kann die Verteilung von R_t^k auch durch $F^k(\cdot; st)$ ausgedrückt werden. Die Zustandsübergangsfunktion ϕ ist dabei sowohl von der Wahl des Armes abhängig als auch von der Belohnung:

$$s_{t+1} = \phi(R_t^k; s_t) \tag{4.5}$$

Dieses Problem ist dann ein *bandit problem*, wenn die Belohnungen mit dem Faktor $\gamma \in (0, 1)$ gewichtet werden und die Belohnung für jedes k nur von $a_t = k$ abhängig ist. In dem Fall gelte für $s_t = (s_t^1, \ldots, s_t^n)$ im Fall $a_t \neq k$:

$$s_{t+1}^k = s_t^k \tag{4.6}$$

und im Fall $a_t = k$:

$$s_{t+1}^k = \phi(s_t^k, R_t) \tag{4.7}$$

Die Komplexität besteht darin, dass zusätzlich zur Lernaufgabe des Reinforcement Learning also auch die erwarteten Belohnungen aller verfügbaren Aktionen geschätzt werden müssen. Deswegen eignen sich die Bandit Models aber besonders gut für Aufgaben, in denen Entscheidungen trotz unbekannter Folgen getroffen werden müssen.

Kapitel 5

Regelung durch Bandit Learner

Dieses Kapitel behandelt die Zusammenführung von maschinellem Lernen und der Regelung von Systemen.
Nach einer Einordnung der Fragestellung in Abschnitt 4.1, wird der Bandit-Learning-Ansatz *Counterfactual Risk Minimization* in Abschnitt 4.2 vorgestellt, indem die Funktionsweise erklärt und der Lernalgorithmus dargestellt wird.
Abschnitt 4.3 befasst sich mit den Vorraussetzungen, die für eine konkrete Anwendung erfüllt sein müssen und stellt ein Modell für die Anwendung im Kontext der Regelungstechnik vor.

5.1 Verknüpfung von Regelungstechnik und Bandit Learning

Für die Anwendung von maschinellem Lernen auf ein optimales Regelungsproblem eines sich dynamisch ändernden Systems, ist es generell schwierig ein überwachtes Lernverfahren zu benutzen, das dem Agenten genau sagt was er tun soll. Das Problem besteht vorwiegend darin, dass die optimale Regelung eventuell keine Informationen über veränderte Systemeigenschaften nach Auslegung in die Steuerung mit einbeziehen kann.
Wir wissen also nicht, was zum Zeitpunkt t die beste Aktion ist. Eine Lösung könnte ein Reinforcement Learning-Ansatz sein. Hierbei müssen nämlich keine Informationen über beste mögliche Lösung vorhanden sein, sondern der Agent erhält ein Feedback über die Auswirkung seiner gewählten Aktion. Das Ziel ist, dass der Agent so selber versucht, die auf lange Sicht beste Lösung zu finden.
Bisherige Arbeiten haben bereits die Schnittstelle zwischen Reinforcement Learning und der Regelung untersucht. So ist für Regelungssysteme mit Zustandsraumdarstellung ein Lernen mit MDPs möglich, dabei werden jedoch Systemeigenschaften wie Abnutzungserscheinungen, die nicht durch einen Zustand im Zustandsraummodell beschrieben werden, vernachlässigt.

Wir werden im Folgenden versuchen, eine Möglichkeit zu finden, durch Bandit Learning unabhängig von Zuständen ein Modell trotz Unsicherheiten über das System zu lernen und Vorhersagen für die Regelung in einem Regelkreis zu treffen.

5.2 POEM

Der Policy Optimizer for Exponential Models (POEM) [29] ist ein *batch learning*-Algorithmus, der aus geloggtem Bandit-Feedback durch Counterfactual Risk Minimization (CRM) lernt. Um den Lernalgorithmus herzuleiten, klären wir zunächst das Prinzip der Counterfactual Risk Minimization.

Man betrachte ein Problem, in dem aus Beobachtung $x \in \mathcal{X}$ eine Vorhersage $y \in \mathcal{Y}$ getroffen werden soll.
Eine Hypothese $h(x) \equiv h(\mathcal{Y}|x) \in \mathcal{H}$ im Hypothesenraum $\mathcal{H}$ beschreibe dabei eine Wahrscheinlichkeitsverteilung über $\mathcal{Y}$, oder konrekt gesagt, wie wahrscheinlich ein y für x ist. Die Vorhersage wird dabei durch Ziehen einer Zufallsstichprobe $y \sim h(\mathcal{Y}|x)$, oder alternativ durch $y = \arg\max_{y \in \mathcal{Y}} h(y \mid x)$, für x getroffen.

Im Rahmen des Bandit-Learning kann nur das Feedback von den Daten beobachtet werden, von denen eine Stichprobe gezogen wird. Dabei soll die Belohnung auf lange Sicht maximiert werden. Da wir anstatt der maximalen Belohnung den minimalen risk betrachten wollen, muss Feedback-Funktion $\delta : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$ minimiert werden. Dies beinhaltet eine Kombination aus dem Wählen einer Aktion mit aktuell höchster Belohnung und dem Ausprobieren neuer Aktionen. Da für unerkundete Eingabe-Ausgabe-Paare nicht einfach zufällige Aktionen gewählt werden sollen, muss die Belohnung geschätzt werden. Für eine maximale Belohnung muss der erwartete Loss(hier: *risk*) $R(h) = \mathbb{E}_{x \sim Pr(\mathcal{X})} \mathbb{E}_{y \sim h(x)}[\delta(x, y)]$ minimal sein. Das bedeutet also, die Belohnung ist größer, je geringer der risk ist.
Um dies zu erreichen muss eine Hypothese $h \in \mathcal{H}$ mit minimalem risk gefunden werden.

Für den Policy Optimizer nehmen wir an, dass das System, das wir betrachten, bereits Vorhersagen über $\mathcal{X}$ nach einer Policy (Verteilung) $h_0(x)$ treffen kann und Log-Daten $\mathcal{D}_0 = \{(x_1, y_1, \delta_1), \ldots, (x_n, y_n, \delta_n)\}$ dieses historischen Systems, mit $y_i \sim h_0(x_i)$ und $\delta_i \equiv \delta(x_i, y_i)$, vorhanden sind.
Da D_0 sowohl biased als auch unvollständig ist, kann $R(h)$ für neue Hypothese h nicht geschätzt werden und muss durch eine Korrektur der Stichprobenverzerrung ausgeglichen werden.
Dies erreichen wir durch *Importance Sampling*:

5.2.1 Definition. Importance Sampling
Gegeben Zufallsvariablen $X_1, \ldots, X_n$ aus einer Dichte $g(x)$ gezogen, schätze den Erwartungswert für $H = \mathbb{E}_f[h(X)] = \int h(x)f(x)dx$ und $w_i = w(X_i) = \frac{f(X_i)}{g(X_i)}$ durch
$\hat{\mathbb{E}}_n[X;P] = \frac{1}{n}\sum_{i=1}^{n} h(X_i)w_i$.

Durch das Importance Sampling können wir nun den Unterschied der Verteilungen h_0 und beliebige h in die Schätzung mit einbeziehen und es ergibt sich der erwartete risk $R(h) = \mathbb{E}_{x\sim Pr(\mathcal{X})}\mathbb{E}_{y\sim h_0(x)}[\delta(x,y)\frac{h(y|x)}{h_0(y|x)}]$.

Da wir $h_0(x)$ für diesen Schritt benötigen, fügen wir die Propensität $p_i \equiv h_0(y_i|x_i)$ zum bestehenden Datensatz hinzu und erhalten so $\mathcal{D} = \{(x_1, y_1, \delta_1, p_i), \ldots, (x_n, y_n, \delta_n, p_n)\}$.

5.2.2 Definition. Monte-Carlo-Simulation[11]
Gegeben ein Wahrscheinlichkeitsraum Ω und eine Größe $\mathcal{A}$, wobei $\mathcal{A}(x)$ den Wert von $\mathcal{A}$ in Zustand x ist, erzeuge eine Markov-Kette $x_1, x_2, \ldots$ von Zuständen in Ω, die die Verteilung von Gewichten $P(x)$ darstellen und schätze den Erwartungswert von $\mathcal{A}$ durch $\mathbb{E}[\mathcal{A}] = \frac{1}{N}\sum_{i=1}^{N}\mathcal{A}(x_i)$.

Für $\mathcal{D}$ kann nun eine Schätzung ohne Bias durch eine Monte-Carlo-Simulation abgegeben werden und wir erhalten

$$\hat{R}(h) = \frac{1}{n}\sum_{i=1}^{n}\delta_i\frac{h(y_i|x_i)}{p_i}. \tag{5.1}$$

Zur Reduktion der Varianz werden die Gewichte des Importance Samplings verringert [13] und wir erhalten $\hat{R}^M(h) = \frac{1}{n}\sum_{i=1}^{n}\delta_i \min\{M, \frac{h(y_i|x_i)}{p_i}\}$ und damit die Gleichung für das *Inverse Propensity Scoring*-Trainingsziel.

$$\hat{h}^{IPS} = \arg\min_{h\in\mathcal{H}}\{\hat{R}^M(h)\}. \tag{5.2}$$

mit Hyperparameter $M > 0$.

Für das Inverse Propensity Scoring wird angenommen, dass die Klassen $\vec{y_i} \in \{0,1\}^q$ mit je q Labels eines vorhandenen Datensatzes nicht zufällig zugeordnet wurden. Die kontrafaktischen Auswirkungen werden dann unter Betracht der tatsächlichen Klasse $\vec{y}$ und $\neg\vec{y}$ geschätzt.

Der Übersichtlichkeit halber, definieren wir wie folgt $u_h^i \equiv \delta_i \min\{M, \frac{h(y_i|x_i)}{p_i}\}$ und $\bar{u_h} \equiv$

$\frac{1}{n}\sum_{i=1}^{n} u_h^i$. Desweiteren gelte $\mathbf{Var}_h(u) \equiv \frac{1}{n-1}\sum_{i=1}^{n}(u_h^i - \bar{u_h})^2$.
Die Autoren [29] geben darüber hinaus eine obere Schranke für den tatsächlichen risk:

$$\forall h \in \mathcal{H} : R(h) \leq \hat{R}^M(h) + \mathcal{O}\left(\sqrt{\frac{Var_h(u)}{n}}\right) \tag{5.3}$$

Motiviert durch die Schranke, nennen sie das Prinzip *Counterfactual Risk Minimization*. Dabei sollen gleichzeitig die Schätzung $\hat{R}^M(h)$ und die empirische Standardabweichung optimiert werden.
Es folgt die Definition des CRM-Trainingsziels:

$$\hat{h}^{CRM} = \arg\min_{h\in\mathcal{H}}\{\hat{R}^M(h) + \lambda\sqrt{\frac{\mathbf{Var}_h(u)}{n}}\}. \tag{5.4}$$

5.2.1 Lernalgorithmus

Allgemein formuliert, soll POEM nach einer Hypothese h_w mit geringstem risk suchen. Dabei wird durch stochastischen Gradientenabstieg (SGD) für das CRM-Trainingsziel ein Optimum gesucht.
Die Gewichte w werden dabei im Laufe des Lernverfahrens nach

$$w^* = \arg\min_{w\in\mathbb{R}^d} \bar{u_w} + \lambda\sqrt{\frac{\mathbf{Var}_w(u)}{n}} \tag{5.5}$$

optimiert.

Im allgemeinen Fall [23] für Label $k \in 1, \ldots, q$ zieht $h_w \in \mathcal{H}_{lin}$ die Stichprobe mit

$$h_w(y_k|x) = w_k \bullet x \tag{5.6}$$

Wir betrachten im folgenden die Logging- und Optimierungsalgorithmen:

procedure GENERIERELOGDATEN(pred,X, Y)
  $Y' \leftarrow \{\}$
  for $i = 1, \ldots, q$ **do**
    $P_i \leftarrow pred.PredictLogProbability(X)$
  end for
  for $j = 1, \ldots, n$ **do**
    $Y'_j \sim h(X_j)$
    $\delta_j \leftarrow \text{Loss}(X_j, Y'_j)$
  end for
  for $k = 1, \ldots, q$ **do**
    if $Y'_k = 1$ **then**
      $p_k \leftarrow p_k + P_k$
    end if
  end for
  return (X, Y', p, δ)
end procedure

Algorithmus 5.1: Logging-Algorithmus

Für alle w_0 gelte[29]:

$$\sqrt{Var_w(u)} \leq A_{w_0} \sum_{i=1}^{n} u_w^i + B_{w_0} \sum_{i=1}^{n} \{u_w^i\}^2 + C_{w_0} \tag{5.7}$$

POEM optimiert die Gewichte durch:

procedure TRAINBANDIT(X, Y, δ, p)
  Kalibriere Hyperparameter M
  $b \leftarrow$ Batchgröße, $N \leftarrow$ Anzahl Beispiele aus X
  for $i = 1, \ldots, N/b$ **do**
    repeat
      $w \leftarrow w - \eta \nabla u_w^i + \lambda \sqrt{n} (A_{w_t} \nabla u_w^i + 2 B_{w_t} u_w^i \nabla u_w^i)$
    until convergence
  end for
end procedure

Algorithmus 5.2: Optimieren der Gewichte

5.3 Anwendung

Im folgenden Unterkapitel setzen wir uns mit der Zusammenführung beider Teilgebiete auseinander. Dazu klären wir zunächst die Vorraussetzungen, die für ein Bandit Learning mit POEM gegeben sein müssen und entwickeln im Anschluss Ansätze für eine Umsetzung.

5.3.1 Vorraussetzungen

Um den Policy Optimizer auf das Regelungsproblem anzuwenden, müssen bestimmte Kriterien erfüllt sein.
Wie bereits beschrieben, verbessert POEM eine bereits vorhandene Policy h_0. Diese muss vor also implementiert sein, bevor der POEM-Ansatz angewendet werden kann.
Zudem müssen die Ein- und Ausgabemengen $\mathcal{X}$ und $\mathcal{Y}$ festgelegt werden.
Letztlich muss eine geeignete Feedback-Funktion $\delta(x, y)$ für $x \in \mathcal{X}, y \in \mathcal{Y}$ gefunden werden, die die Qualität der Eingabe/Ausgabe-Paare bewertet.

Für die Merkmale gibt es keine besonderen Anforderungen. $\mathcal{X} := \{\vec{x_1}, \ldots, \vec{x_n}\}$ mit $\vec{x_i} = (x_{i_1}, \ldots, x_{i_k})$ sind im Falle der Regelungen einer Maschine alle Variablen des Systems, die beobachtet werden können, wie z.B. die Führungsgröße oder Sensorinformationen über die Maschine und helfen, das Verhalten, die Regelstrecke, zu lernen.

Wir betrachten den Fall, in dem die Ausgabe ein Label-Vektor $\vec{y} = (y_1, \ldots, y_q)^T$ ist mit $\vec{y} \in \mathcal{Y} = \{0, 1\}^q$. Dabei steht q für die Anzahl der Klassen. Da in unserem Fall nur eine Klasse pro Label gewählt werden kann, gilt $y_{i_j} = 1$ für $y_{i_j} \in (y_{i_1}, \ldots, y_{i_q})$, wenn j die Klasse von $\vec{x_i}$ ist, 0 sonst.

Die Wahrscheinlichkeitsverteilung $h_0(\mathcal{Y}|x)$ weist einer Eingabe x ein Label $y \in \mathcal{Y}$ zu. Falls keine solche Policy für $\mathcal{X}, \mathcal{Y}$ existiert, kann eine diskrete Gleichverteilung $h_0(y|x) := \frac{1}{q}, \forall x \in \mathcal{X}$ gewählt werden.

Um ein Lernen zu ermöglichen, muss ebenfalls eine Feedback-Funktion (oder Loss-Funktion) $\delta : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$ vorhanden sein, die der gewählten Aktion eine Belohnung zuordnet. Für POEM muss diese jedoch zwingend auf $[-1, 0]$ abbilden, da sonst beim Bandit learning mit POEM durch $\hat{R}(h)$ und $\hat{R}^M(h)$ keine zuverlässigen Schätzungen erreicht werden können.

5.3.2 Regelung mit POEM

Bei der Auslegung eines Reglers werden nur die aktuellen Systemeigenschaften betrachtet. Für ein sich selbst regelndes System ist es ein Problem nach der Auslegung auf veränderte Gegebenheiten zu reagieren. So werden beispielweise Parameter für die Auslegung verwendet, die sich im Laufe der Zeit ändern und die Funktionsweise beeinflussen, was dazu führen kann, dass der Regler ohne Neuauslegung im Laufe der Zeit an Optimalität verliert. Im Realfall, kann das das Gewicht einzelner Maschinenkomponenten sein, die im Laufe der Zeit abgenutzt werden, veränderte Reibung, etc.

Agent als Regler-Ersatz:
Eine Möglichkeit, die untersucht werden soll, ist das Trainieren eines Agenten als Regler-Ersatz, sodass er in der Lage ist das System durch Vorhersage der Stellgröße zu regeln.

Für POEM müssen die Logdaten in der Form $\mathcal{D} = \{(x_1, y_1, \delta_1, p_i), \ldots, (x_n, y_n, \delta_n, p_n)\}$ für n Beispiele vorhanden sein. n kann dabei beispielsweise die Anzahl von Zeitschritten sein, an denen Daten aufgezeichnet worden sind.
Für einen Regelkreis sind diese Daten in der Form jedoch nicht vorhanden. Wir nehmen im Folgenden an, dass wir messbare Daten aus dem betrachteten Regelkreis erhalten können. Daraus soll dann am Ende der Datensatz $\mathcal{D}$ konstruiert werden.

Wir wählen $\mathcal{X} := (\vec{x_1}, \ldots, \vec{x_n})$, wobei $\vec{x_t}$ mit $t \in \{1, \ldots, n\}$ die verfügbaren Messwerte aus dem Regelkreis zu Zeitpunkt t sind, die vor der Regelung vorhanden sind.
Desweiteren sei $\mathcal{Y} = \{0, 1\}^q$ eine diskretisierte Darstellung für die Domäne der Regelgröße $u(t)$, also der Größe, die wir später vorhersagen wollen.
Um das Label y_i für ein $\vec{x_i}$ bestimmen zu können, trainieren wir ein Conditional Random Field mit einem logistischem Regressor pro Label. Daraus wird dann das Label mit höchster Wahrscheinlichkeit y_{i_j} durch $y_{i_j} = 1$ für $y_{i_j} \in (y_{i_1}, \ldots, y_{i_q})$ bestimmt.
Für die Feedback-Funktion δ bietet es sich an, den Fehler, bei dem es unser Ziel ist, ihn zu minimieren, zu verwenden. Je nach Anwendung muss die Funktion so skaliert werden, dass für einen maximal großen Fehler gilt: $\delta = -1$ und respektive $\delta = 0$ für den kleinstmöglichen Fehler.
Letztlich verwenden wir für die Propensitäten $p_i \equiv h_0(y_i|x_i)$ die Wahrscheinlichkeiten, mit denen die y_i ausgewählt wurden.

Nach dem Training soll der Agent dann Vorhersagen über die Stellgröße treffen können. Abbildung 5.1 zeigt eine Skizze des Prinzips.

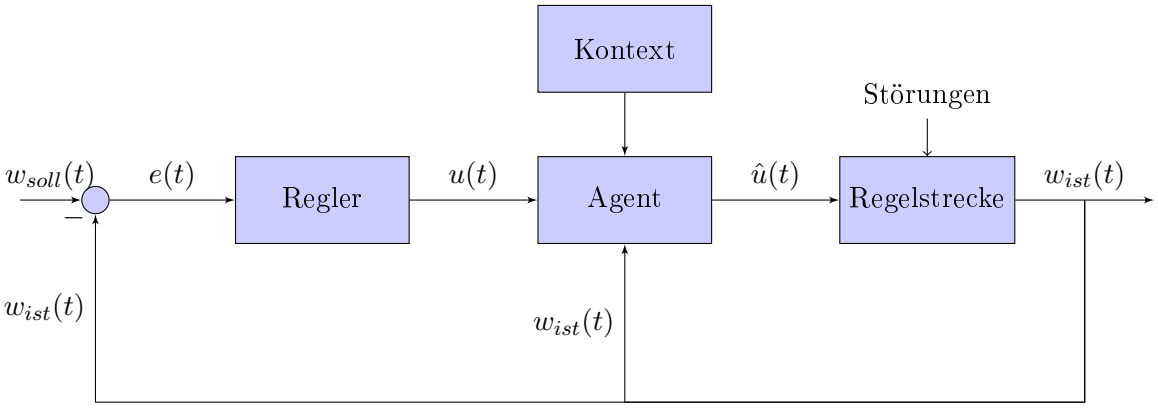

Abbildung 5.1: Regelkreis mit Bandit Learner

Hier hat der Regler keine direkte Kontrolle über das System, sondern der Agent gibt die Stellgröße $\hat{u}(t)$ an die Regelstrecke weiter. Dabei soll er bestenfalls Kontextinformationen über das System erhalten, die dem Regler möglicherweise nicht zur Vergügung stehen (oder die er nicht verarbeiten kann), um eine Vorhersage nach gelerntem Modell zu treffen.
Das Ziel ist es, den Regler nicht neu auslegen zu müssen, sondern die Regelstrecke als Funktion zu lernen. Eine Evaluierung dieses Ansatzes findet in 6 statt.

Agent als Korrekturmechanismus: Ein Prinzip der adaptiven Regelung ist es, durch einen Korrekturmechanismus direkt auf den Regler einzuwirken, indem Regelparameter geschätzt werden. Man bezeichnet ein solches Regelungsprinzip als *Model Identification Adaptive Control*, kurz: MIAC.

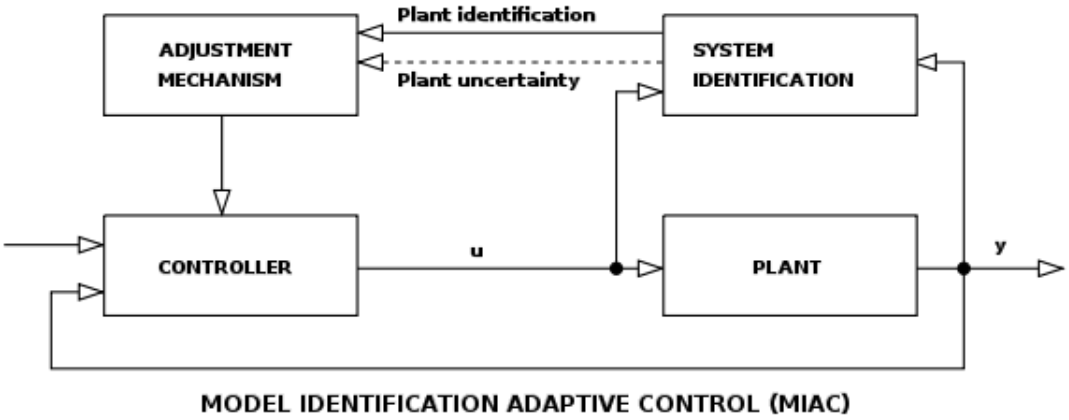

Abbildung 5.2: Geregelte Adaption ohne Vergleichsmodell (MIAC), Quelle: Wikipedia

In diesem Modell würde die Regelstrecke ebenfalls gelernt werden, jedoch nicht die Ausgabe des Reglers, sondern die Regelparameter vorhergesagt werden.
Dafür würden wir annehmen, dass nach Auslegung des Reglers eine optimale Regelungsma-

trix K existiert und der betrachtete Regler die Regelstrecke nach einer Funktion $u(x) = Kx$ regelt.
Der Agent soll dann ein A'_K bzw. eine Regelungsmatrix K' mit maximaler Belohnung wählen unter Betracht der veränderten Bedingungen.

Die Problematik dabei besteht vor Allem darin, Matrizen K bzw. K' mit reellwertigen Einträgen zu schätzen. Der in dieser Arbeit untersuchte Ansatz mit logistischer Regression würde sich dafür nicht eignen.

Kapitel 6

Experimente

Im folgenden Kapitel soll die Regelung durch einen mit CRM trainiertem Agenten evaluiert werden.
Da im Rahmen der Arbeit keine realen Maschinen zur Verfügung standen, wird dies im Folgenden anhand einer Simulation getestet. Die Simulationssumgebung ist in python entwickelt worden, um die Schnittstelle zur bestehenden POEM-Implementierung von Swaminathan in python zu erleichtern.
Die Simulation der Regelung baut auf [26] auf und stützt sich auf die python control library [4].

6.1 Inverses Pendel

Das inverse Pendel ist eine der Standardaufgaben der Regelungstechnik für die Stabilisierung einer instabilen Regelstrecke. Für eine Evaluation der Anwendung von Bandit learning auf die Regelung, stellt dieses Experiment also einen guten Startpunkt dar.

6.1.1 Grundlagen

Das inverse Pendel ist ein Pendel mit dem Schwerpunkt oberhalb der Achse. Das Pendel befindet sich in seinem höchsten Punkt in einer instabilen Ruhelage.

Ein Standardbeispiel für ein inverses Pendel ist ein Wagen mit einem darauf montierten Pendelstab. Die Pendelbewegung kann dabei nur durch die horizontale Bewegung des Wagens beeinflusst werden.

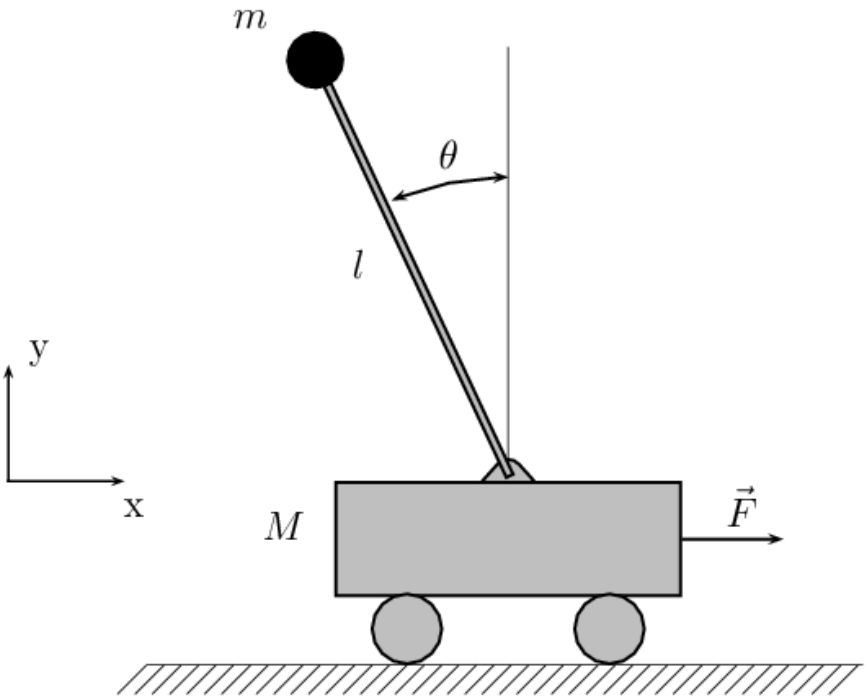

Abbildung 6.1: Inverses Pendel auf Kart, Quelle: Wikipedia

Abbildung 6.1 stellt ein inverses Pendel auf einem Kart dar.

Dabei sind die Variablen wie folgt definiert:
m: Masse des Pendels
M: Masse des Karts
F: Kraft in x-Richtung
l: Länge des Pendelstabs
θ: Auslenkung des Pendels. In der Regelungstechnik ist es üblich θ zu verwenden, um einen Winkel zu beschreiben. Um Verwirrung mit dem im maschinellen Lernen verwendeten θ für die Parametrisierung zu vermeiden, verwenden wir im Folgenden ϑ.
g: Gravitationskonstante mit $g = 9,81 \frac{m}{s^2}$

6.1.2 Bewegungsgleichungen für das Inverse Pendel

Für die Simulation des Systems müssen die physikalischen Gegebenheiten berechnet werden.[11] Wir teilen zunächst die Geschwindigkeit v_p des Pendels in y- und x-Komponente auf und erhalten

$$v_p^2 = v_x^2 + v_y^2 \tag{6.1}$$

Ebenso benötigen wir die Energiegleichungen für die potentielle und die kinetische Energie des Systems. Wir erhalten:

$$E_{pot} = mgl\cos\vartheta \tag{6.2}$$

und

$$E_{kin} = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}v_p^2 = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m(v_x^2 + v_y^2) = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m(\dot{x} - l\dot{\vartheta}cos\vartheta)^2 + \frac{1}{2}(-l\dot{\vartheta}sin\vartheta)^2$$

Durch Umformen ergibt sich:

$$E_{kin} = \frac{1}{2}(M+m)\dot{x}^2 + \frac{1}{2}m(-2\dot{x}l\dot{\vartheta}\cos\vartheta + l^2\dot{\vartheta}^2(\cos^2\vartheta + \sin^2\vartheta)) \tag{6.3}$$

Durch den Lagrange-Formalismus lässt sich die Dynamik eines Systems durch eine skalare Funktion (Lagrange-Funktion)

$$L = T - V = E_{kin} - E_{pot} \tag{6.4}$$

beschreiben.

Einsetzen von (6.2) und (6.3) in (6.4) ergibt:

$$L = \frac{1}{2}(M+m)\dot{x}^2 + \frac{1}{2}m(-2\dot{x}l\dot{\vartheta}\cos\vartheta + l^2\dot{\vartheta}^2(\cos^2\vartheta + \sin^2\vartheta)) - mgl\cos\vartheta \tag{6.5}$$

Für generalisierte Koordinaten $q_i = (x, \vartheta)$ und $\dot{q}_i$ lautet die Euler-Lagrange-Gleichung

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = Q_i \tag{6.6}$$

Dabei gilt für unser konkretes Beispiel $Q_i = 0$.
Wir betrachten zuerst den Fall $q_i = x$ und erhalten:

$$\frac{\partial L}{\partial \dot{x}} = (M+m)\dot{x} - ml\dot{\vartheta}\cos\vartheta \tag{6.7}$$

und

$$\frac{\partial L}{\partial x} = 0 \tag{6.8}$$

Somit folgt

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{x}} = (M+m)\ddot{x} - ml\ddot{\vartheta}\cos\vartheta + ml\dot{\vartheta}^2\sin\vartheta \tag{6.9}$$

und wir erhalten schließlich durch Einsetzen von (6.8) und (6.9) in (6.6) die erste Bewegungsgleichung:

$$(M+m)\ddot{x} - ml\ddot{\vartheta}\cos\vartheta + ml\dot{\vartheta}^2\sin\vartheta - 0 = 0 \tag{6.10}$$

Im Fall $q_i = \vartheta$ gilt:

$$\frac{\partial L}{\partial \dot{\vartheta}} = m(l^2\vartheta - l\dot{x}\cos\vartheta) \tag{6.11}$$

und wir erhalten mit

$$\frac{\partial L}{\partial \vartheta} = ml\dot{\vartheta}\cdot\dot{x}\sin\vartheta + mgl\sin\vartheta \tag{6.12}$$

dann die zeitliche Ableitung

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\vartheta}} = m(l^2\ddot{\vartheta} - l\ddot{x}\cos\vartheta + l\dot{x}\dot{\vartheta}\sin\vartheta) \tag{6.13}$$

Wir setzen (6.13) und (6.12) in (6.4) ein und es folgt die zweite Bewegungsgleichung:

$$m(l^2\ddot{\vartheta} - l\ddot{x}\cos\vartheta + l\dot{x}\dot{\vartheta}\sin\vartheta) - ml\dot{\vartheta}\cdot\dot{x}\sin\vartheta + mgl\sin\vartheta = 0 \tag{6.14}$$

Umformen ergibt:

$$ml^2\ddot{\vartheta} - ml\ddot{x}\cos\vartheta + m\dot{x}l\dot{\vartheta}\sin\vartheta - (ml\dot{\vartheta}\dot{x}\sin\vartheta + mgl\sin\vartheta) = 0$$

$$ml\ddot{x}\cos\vartheta - ml^2\ddot{\vartheta} = -mlg\sin\vartheta \tag{6.15}$$

Um den Zustand des Systems über die Zeit zu beschreiben, benötigen wir Gleichungen für $x(t), \dot{x}(t), \vartheta(t)$ und $\ddot{\vartheta}$.
Wir formen zunächst (5.10) nach $\ddot{x}$ um:

$$\ddot{x} = \frac{m}{M+m}l(\ddot{\vartheta}\cos\vartheta - \dot{\vartheta}\sin\vartheta) \tag{6.16}$$

und (5.14) nach $\ddot{\vartheta}$:

$$\ddot{\vartheta} = \frac{1}{l}(\ddot{x}\cos\vartheta + g\sin\vartheta) \tag{6.17}$$

Um eine GDGL 2. Ordnung herzuleiten, müssen wir die Ableitungen zweiter Ordnung jeweils auf der rechten Seite der Gleichungen eliminieren. Wir setzen also (6.17) in (6.16) ein. Es folgen:

$$\ddot{x} = \frac{(M+m)(g\sin\vartheta\cos\vartheta - \dot{\vartheta}^2\sin\vartheta)}{1-(M+m)\cos^2\vartheta} \tag{6.18}$$

und damit dann

$$\ddot{\vartheta} = \frac{\cos\vartheta\sin\vartheta(M+m)(g\cos\vartheta - \dot{\vartheta}^2)}{l-(M+m)l\cos^2\vartheta} + \frac{g}{l}\sin\vartheta \tag{6.19}$$

Wir erhalten letztendlich ein Gleichungssystem:

$$\dot{x}_{t+1} = \dot{x}_t$$
$$\ddot{x}_{t+1} = (5.16)$$
$$\dot{\vartheta}_{t+1} = \dot{\vartheta}_t$$
$$\ddot{\vartheta}_{t+1} = (5.18)$$

Zustandsraumdarstellung:
Für die Beschreibung eines Systems wird in der Regelungstechnik im Allgemeinen Zustandsraummodelle an Stelle von Diffenrentialgleichungen 1. Ordnung bevorzugt, da sich diese unter Anderem leichter interpretieren und in Simulationen verarbeiten lassen[19].

Für die Simulation verwenden wir daher eine Zustandsraumdarstellung, die das System beschreibt.

Eine allgemeine Form der Zustandsraumdarstellung für lineare Systeme [19] lautet:

$$\dot{\mathbf{x}}^T = A \cdot \mathbf{x}^T + B \cdot \mathbf{u} \tag{6.20}$$

Dabei ist $\mathbf{x} = (x, \dot{x}, \vartheta, \dot{\vartheta})$ der Zustands- und $\mathbf{u}$ der Regelungsvektor.

Zunächst müssen die Gleichungen linearisiert werden. Für kleine Abweichungen α von der Ruhelage ($\vartheta = 0$) approximieren wir[2]

$$\cos\vartheta = cos(0 + \alpha) \approx 1 \tag{6.21}$$
$$\sin\vartheta = sin(0 + \alpha) \approx 0 \tag{6.22}$$
$$\dot{\vartheta}^2 = \dot{\alpha}^2 \approx 0 \tag{6.23}$$

Mit diesen Annahmen erhalten wir nun die linearisierten Bewegungsgleichungen:

$$(M + m)\ddot{x} - ml\ddot{\alpha} = 0 \tag{6.24}$$
$$ml\ddot{x} - ml^2\ddot{\alpha} = 0 \tag{6.25}$$

Für $a = ((M + m) + Mml^2)^{-1}$ [3] folgt

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\vartheta} \\ \ddot{\vartheta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -ml^2a & m^2gl^2 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -mla & (M+m)mgla & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \vartheta \\ \dot{\vartheta} \end{bmatrix} + \begin{bmatrix} 0 \\ ml^2a \\ 0 \\ mla \end{bmatrix} u$$

6.1.3 LQ-Regler

Der Linear-Quadratische Regler, auch LQ-Regler gennant, regelt ein lineares dynamisches basierend auf einem Zustandsraummodell. Er findet Anwendung in der optimalen Regelung.

6.1.1 Definition. LQ-Regelung[20]
Gegeben ein Zustandsraumsmodell der Regelstrecke Σ_S und ein Gütefunktional J.
Gesucht eine Zustandsrückführung K^*.

Dabei gilt:

$$K^* = arg\min_K J(x_0, -Kx(t)) \tag{6.26}$$

Sei $\dot{\mathbf{x}}^T = A \cdot \mathbf{x}^T + B \cdot \mathbf{u}$ unser lineares System des inversen Pendels. Wir suchen ein Regelungsgesetz, sodass

$$J = \frac{1}{2}\sum_{t=0}^{\infty} x^T(t)Qx(t) + u^T(t)R(t)u(t) \tag{6.27}$$

minimiert wird. Mit der Riccati-Gleichung

$$A^T X + XA - XBR^{-1}B^T X + Q = 0 \tag{6.28}$$

kann dann die Parametersmatrix K des LQ-Reglers berechnen werden durch $K = R^{-1}B^T P$, wobei P die Lösung der Riccati-Gleichung ist[21].

Wir werden im folgenden Teil die Experimente auf einem LQ-geregelten System durchführen.

6.1.4 Finden einer linearen Feedback-Funkion

Der Auslenkungswinkel des Pendels wird durch $\vartheta \in [-2\pi, 2\pi]$ dargestellt. Beim inversen Pendel lässt sich der Soll-Wert leicht auf $\vartheta = 0$ festlegen, also dem Zustand, in dem sich das Pendel oberhalb des Karts befindet und senkrecht auf diesem steht. Für den Zustand, in der sich das Pendel unterhalb des Karts in der Ruhelage befindet, gelte $\vartheta = \pi$ bzw. $\vartheta = -\pi$.

Für eine geeignete Feedback-Funktion δ betrachten wir den Winkel des Pendels ϑ_{t+1} zu einem Zeitpunkt t, also der Winkel des Pendels im neuen Systemzustand nach Regelungsschritt t. Da nur Werte im Intervall $[-1, 0]$ als Feedback akzeptiert werden können, wird eine geeignete Skalierungsfunktion $\phi : [-2\pi, 2\pi] \to [-1, 0]$ benötigt. Für $\vartheta \in [-2\pi, 2\pi]$ definieren wir

$$\phi(\vartheta) = \begin{cases} -\frac{\vartheta \pmod{2\pi}}{\pi}, & \text{falls } \vartheta \pmod{2\pi} \leq \pi \\ \frac{\vartheta \pmod{2\pi}}{\pi} - 2, & \text{sonst} \end{cases} \tag{6.29}$$

für $\vartheta' = \vartheta \pmod{2\pi}$:

$$\phi(\vartheta) = \begin{cases} -\frac{\vartheta'}{\pi}, & \text{falls } \vartheta' \leq \pi \\ \frac{\vartheta'}{\pi} - 2, & \text{sonst} \end{cases} \tag{6.30}$$

Es ergibt sich daher für $\vartheta_t \in (x_t, \dot{x}_t, \vartheta_t, \dot{\vartheta}_t) = \vec{x}_t$ zu Zeitpunkt t:

$$\delta(\vec{x}_t, y_t) = \phi(\vartheta_{t+1}) \tag{6.31}$$

6.2 Aufbau der Simulation

Für die Simulation des inversen Pendels mit LQ-Regler zeichnen wir zunächst das Verhalten des Reglers über einen Zeitraum von t Sekunden auf. Die Daten werden zu jedem Zeitschritt Δt erfasst. Das Kart muss dabei alle k Sekunden an eine andere Position auf der Schiene fahren und sich möglichst schnell stabilisieren.

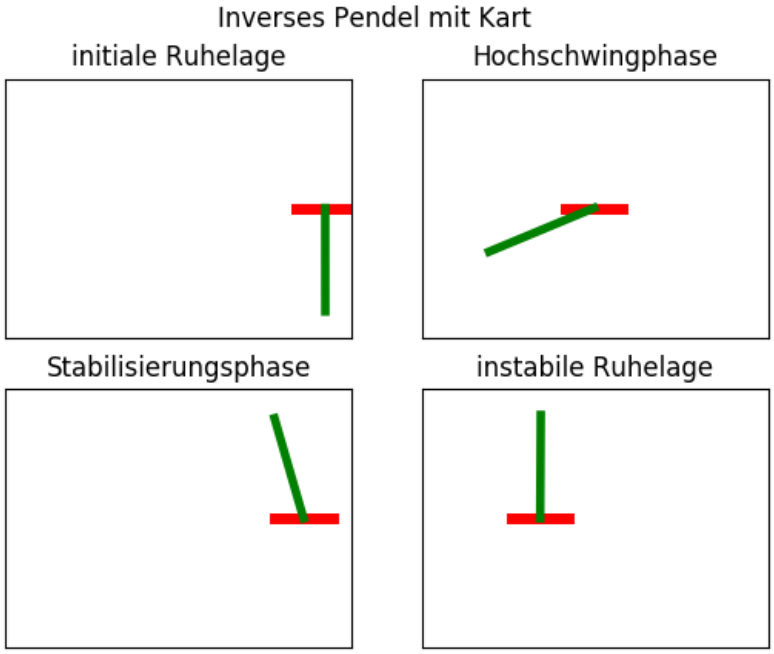

Abbildung 6.2: Die vier Zustände des Pendels in der Simulation

Abbildung 6.2 zeigt das Pendel (grün), welches seitlich an einem Kart (rot) befestigt ist. Aus einer stabilen Ruhelage, in dem sich das Pendel unterhalb des Karts befindet (*oben links*), heraus, soll es nun in die aufrechte Position in eine stabile Ruhelage (*unten rechts*) überhalb des Karts gebracht werden.
Dazu schwingt das Kart durch Hin- und Herfahren das Pendel hoch und stabilisiert sich (*unten links*) ab einem Winkel $\vartheta < \frac{\pi}{5} \sim 36\,\text{deg}$. Danach soll es in der instabilen Ruhelage gehalten werden und bei einer Änderung der Kart-Position möglichst schnell wieder die instabile Ruhelage erreichen.

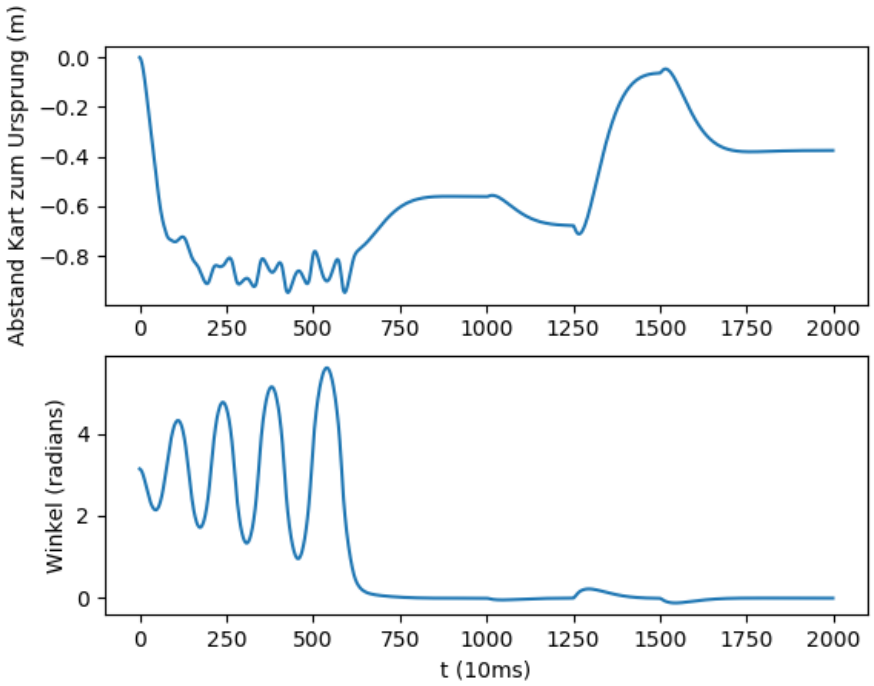

Abbildung 6.3: Pendelauslenkung und Kartposition über die Zeit

6.2.1 Ablauf und Ergebnisse

Wir untersuchen zunächst das Verhalten des LQ-Reglers in der Simulation.

In Abbildung 6.3 sehen wir den zeitlichen Verlauf der Position des Karts und der Auslenkung des Pendels über eine Dauer von 20 Sekunden ($T = 2000$)).
Das Kart sollte hierbei bei $t = 1000$, $t = 1250$ und $t = 1500$ an eine vorgegebene Stelle fahren, das Pendel balancieren und sich dann stabilisieren. Gut erkennbar ist die Aufschwingphase bis ca. $t = 600$, sowie die Stabilisierungsphase nach der Aufschwingphase und nach dem gewollten Verändern der Kartposition.

Wenn sich die Pendelmasse nach der Auslegung des Reglers von $m = 0,3kg$ auf $m = 0,25kg$ ändert, braucht der Regler deutlich länger um sich in die instabile Ruhelage zu bringen (s. Abbildung 6.4).

Motiviert durch dieses Ergebnis, versuchen wir nun die Regelstrecke zu lernen, mit dem Ziel geringere Performanz-Verluste zu haben, als der LQ-Regler.

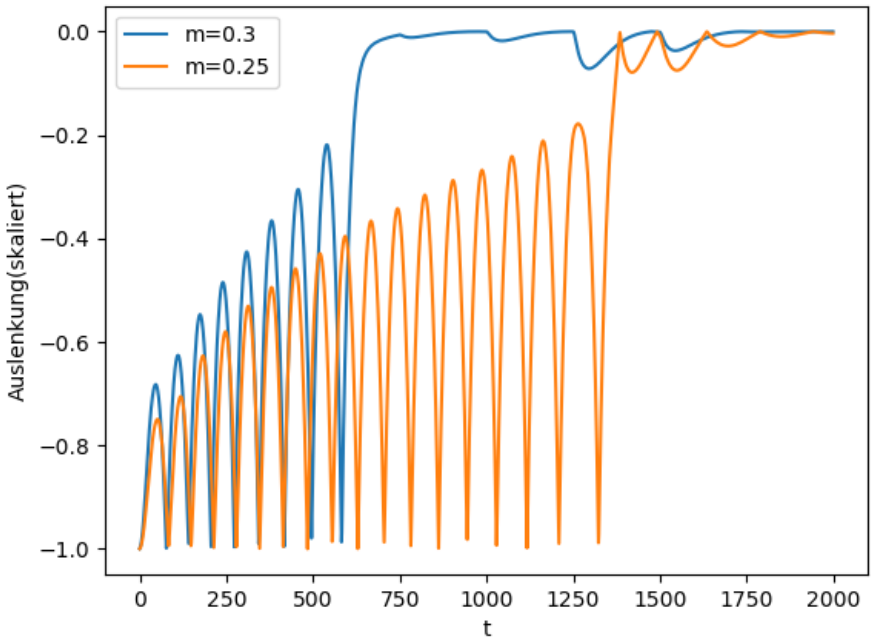

Abbildung 6.4: Vergleich der Reglerabweichung ϑ skaliert auf $[-1, 0]$

Evaluation bei binärer Klasse:

Zunächst führen wir die Simulation des inversen Pendels mit Pendelmasse $m = 0,3kg$ und Kartmasse $M = 0,3kg$ über einen Zeitraum von 20 Sekunden durch. Für die dabei aufgezeichneten Label gilt $\mathbf{y}_i \in \{0, 1\}$. Ein Label 0 bedeutet hier eine Regelung des Kartes nach links auf der x-Achse, Label 1 eine Bewegung nach rechts.

Im nächsten Schritt trainieren wir ein CRF auf dem Datensatz $\mathcal{D}_{sup} = \{(\vec{x_1}, \mathbf{y}_1), \ldots, (\vec{x_T}, \mathbf{y}_T)\}$. Dabei ist $\vec{x_i} = (x_i, \dot{x}_i, \vartheta_i, \dot{\vartheta}_i, m_i)$ und $\mathbf{y}_i \in \{0, 1\}$. Der Datensatz wird dann gemäß POEM mit der durch das CRF gelernten *logging policy* h_0 um Label y_i, δ_i und p_i erweitert und wir erhalten $\mathcal{D} = \{(\vec{x}_1, y_1, \delta_1, p_1), \ldots, (\vec{x}_T, y_T, \delta_T, p_T)\}$.

Nachdem der Bandit Learner auf $\mathcal{D}$ trainiert wurde, vergleichen wir die Winkel des Pendels für je einen Schritt, die aus der Regelung durch Label aus h_0 und dem optimierten h_w folgen.

Wir wiederholen das Experiment mit einem größeren Datensatz, der zusätzlich zu dem aus Experiment 1 noch Daten eines Simulationsdurchlaufs mit geringerer Pendelmasse

$m = 0,25kg$ beinhaltet.

Wir evaluieren auf einer Simulation, bei dem sich die Pendelmasse nach Auslegung des Reglers von $m = 0,3kg$ auf $m = 0,25kg$ ändert, wie in Abbildung 6.4 gezeigt.

Für den Fall $m = 0,25kg$ erhalten wir folgenden Verlauf der Auslenkung:

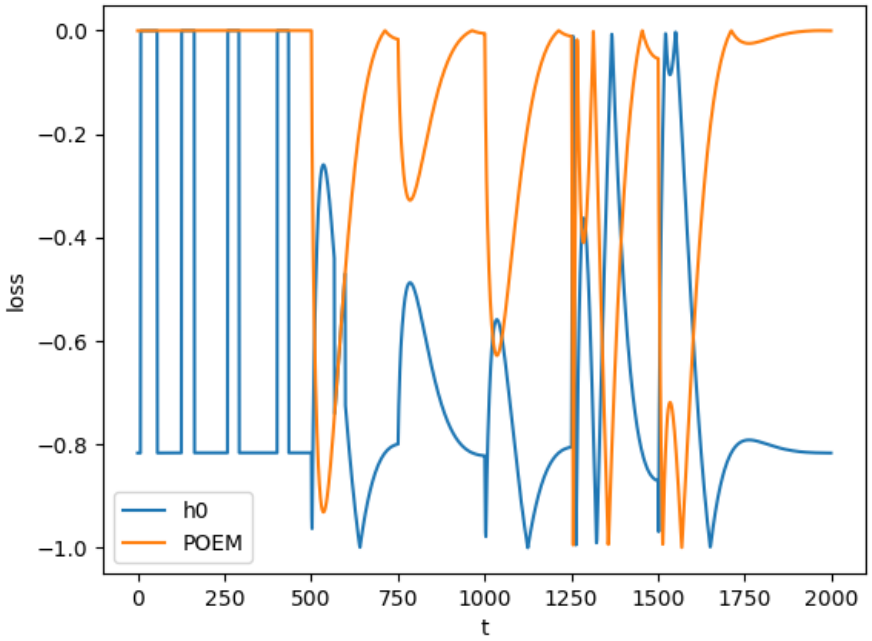

Abbildung 6.5: Loss h_0 und h bei $m = 0,25kg$

Das ganze wird 10 mal wiederholt und es ergeben sich folgende Daten für die durchschnittliche Pendelauslenkung:

	$m = 0,3$	$m = 0,25$
h_0	$-0,684$	$-0,687$
POEM	$-0,181$	$-0,182$

Dabei sind die Werte mit Skalierungsfunktion ϕ (Gleichung (6.29)) skaliert. Wir sehen also, dass sowohl h_0 als auch die durch POEM gefundene Hypothese h nicht viel schlechter sind und besser mit Änderungen im System klarkommen, wenn sie als Kontextinformationen gegeben werden. Ebenfalls ist zu vermerken, dass h im Schnitt bessere Ergebnisse erzielt als h_0.

Die tatsächliche Regelung über die Dauer einer Simulation funktioniert jedoch mit binären Labeln nicht. Das Problem besteht vorraussichtlich darin, dass die binären Labels zu

ungenau sind, da mit ihnen nicht ausgedrückt werden kann, mit welcher Beschleunigung sich das Kart bewegen soll.

Kapitel 7

Fazit

Die Experimente haben gezeigt, dass ein Lernen des Reglerverhaltens generell mit POEM möglich ist, eine Regelung eines Systems wie dem inversen Pendel mit LQ-Regler jedoch problematisch ist.
Zukünftige Forschung könnte weitere Bandit Learning-Algorithmen mit stetigen Vorhersagen testen, die ein, wie in Kapitel 5 vorgeschlagen, Modell für die Approximierung einer Regelmatrix lernen.
Eine weitere Option wäre es, die Regelung mit Entscheidungsbäumen anknüpfend an [24] und Bandit Learning zu untersuchen.

Abbildungsverzeichnis

Algorithmenverzeichnis

Literaturverzeichnis

[1] *Dynamic System Modeling and Control.* http://engineeronadisk.com/.

[2] *Inverted Pendulum: System Modeling.* http://ctms.engin.umich.edu.

[3] *Modeling an Inverted Pendulum.* https://www.ee.usyd.edu.au/tutorials_online/matlab/examples/pe

[4] *Python Control Library 0.7.0.* https://pypi.python.org/pypi/control/0.7.0.

[5] Anderson, C. W.: *Learning to control an inverted pendulum using neural networks.* IEEE Control Systems Magazine, 9(3):31–37, April 1989.

[6] Åström, K.J. und B. Wittenmark: *Adaptive Control.* Dover Books on Electrical Engineering. Dover Publications, 2008.

[7] Barto, A. G., R. S. Sutton und C. W. Anderson: *Neuronlike adaptive elements that can solve difficult learning control problems.* IEEE Transactions on Systems, Man, and Cybernetics, SMC-13(5):834–846, Sept 1983.

[8] Bellman, Richard: *Adaptive control processes: a guided tour.* 1961.

[9] Bergemann, Dirk und Juuso Valimaki: *Bandit Problems.* January 2006.

[10] Bertram, Torsten: *Vorlesungsskript: Steuerungs- und Regelungstechnik I*, 2013.

[11] Gross, Dietmar, Werner Hauger, Jörg Schröder und Wolfgang Wall: *Kinetik.* Springer Verlag, 2015.

[12] Heumann, Christian, Gerhard Tutz, Iris Pigeot, Ludwig Fahrmeir und Rita Künstler: *Statistik.* Springer-Lehrbuch. Springer, 2016.

[13] Ionides, Edward L: *Truncated Importance Sampling.* Journal of Computational and Graphical Statistics, 17(2):295–311, 2008.

[14] Kober, Jens, J Andrew Bagnell und Jan Peters: *Reinforcement learning in robotics: A survey.* The International Journal of Robotics Research, 32(11):1238–1274, 2013.

[15] Komarek, Paul: *Logistic Regression for Data Mining and High-Dimensional Classification.* Technischer Bericht CMU-RI-TR-04-34, Pittsburgh, PA, May 2004.

[16] Lafferty, J., A. McCallum und F. Pereira: *Conditional random fields: Probabilistic models for segmenting and labeling sequence data.* In: *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, Seiten 282–289, 2001.

[17] Lewis, Frank L und Draguna Vrabie: *Reinforcement learning and adaptive dynamic programming for feedback control.* IEEE circuits and systems magazine, 9(3), 2009.

[18] Lewis, Frank L, Draguna Vrabie und Kyriakos G Vamvoudakis: *Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers.* IEEE Control Systems, 32(6):76–105, 2012.

[19] Lunze, J.: *Regelungstechnik 1: Systemtheoretische Grundlagen, Analyse Und Entwurf Einschleifiger Regelungen.* Regelungstechnik / Jan. Lunze. Springer, 2005.

[20] Lunze, J.: *Regelungstechnik 2*, 2008.

[21] MIT, Open Courses: *Maneuvering and Control of Surface and Underwater Vehicles*, 2004.

[22] Morik, Katharina: *Vorlesungsskript: Maschinelles Lernen und Data Mining*, 2013.

[23] Peng, Chao-Ying Joanne, Kuk Lida Lee und Gary M. Ingersoll: *An Introduction to Logistic Regression Analysis and Reporting.* The Journal of Educational Research, 96(1):3–14, 2002.

[24] Pyeatt, L.D., A.E. Howe et al.: *Decision tree function approximation in reinforcement learning.* In: *Proceedings of the Third International Symposium on Adaptive Systems: Evolutionary Computation and Probabilistic Graphical Models*, Band 2, Seiten 70–77, 2001.

[25] Ross, S.M.: *Introduction to Probability Models.* Academic Press, 2009.

[26] Sifleet, Todd: *Inverted Pendulum.* http://www.toddsifleet.com/projects/inverted-pendulum.

[27] Sutton, Richard S. und Andrew G. Barto: *Introduction to Reinforcement Learning.* MIT Press, Cambridge, MA, USA, 1st Auflage, 1998.

[28] Sutton, Richard S, Andrew G Barto und Ronald J Williams: *Reinforcement learning is direct adaptive optimal control.* IEEE Control Systems, 12(2):19–22, 1992.

[29] Swaminathan, Adith und Thorsten Joachims: *Counterfactual Risk Minimization: Learning from Logged Bandit Feedback.* In: *ICML*, Seiten 814–823, 2015.

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet sowie Zitate kenntlich gemacht habe.

Dortmund, den 6. November 2017

Pierre Haritz