

# Computational Engineering und Robotik

Prof. Jan Peters, J. Carvalho, P. Klink, P. Liu, S. Stark

Sommersemester 2020



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

9. Übung

## Hinweise zu dieser Übung

- Für die Teilnahme an der Übung ist eine Anmeldung beim **Moodle** notwendig. Dort sind auf der Kursseite zusätzliche Informationen zur Veranstaltung und die Regelungen zur Anrechnung der Übungsleistungen in die Endnote aufgeführt:

<https://moodle.tu-darmstadt.de/course/view.php?id=20214>

- Von dieser Übung wird die **1. Aufgabe** bewertet.
- Abgabe der schriftlichen Lösungen** zu den Übungsaufgaben bis **Montag, den 06.07.2019 um 11:59 Uhr** per Datei-Upload im Moodle.
- Lade im Moodle die Lösungen hoch als: **UebungXX.pdf** (z.B. für die Übung 04, Uebung04.pdf); **taskX.py** (z.B. für die zweite Programmieraufgabe task2.py). Gib deine Gruppennummer sowie die Namen und Matrikelnummern aller Beteiligten am Anfang der Datei an.
- Begründe alle deine Antworten. Falls du einen Scan deiner Antworten hochlädst, stelle sicher, dass sie lesbar sind.
- Stelle Fragen zu dieser Übung bitte im Moodle-Forum oder in den Sprechstunden.

## 1 Linear Least Squares Regression (12 Punkte)

In der Vorlesung haben wir zwei Arten von Modellen zur Systemidentifikation gesehen: White-Box und Black-Box. Ein Typ von Black-Box-Modellen ist die *Linear Least Squares Regression*. In dieser Übung beschäftigen wir uns mit den Grundlagen der linearen Regression.

- Was sind die Unterschiede zwischen White-Box- und Black-Box-Modellen? Was sind die Vor- und Nachteile des einen oder des anderen?
- Ausgehend von einem parametrisierten linearen Modell  $\hat{y} = \theta^T \psi(x)$  wollen wir die Parameter  $\theta$  so finden, dass unsere Schätzung nahe an den Daten  $D = \{(x_i, y_i)\}$  liegt, wo  $x_i \in \mathbb{R}^{d_x}$ ,  $y_i \in \mathbb{R}$ ,  $\theta \in \mathbb{R}^{d_\theta}$  und  $\psi: \mathbb{R}^{d_x} \rightarrow \mathbb{R}^{d_\theta}$ . Die optimalen Parameter  $\theta^*$  können durch Minimierung der Funktion der *Least Squares Error* gefunden werden

$$\theta^* = \arg \min_{\theta} L(\theta) = \arg \min_{\theta} \frac{1}{N} \left( y - \psi^T \theta \right)^T \left( y - \psi^T \theta \right), \quad (1)$$

mit den gesammelten Zieldaten  $y \in \mathbb{R}^N$ , und der *Feature Matrix*  $\psi \in \mathbb{R}^{d_\theta \times N}$ , die in den Spalten die Ausgabe der *Feature function*  $\psi(x)$  für jeden Datenpunkt enthält.

Finde analytisch den Parametervektor  $\theta$ , der die *Least Squares Error*  $L(\theta)$  in (1) minimiert.

*Hinweise:*  $(AB)^T = B^T A^T$ .  $\nabla_z (z^T A z) = 2Az$ , falls  $A$  quadratisch und symmetrisch ist.  $\nabla_z (z^T a) = \nabla_z (a^T z) = a$

- c) Wenn wir bei der Anpassung eines parametrisierten Modells an die Daten nicht vorsichtig sind, kann es passieren, dass das Modell sich zu sehr an die Daten anpasst (*overfit* auf Englisch), d.h. der durchschnittliche Fehler (*Mean Squared Error*) ist für die Trainingsdaten fast null, aber wenn du mit dem Modell eine Vorhersage in einem neuen Punkt  $x$  machst, wird dein Ergebnis völlig vom wahren Ergebnis abweichen. Eine Möglichkeit, das *overfit* zu reduzieren, besteht darin, der Kostenfunktion  $L(\theta)$  eine Strafe in den Parametern hinzuzufügen. Wir fügen eine Strafe proportional zur Norm von  $\theta$  hinzu (heißt *Ridge Regression*),  $\lambda \theta^T \theta$  für  $\lambda > 0$ , die Kostenfunktion ist dann  $L(\theta) + \lambda \theta^T \theta$ . Finde analytisch mit dieser modifizierten Kostenfunktion wieder die neuen optimalen Parametern.

- d) Bitte lade die Datei uebung09-programming.zip aus dem Moodle herunter.

Wir werden einen Roboter in einem unwegsamen Gelände einsetzen und möchten wissen, welche Kraft auf die Geschwindigkeit des Roboters wirkt. Wir gehen davon aus, dass es einen linearen Zusammenhang zwischen der resultierenden Kraft und der Geschwindigkeit des Roboters gibt (die Kraft kann ungleich Null sein, wenn sich der Roboter nicht bewegt, z.B. unter der Annahme, dass eine konstante äußere Kraft vorhanden ist). Wir bilden diese lineare Beziehung als Black-Box-Modell  $\hat{y} = \theta^T \psi(x)$  ab, bei dem  $x$  die Geschwindigkeit und  $y$  die Kraft sind. Wir sammeln  $N$  Messungen aus dem realen System in Paaren  $(x_i, y_i)$ , wobei  $x_i$  die Geschwindigkeit des Roboters und  $y_i$  die Kraft ist (angenommen, wir können die Geschwindigkeit einstellen und die Kraft mit einem Gerät messen).

Du bekommst zwei Datensätze von zwei verschiedenen Sensoren  $D_a = \{(x_i, y_i)\}_{i=1}^N$  und  $D_b = \{(x_i, y_i)\}_{i=1}^N$ , in den Dateien `x_a.npy` und `y_a.npy` bzw. `x_b.npy` und `y_b.npy`.

Implementiere ein Programm in Python, das die optimalen Parameter  $\theta^*$  von der *Linear Least Squares Regression* berechnet. Du kannst als Ausgangspunkt die in der .zip-Datei enthaltene Python-Datei verwenden. Nimm an, dass dein Modell linear in den Parametern ist,  $\hat{y} = \theta^T \psi(x)$ , und die *feature function* ist  $\psi(x) : x \mapsto [1, x]^T$ .

Gib für jeden Datensatz die optimalen Parameter  $\theta^*$  und den *Mean Squared Error* (MSE)  $L(\theta)$  an. Zeichne die gesammelten Daten und die Vorhersage der gesammelten Daten (z.B. mit der `scatter`-Funktion von Matplotlib), sowie eine Kurve mit der Vorhersage für jeden Punkt im Intervall der gesammelten Daten (z.B. mit der `plot`-Funktion von Matplotlib), d.h. wenn die gesammelten Daten  $x$  zwischen -2 und 2 liegen, zeichne die Funktion zwischen -2 und 2.

**Bitte lade auch deine bearbeitete Python-Datei hoch. Ändere den Namen der Datei nicht. Füge einfach deine Informationen (Name, Immatrikulationsnummer, Gruppennummer) am Anfang der Datei hinzu.**

- e) Die Daten von d) wurden mit zwei verschiedenen Sensoren erstellt, einem sehr billigen, daher haben die Messungen ein hohes Rauschen, und einem sehr teuren, und daher haben die Messungen weniger Rauschen. Welcher Datensatz wurde mit welchem Sensor erzeugt?
- f) Die in d) angenommene lineare Beziehung ist vielleicht nicht der richtige Weg, um das Kraft-Geschwindigkeits-Verhältnis zu modellieren. Hier haben wir mehr Daten in einer anderen Art von Gelände gesammelt. Wir wissen nicht mehr, ob es eine lineare Beziehung zwischen Kraft und Geschwindigkeit gibt, so dass wir keine lineare *identity feature function* mehr verwenden können. Deshalb werden wir nun verschiedene Typen von *feature functions* verwenden, um zu sehen, wie sich die Anpassung der Daten verhält. Wir werden die folgenden *feature functions* verwenden

Linear features:  $\psi(x) = [1, x]^T$

Quadratic features:  $\psi(x) = [1, x, x^2]^T$

Cubic features:  $\psi(x) = [1, x, x^2, x^3]^T$

Quartic features:  $\psi(x) = [1, x, x^2, x^3, x^4]^T$

Tenth features:  $\psi(x) = [1, x, x^2, x^3, \dots, x^{10}]^T$

Twentieth features:  $\psi(x) = [1, x, x^2, x^3, \dots, x^{20}]^T$

Wichtig: die *features* korrelieren nicht zwischen den Dimensionen von  $x$ , d.h.

$\psi(x) = [1, x, x^2]^T = [1, x_1, x_2, \dots, x_{d_x}, x_1^2, \dots, x_{d_x}^2]^T$ , und das analog für die andere *features*.

Die Messungen des Sensors sind im Datensatz  $D_c = \{(x_i, y_i)\}_{i=1}^N$  gesammelt, zu finden in den Dateien `x_c.npy` und `y_c.npy`.

---

Implementiere die lineare Regression mit den verschiedenen *feature functions* und für den Datensatz  $D_c$  unter Verwendung der Python-Datei von d).

Gib für jeden Typ *feature function* die optimalen Parameter  $\theta^*$  und den *Mean Squared Error* (MSE)  $L(\theta)$  an und zeichne die gesammelten Daten sowie die Vorhersage der gesammelten Daten (z.B. mit die `scatter`-Funktion von Matplotlib), und eine Kurve mit der Vorhersage für jeden Punkt im Intervall der gesammelten Daten (z.B. mit der `plot`-Funktion von Matplotlib), d.h. wenn die gesammelten Daten  $x$  zwischen -2 und 2 liegen, zeichne zwischen -2 und 2.

Was kannst du aus den Ergebnissen über das Verhältnis zwischen der Kraft und der Geschwindigkeit sagen?

**Bitte lade auch deine bearbeitete Python-Datei hoch. Ändere den Namen der Datei nicht. Füge einfach deine Informationen (Name, Immatrikulationsnummer, Gruppennummer) am Anfang der Datei hinzu.**

- g) Wir wollen nun ein lineares Modell in einem echtem Roboter verwenden, um in Echtzeit zu arbeiten. Wir wollen das Verhältnis zwischen der Beschleunigung des Roboters und allen anderen Sensormessungen modellieren. Wir werden annehmen, dass diese Beziehung in den Parametern unseres Modells linear ist. Zu diesem Zweck sammeln wir Daten und wollen unser Modell so schnell wie möglich an den Roboter anpassen, damit es auf dem Roboter eingesetzt werden kann. Statt dass  $x$  eine einzige reelle Zahl ist, handelt es sich jetzt um einen Vektor von Sensormessungen, und wir haben insgesamt etwa 10000 Sensoren, und vielleicht möchten wir in Zukunft sogar noch mehr hinzufügen.

Nehmen wir zunächst einmal an, dass wir nur eine *feature function* wie in d) verwenden. In der Theorie können wir die optimalen Parameter  $\theta^*$  in analytischer Form berechnen. Dies könnte jedoch problematisch sein, wenn wir die Ergebnisse so schnell wie möglich erhalten wollen. Warum ist das so? Was ist das Problem bei der Verwendung der analytischen Form? Um es dir leicht zu machen, kannst du bedenken, dass du nur wenige Datenpunkte hast, zum Beispiel  $N = 100$ .

*Hinweise: Denke an die Komplexität der Berechnung von  $\theta^*$  im Sinne der Big-O-Notation.*

---

## Hinweis zu wissenschaftlichem Arbeiten

---

Es ist nicht gestattet, Lösungen anderer Personen als die der Gruppenmitglieder als Lösung der Aufgabe abzugeben. Des Weiteren müssen alle zur Lösungsfindung verwendeten, darüber hinausgehenden, relevanten Quellen explizit angegeben werden. Dem widersprechendes Handeln ist Plagiarismus und ist ein ernster Verstoß gegen die Grundlagen des wissenschaftlichen Arbeitens, das ernsthafte Konsequenzen bis hin zur Exmatrikulation haben kann.