Computational Engineering und Robotik

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

Sommersemester 2020

TECHNISCHE UNIVERSITÄT DARMSTADT

Lösungsvorschlag der 9. Übung

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.

a) Was sind die Unterschiede zwischen White-Box- und Black-Box-Modellen? Was sind die Vor- und Nachteile des einen oder des anderen?

Lösungsvorschlag: (1.0 Punkt(e))**White-Box Modellen**: Nutzten unser physikalisches Wissen, um parametrische Funktion festzulegen.

- Vorteil: Wir können Vorkenntnisse in unser Modell einbeziehen; In einigen Fällen kann dies die Komplexität des Erlernens der Parameter reduzieren
- Nachteil: Was wir nicht modellieren, kann nicht repräsentiert werden; Un- oder falsch modellierte Aspekte können systematisch die Parameterschätzung verfälschen

Black-Box Modellen: Anstatt komplizierten Physik Effekte zu modellieren, kann man direkt mit Modellen $y = f(x;\theta)$ arbeiten, welches sehr viele verschiedene Funktionen darstellen kann

- Vorteil: Das Training dieser Modelle ist manchmal einfacher als White-Box-Modelle; robuster gegenüber Rauschen bei den Messungen
- Nachteil: Keine physikalischen Prinzipien werden verwendet um das Modell herzuleiten

b) Ausgehend von einem parametrisierten linearen Modell $\hat{y} = \boldsymbol{\theta}^T \psi(\boldsymbol{x})$ wollen wir die Parameter $\boldsymbol{\theta}$ so finden, dass unsere Schätzung nahe an den Daten $D = \{(\boldsymbol{x}_i, y_i)\}$ liegt, wo $\boldsymbol{x}_i \in \mathbb{R}^{d_x}$, $y_i \in \mathbb{R}$, $\boldsymbol{\theta} \in \mathbb{R}^{d_\theta}$ und $\psi : \mathbb{R}^{d_x} \to \mathbb{R}^{d_\theta}$. Die optimalen Parameter $\boldsymbol{\theta}^*$ können durch Minimierung der Funktion der *Least Squares Error* gefunden werden

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

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

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

Hinweise: $(\boldsymbol{AB})^T = \boldsymbol{B}^T\boldsymbol{A}^T$. $\nabla_{\boldsymbol{z}} \left(\boldsymbol{z}^T\boldsymbol{A}\boldsymbol{z}\right) = 2\boldsymbol{A}\boldsymbol{z}$, *falls* $\boldsymbol{A}$ *quadratisch und symmetrisch ist.* $\nabla_{\boldsymbol{z}} \left(\boldsymbol{z}^T\boldsymbol{a}\right) = \nabla_{\boldsymbol{z}} \left(\boldsymbol{a}^T\boldsymbol{z}\right) = \boldsymbol{a}$

Lösungsvorschlag: (2.0 Punkt(e))

$$
\begin{aligned}
L(\boldsymbol{\theta}) &= \frac{1}{N}\left(\boldsymbol{y} - \boldsymbol{\psi}^T\boldsymbol{\theta}\right)^T\left(\boldsymbol{y} - \boldsymbol{\psi}^T\boldsymbol{\theta}\right) \\
&= \frac{1}{N}\left(\boldsymbol{y}^T\boldsymbol{y} - \boldsymbol{y}^T\boldsymbol{\psi}^T\boldsymbol{\theta} - \boldsymbol{\theta}^T\boldsymbol{\psi}\boldsymbol{y} + \boldsymbol{\theta}^T\boldsymbol{\psi}\boldsymbol{\psi}^T\boldsymbol{\theta}\right) \\
\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) &= -2\boldsymbol{\psi}\boldsymbol{y} + 2\boldsymbol{\psi}\boldsymbol{\psi}^T\boldsymbol{\theta} = 0 \\
\boldsymbol{\theta}^* &= \left(\boldsymbol{\psi}\boldsymbol{\psi}^T\right)^{-1}\boldsymbol{\psi}\boldsymbol{y}
\end{aligned}
$$

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 durchschnittlicher Fehler (*Mean Squared Error*) ist für die Trainingsdaten fast null, aber wenn du mit dem Modell eine Vorhersage in einem neuem Punkt $\boldsymbol{x}$ machst, wird dein Ergebnis völlig vom wahren Ergebnis abweichen. Eine Möglichkeit, das *overfit* zu reduzieren, besteht darin, der Kostenfunktion $L(\boldsymbol{\theta})$ eine Strafe in den Parametern hinzuzufügen. Wir fügen eine Strafe proportional zur Norm von $\boldsymbol{\theta}$ hinzu (heißt *Ridge Regression*), $\lambda\boldsymbol{\theta}^T\boldsymbol{\theta}$ für $\lambda > 0$, die Kostenfunktion ist dann $L(\boldsymbol{\theta}) + \lambda\boldsymbol{\theta}^T\boldsymbol{\theta}$. Finde analytisch mit dieser modifizierten Kostenfunktion wieder die neuen optimalen Parametern.

Lösungsvorschlag: (1.0 Punkt(e))

$$
\begin{aligned}
L(\boldsymbol{\theta}) &= \frac{1}{N}\left(\boldsymbol{y} - \boldsymbol{\psi}^T\boldsymbol{\theta}\right)^T\left(\boldsymbol{y} - \boldsymbol{\psi}^T\boldsymbol{\theta}\right) + \lambda\boldsymbol{\theta}^T\boldsymbol{\theta} \\
\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) &= -\frac{2}{N}\boldsymbol{\psi}\boldsymbol{y} + 2\boldsymbol{\psi}\boldsymbol{\psi}^T\boldsymbol{\theta} + 2\lambda\boldsymbol{\theta} = 0 \\
\boldsymbol{\theta}^* &= \left(\boldsymbol{\psi}\boldsymbol{\psi}^T + \lambda N\boldsymbol{I}\right)^{-1}\boldsymbol{\psi}\boldsymbol{y}
\end{aligned}
$$

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 $\hat{y} = \boldsymbol{\theta}^T\psi(\boldsymbol{x})$ ab, bei dem $\boldsymbol{x}$ die Geschwindigkeit und y die Kraft sind. Wir sammeln N Messungen aus dem realen System in Paaren $(\boldsymbol{x}_i, y_i)$, wobei $\boldsymbol{x}_i$ 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 $D_a = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ und $D_b = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, 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 $\boldsymbol{\theta}^*$ 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, $\hat{y} = \boldsymbol{\theta}^T\psi(\boldsymbol{x})$, und die *feature function* ist $\psi(\boldsymbol{x}) : \boldsymbol{x} \mapsto [1, \boldsymbol{x}]^T$.

Gib für jeden Datensatz die optimalen Parameter $\boldsymbol{\theta}^*$ und den *Mean Squared Error* (MSE) $L(\boldsymbol{\theta})$ 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, zeichnedie Funtion zwischen -2 und 2.

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

Lösungsvorschlag: (3.0 Punkt(e))Siehe die Lösung in der Python Datei.

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?

Lösungsvorschlag: (1.0 Punkt(e))Wir können die *Mean Squared Error* (MSE) vergleichen. Die Daten mit der niedrigsten MSE stammen von dem Sensor, der Ergebnisse mit weniger Rauschen liefert.

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

$$\begin{aligned}
&\text{Linear features: } \psi(\boldsymbol{x}) = [1, \boldsymbol{x}]^T \\
&\text{Quadratic features: } \psi(\boldsymbol{x}) = \left[1, \boldsymbol{x}, \boldsymbol{x}^2\right]^T \\
&\text{Cubic features: } \psi(\boldsymbol{x}) = \left[1, \boldsymbol{x}, \boldsymbol{x}^2, \boldsymbol{x}^3\right]^T \\
&\text{Quartic features: } \psi(\boldsymbol{x}) = \left[1, \boldsymbol{x}, \boldsymbol{x}^2, \boldsymbol{x}^3, \boldsymbol{x}^4\right]^T \\
&\text{Tenth features: } \psi(\boldsymbol{x}) = \left[1, \boldsymbol{x}, \boldsymbol{x}^2, \boldsymbol{x}^3, \ldots, \boldsymbol{x}^{10}\right]^T \\
&\text{Twentieth features: } \psi(\boldsymbol{x}) = \left[1, \boldsymbol{x}, \boldsymbol{x}^2, \boldsymbol{x}^3, \ldots, \boldsymbol{x}^{20}\right]^T
\end{aligned}$$

Wichtig: die *features* korrelieren nicht zwischen den Dimensionen von $\boldsymbol{x}$, d.h. $\psi(\boldsymbol{x}) = \left[1, \boldsymbol{x}, \boldsymbol{x}^2\right]^T = \left[1, x_1, x_2, \ldots, x_{d_x}, x_1^2, \ldots, x_{d_x}^2\right]^T$, und das analog für die andere *features*.

Die Messungen des Sensors sind im Datensatz $D_c = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ 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 $\boldsymbol{\theta}^*$ und den *Mean Squared Error* (MSE) $L(\boldsymbol{\theta})$ 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.

Lösungsvorschlag: (3.0 Punkt(e))Schaue die Lösung in der Python Datei.

Von der Plots schauen wir dass das Verhältnis zwischen der Kraft und der Geschwindigkeit nicht linear ist. Wenn sie linear wäre, dann würde eine lineare *feature function* gut zu den Daten passen.

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 $\boldsymbol{x}$ 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 $\boldsymbol{\theta}^*$ 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 $\boldsymbol{\theta}^*$ im Sinne der Big-$\mathcal{O}$-Notation.

Lösungsvorschlag: (1.0 Punkt(e))Das Problem bei der analytischen Lösung liegt in der Berechnung der Inversen einer Matrix. Die Matrix $\boldsymbol{\psi}\boldsymbol{\psi}^T$ ist $D \times D$. Die Berechnung der Inversen einer Matrix kostet $\mathcal{O}(D^3)$, so dass für große D die Berechnungszeit problematisch wird. Extra: Die Invertierung einer Matrix kann auch zu numerischen Fehlern führen.