Summercamp 2022

Projekt Automatisiertes Fahren

Karthikeyan Chandra Sekaran 01.03.2022

Outline

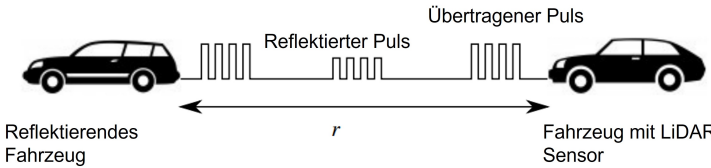
Szenariobeschreibung

Das mit LiDAR- und ADMA-Sensorik ausgestattete Ego-Fahrzeug folgt einem Fahrzeug der gleichen Spur, welches überraschend eine Notbremsung einleitet. Um den Auffahrunfall zu verhindern, ist der kritsche Zeitpunkt t_{crit} zu bestimmen, zu dem das Ego-Fahrzeug spätestens seine Bremsung einleiten muss.

Ziele:

1. Verarbeitung der Sensorrohdaten des Szenarios
2. Entwicklung eines KI-Algorithmus zur Ermittlung der kritischen Zeit t_{crit}.
3. Anwendung der Algorithmen in einer Echtzeit-Simulation des Szenarios.

Fragen zur Szenariobeschreibung



Fragen:

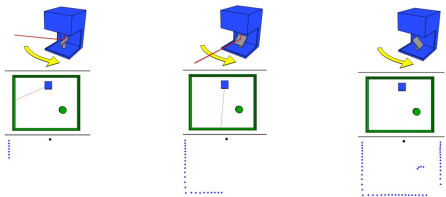
1. Welche Assistenzsysteme sind in Fahrzeugen mit LiDAR-Sensor vorstellbar?
2. Welches Assistenzsystem könnte mit diesem Aufbau erprobt werden?
3. Wie kann eine Implementierung aussehen? Welche Eingangsgrößen benötigt die Entscheidungslogik?

Sensor Setup
Lidar Sensor

LiDAR...

- ... steht als Akronym für „Light Detection And Ranging“
- ... ist ein optisches Verfahren zur Messung von Distanzen
- ... sendet optisches Laserlicht in Pulsen aus.
- ... arbeitet mit Wellenlängen nahe des sichtbaren Bereichs (μm-Wellen)

Visualisierung des Lidar-Prinzips [Wik.2022]



Sensor Setup
LiDAR Sensor

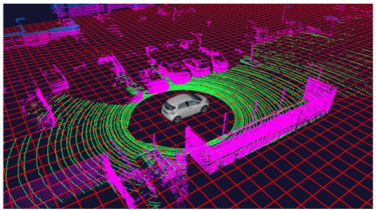
- **Distanzmessung** per „time-of-flight“-Prinzip:
  - Sende und Empfangszeit des Pulses wird gemessen
  - Aus der Zeitdifferenz τ und der Geschwindigkeit des Lichts $c = 3 \cdot 10^8 \frac{m}{s}$

$$r = \frac{c}{2} \cdot \tau \tag{1}$$

- **Geschwindigkeitsbestimmung:** Nur indirekt auf Basis von Signalverarbeitungstechniken aus der Distanzmessung ableitbar
- **Vorteile**: Großer Öffnungswinkel, Detektion aller möglichen Objekte (ohne Training, da aktiver Sensor), Klassifizierung von Objekten möglich, etc.
- **Nachteile:** Wetteranfälligkeit, keine direkte Geschwindigkeitsmessung, etc.

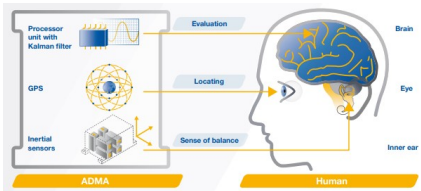
Sensor Setup
LiDAR Sensor: Rohdaten Ausgabe

- **Ausgabeformat:** [x, y, z, Intensität] $\rightarrow$ Nx4 Matrix für eine Punktwolke mit N Punkten
- **Speicheranforderungen** am Beispiel des Velodyne HDL-64E (rotierender 3D Laserscanner, der z.B. im Open-Source Kitti-Datensatz verwendet wird):
  - Sichtfeld: 360° horizontal, 36.8° vertikal | Reichweite: 120 m | Frequenz: 10 Hz
  - Ausgegebene Punkte: $1.3 \cdot 10^6 \ \frac{Punkte}{Sekunde}$
  - Benötigter Speicher: $1.8 \ \frac{MB}{Frame} \cdot 10 \ \frac{Frames}{s} = 18 \ \frac{MB}{s}$
- **Beispiel:** LiDAR-Aufnahme auf einem Parkplatz [Rummelhard.2017]

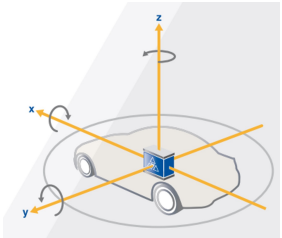


ADMA...

- ... steht als Akronym für „Automotive Dynamic Motion Analyzer“
- ... setzt sich aus IMU, GNSS und einem Prozessor zusammen und kann somit Geschwindigkeit, räumliche Lage des Fahrzeugs und Standort in Echtzeit ausgeben



(a) ADMA als „Sinnesorgan“ des Fahrzeugs [AD14]



(b) ADMA im Fahrzeug [GeneSys.2022]

Datenverarbeitung
Erläuterung

Motivation:

- Filtern von störenden oder für den Verwendungszweck nicht relevanten Anteile
- Reduktion des Speicher- und Rechenbedarfs in folgenden Algorithmen
- Ausgabe der Daten entsprechend der Schnittstellenanforderungen folgender Algorithmen
- Nutzen von A-Priori Wissen zur Verkleinerung des ausgewerteten Lösungsraums für folgende Alogrithmen $\rightarrow$ Algorithmus muss diese Eingrenzung nicht mehr erlernen
- Erfüllung der Echzeitanforderungen

Aufgabenstellung: Ausgabe der Punktewolke für die wichtigen Objekte der Aufnahme

Vorgehensweise:

1. „Region of Interest“-Filterung: Filterung der für den Verwendungszweck unrelvanten Daten
2. „Ground Subtraction“: Filterung der Lidarpunkte der Fahrbahnfläche
3. „Clustering“: Erkennen zusammenhängender Punkte zur Ausgabe von Objekten

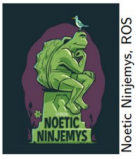
Datenverarbeitung
Erläuterung ROS

Motivation:

- bietet Struktur, Tools und Algorithmen für die Implementierung eigener Applikationen
- Sensorausgabe kann in dieses System eingebunden und somit verarbeitet werden

Allgemein:

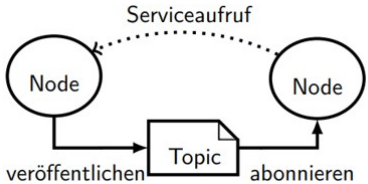
- Open-Source Framework
- Aufgaben: Hardwareabstraktion, Hilfsfunktionen, Interprozesskommunikation, Gerätetreiber, Paketmanagement



Noetic Ninjemys, ROS

Foxy Fitzroy, ROS 2

- **ROS Nodes**: Eigenständige Programme, die ihre vorgebene Logik ausführen
- **ROS Master**: Zentraler Haupt-Knoten im System → kennt die Nodes des Systems und ermöglicht dadurch die Kommunikation
- **ROS Topic**: eigener „Kommunikationskanal“, wodurch Erzeugung und Nutzung von Information entkoppelt ist
- Serviceorientierte Architektur, die mit Publisher-Subscriber Beziehungen arbeitet
- Programmiersprachen: C++, Python



Erklärung der Skripte und des Aufbaus

Datenverarbeitung
Erläuterung der geplanten Architektur

Hier die Verbindung zwischen Node des Sensors und Empfang im implementierter Logik

darstellen Klassendiagramm –> Welche Funktion macht was

Motivation

- Je nach Algorithmus ist nur ein Teil des Datensatzes für dessen Auswertung relevant
- Beispiel für A-Piori Wissen im automatisierten Fahren: HD-Maps
  - Im Fahrzeug verfügbare Information über die Umgebung
  - Daraus ableitbar: Regionen mit möglichem Verkehrsaufkommen und potentielle Gefahrenbereiche
  - Folgerung: gezielte Reduktion des ausgewerteten Datenraums auf den für den Algorithmus relevanten Bereich
  - Beispiel: Adaptive Cruise Control wertet Region aus, in der das vorausfahrendes Fahrzeug vermutet wird, da nur jene für die Aufgabe relevant ist.

Region of Interest
HD-map

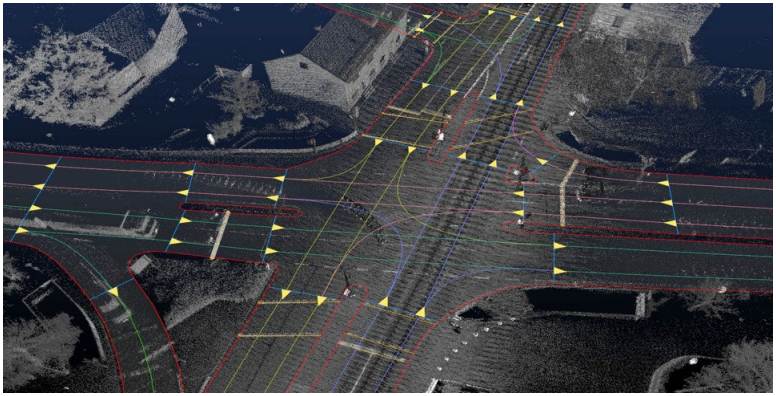


Abbildung: Beispiel einer HD-map [Miller.2022]

Implementierung: Datenverarbeitung
Region of Interest

Anwendung: Wissen über die Teststrecke als im Fahrzeug verfügbare HD-map

Aufgabenstellung: Begrenzung des Datensatzes auf Basis von 4 gegebenen Punkten ${}^{G}P_i$ des globalen Koordinatensystems G der Teststrecke

Datenverarbeitung
Implementierung: Region of Interest

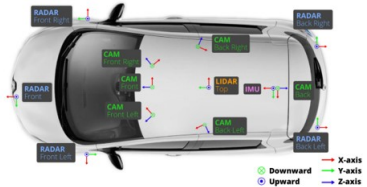
Gegeben:

- Punkte: ${}^{G}P_1 = (x, y, z)$, ${}^{G}P_2 = (x, y, z)$, ${}^{G}P_3 = (x, y, z)$, ${}^{G}P_4 = (x, y, z)$
- Transformationsmatrizen:

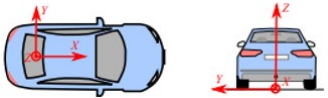
Vorgehensweise:

1. Transformation der LiDAR-Punkte ${}^{L}P_i$ in das globale Koordinatensystem
2. Filterung aller Punkte mit Koordinaten (x, y) außerhalb des durch die 4 Punkte ${}^{G}P_i$ aufgespannten Bereichs

Region of Interest
Implementierungshinweise: Lokale Koordinatensysteme



(a) Sensorkoordinatensysteme in nuScenes [Caesar.2020]



(b) Fahrzeugkoordinatensystem exterozeptiver Sensoren

(c) Fahrzeugkoordinatensystem der Fahrdynamikgleichungen [Breuer.2015]

Region of Interest
Implementierungshinweise: Lokale Koordinatensysteme

- Zahlreiche Koordinatensysteme im Fahrzeug, die eine Betrachtung der Information aus verschiedenen Perspektive ermöglichen
- Zwischen den lokalen Koordinatensystemen des Fahrzeugs kann ein statischer Zusammenhang, analog einer Starrkörperbewegung, angenommen werden $\rightarrow$ Transformationsmatrizen sind zeitunabhängig
- Fahrzeugkoordinatensysteme sind bei der Berechnung zwingend zu beachten, da Sie den Variablen Bedeutung verleihen
- Beispiel einer Fahrdynamikgleichung:

$$ {}^{F}v_x = \frac{\delta}{\delta t} x \text{ gilt für } x = {}^{F}x \tag{2} $$

Region of Interest
Implementierungshinweise: Globale Koordinatensysteme

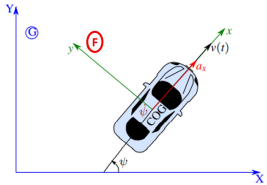


Abbildung: Globales Koordinatensystem

- Stationäres Referenzkoordinatensystem
- Ermöglicht die Beschreibung von Objekten relativ zu einem konstanten Koordinatensystem
- Transformationsmatrix von lokalem Fahrzeugkoordinatensystem zu globalem Koordinatensystem ist für instationäre Objekte zeitabhängig

Region of Interest: Implementierungshinweise
Anwendung auf das Szenario

Nomenklatur:

- ${}^{G}P$: Punkt des globalen Koordinatensystem
- ${}^{L}P$: Punkt des LiDAR Koordinatensystems
- ${}^{F}P$: Punkt des Fahrzeugkoordinatensystem (CoG)
- ${}^{EF}P$: Punkt des Fahrzeugkoordinatensystems der exterozeptiven Sensoren

Region of Interest
Implementierungshinweise: Anwendung auf das Szenario

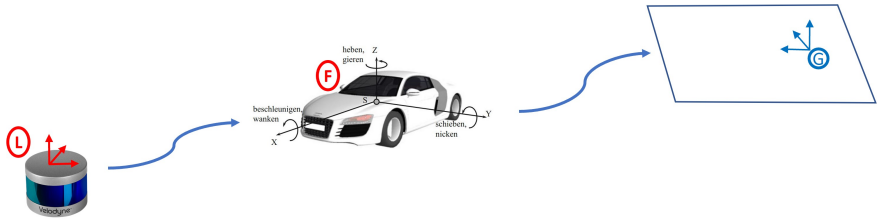


Abbildung: Anwendung der Koordinatentransformation auf das Szenario. Zusammengesetzt aus Abbildungen in [Breuer.2015] und [Velodyne.2020]

- statische Koordinatentransformation $\rightarrow$ zeitabhängige Koordinatentransformation
- ADMA gibt Position des Fahrzeugs im globalen Koordinatensystem
- Umsetzung mit tf-Modul aus ROS

Region of Interest
Implementierungshinweise: Transformation in ROS

auf ROS Transformatoren eingehen

Motivation

- LiDAR Sensor gibt zahlreiche Reflektionen der Fahrbahnoberfläche aus
- Diese Punkte gehören nicht zu den gesuchten Objekte und haben deswegen für die Objekterkennung keine Relevanz

Ziele

- Objektdetektionsalgorithmus muss Vernachlässigung von Reflektionen des Bodens nicht mehr erlernen
- Verkleinerung des Lösungsraums

Datenverarbeitung
Implementierung: Ground Subtraction

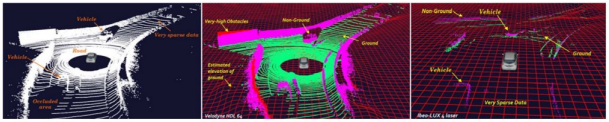
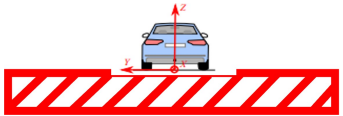


Abbildung: Beispiel Ground Subtraction [Rummelhard.2017]

Anwendung:

- Es wird eine ebene Fahrbahnoberfläche auf der Teststrecke angenommen
- Zur Vereinfachung wird die Eliminierung der Punkte der Fahrbahnoberfläche auf Basis Höhe z durchgeführt
- Alternativen: RANSAC Algorithmus und weitere $\rightarrow$ weniger intuitiv, aber robuster



Aufgabenstellung: Verkleinerung des Datensatzes um Punkte der Fahrbahnoberfläche:

Bedingung ${}^{EF}z \leq 0.01m$

Motivation:

- Verwendung der „Ähnlichkeit“ von Datenpunkten (bspw. Koordinaten), um zusammengehörende Punkte zu erkennen und zu segmentieren
- Ziel ist, die Struktur der Eingangsdaten zu lernen und eine automatische Segmentierung in Gruppen ähnlicher Datenpunkte vorzunehmen

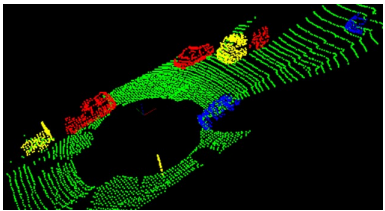


Abbildung: Beispiel Clustering [Hyin.2018]

Anwendung: Angewandt auf unser Szenario bedeutet dies, das einzelne Gruppen LiDAR-Punkte zu Clustern vereint werden. Diese Punktwolken repräsentieren Objekte der Umgebung, wodurch final eine Lokalisierung des vorausfahrenden Fahrzeugs ermöglicht wird.

Aufgabenstellung: Auf Basis des k-Means Clusteralgorithmus sollen die Datenpunkte des vorausfahrenden Autos gruppiert und segmentiert werden.

Allgemein:

- Mehrere Optionen bei der Art der durchgeführten Clusteranalyse: partionierende, modellbasierte, hierachische, wahrscheinlichkeitsdichtebasierte, gitterbasierte Verfahren und Spectral Clustering
- In unserem Fall (k-Means Clustering): partionierendes Cluster-Verfahren, bei dem auf Basis der Merkmale der Datenpunkte ein Clustering vorgenommen wird. Jeder Punkt v_m des Datensatzes $\mathcal{D}$ wird eindeutig einem Cluster $C_1, ..., C_k$ zugewiesen. Das Cluster C_K kann dementsprechend als Partion des Eingangsraums $\mathbb{R}^N$ interpretiert werden.

$$\mathcal{D} = \{v_1, ..., v_M\} \tag{3}$$

Clustering
Implementierungshinweise: k-Means Algorithmus

Beschreibung: Aus einer Menge von ähnlichen Objekten wird eine vorher bekannte Anzahl von k Clustern gebildet. Die Aufgabe wird gelöst, indem jeweils ein Repräsentant c_k für jedes Cluster C_k gesucht wird und zwar derart, dass alle Datenpunkte aus $\mathcal{D}$ nahe an ihrem jeweiligen Repräsentanten sind.

Umsetzung:

- Wenn $d(v_m, c_k)$ eine Distanz zwischen Eingangspunkt v_m und Repräsentant des Clusters c_k darstellt, so lässt sich die zu lösende Optimierungsaufgabe schreiben als

$$\{c_1, ..., c_k\} = \underset{\{c'_1,...,c'_k\}}{\operatorname{argmin}} \left\{ \sum_{m=1}^{M} \min_{k=1,...,K} d(v_m, c'_k) \right\} \tag{4}$$

- Diese Optimierungsaufgabe ist nicht konvex.

- Um die nicht konvexe Optimierungsaufgabe zu lösen wird ein iteratives Verfahren, der k-Means Algorithmus verwendet. Dieser findet ein lokales Optimum.
- Nutzt man im k-Means Algorithmus als Distanzmaß die quadrierte Euklidische Distanz

$$d(v_m, c_k) = \|v_m - c_k\|_2^2 \tag{5}$$

- Wird die Optimierungsaufgabe zu einer Minimierung der quadratischen Abweichungen von den Cluster-Schwerpunkten der Cluster S_i. Es kann von Clustering durch Varianzminimierung gesprochen werden.

$$J = \sum_{i=1}^{k} \sum_{x_j \in S_i} \|v_m - c_k\|_2^2 \tag{6}$$

Umsetzung Lloyd-Algorithmus:

1. Initialisierung: Wähle k zufällige Mittelwerte (Means): $m_1^1, ..., m_k^1$ aus dem Datensatz
2. Zuordnung: Jedes Datenobjekt wird dem Cluster zugeordnet, bei dem die Cluster-Varianz am wenigsten erhöht wird

$$S_i^t = x_j : \|x_j - m_i^t\|^2 \text{ für alle } i^* = 1, ..., k \tag{7}$$

3. Aktualisierung: Neuberechnung der Mittelpunkte der Cluster:

$$m_i^{t+1} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i} x_j \tag{8}$$

4. Schritte 2-3 werden wiederholt, bis die Zentren $c_1, ..., c_k$ stabil sind, d. h. sich nicht mehr signifikant ändern.

k-Means-Algorithmus
Vor- und Nachteile

Vorteil:

- Leistungsfähigkeit (Komplexität O(kM) pro Iteration)
- Einfache Umsetzung, die lokales Minimum für die Optimierungsaufgabe liefert
- Eines der am häufigsten verwendeten Datensätze

Nachteile:

- Gefundene Lösung hängt stark von der initialen, zufälligen Startpunkten ab
- Ausreißer in den Daten aus $\mathcal{D}$ beeinflussen , aufgrund der Mittelwertbildung für die Bestimmung der Clusterzentren, das finale Ergebnis stark
- Anzahl der Clusterzentren k bereits im Voraus gewählt

Definition der Bounding Box

Ableitung der Objekt Informationen als Bounding Box auf Basis der Ränder der Cluster

KI-Algorithmus
Erläuterung: Grundlagen des Maschinellen Lernens

Im Folgenden wird ein kurzer Einstieg in das Thema Maschinelles Lernen vermittelt. Es besteht kein Anspruch auf Vollständigkeit und so wird z. B. gezielt auf tiefere Mathematik verzichtet. Damit soll ein oberflächliches Verständnis für Begrifflichkeiten, Methoden und damit die spätere Implementierung erzeugt werden.
Maschinelles Lernen basiert auf Mathematik und Statistik, welche für ein tieferes Verständnis nötig ist, aber den Rahmen dieses Projektes übertreffen würde! Für einen tieferen Einstieg empfiehlt sich bspw.:

- Literatur: z.B. [Quelle M. Botsch, W. Utschik]
- Diverse Online-Quellen und Videos
- Studium im entsprechenden Bereich

KI-Algorithmus

Erläuterung: Grundlagen des Maschinellen Lernens

Allgemein: Der Begriff Maschinelles Lernen fasst Methoden der Signalverarbeitung zusammen, die statistische Zusammenhänge in Daten mit Hilfe von Computern finden und damit in der Lage sind, zukünftige Daten vorherzusagen. Man spricht auch von der künstlichen Generierung von Wissen aus Erfahrung. Das maschinelle Lernen beruht auf der mathematischen Statistik und beschäftigt sich mit dem „Lernen“ aus Daten, d. h. mit dem Finden von Gesetzmäßigkeiten in den Daten

Hinweis: Der bereits angewandte k-Means-Algorithmus entspricht damit Maschinellem Lernen. Die Art wird des unüberwachten Lernens (eng. Unsupervised Learning) genannt. Bei dieser lernt Algorithmus allein aus den Eingängen, es gibt keine zugehörigen Ausgangsdaten. Ziel ist es, die Struktur in den Eingangsdaten zu lernen und ein Modell dafür zu erstellen

KI-Algorithmus
Erläuterung: Grundlagen des Maschinellen Lernens

Das maschinelle Lernen ist ein Teilbereich der künstlichen Intelligenz und erhält eine zunehmend wichtige Rolle im Bereich der Ingenieurwissenschaften. Dies ist dadurch begründet, dass die verfügbaren Daten und die Rechenleistung in den letzten Jahren stark zugenommen haben und damit komplexe praktische Anwendungen in unterschiedlichen Ingenieurgebieten mit maschinellen Lernalgorithmen erfolgreich bearbeitet werden können.

Überwachtes Lernen:

- eng. Supervised Learning
- Repräsentiert eine mögliche Art des maschinellen Lernens (Andere: Unsupervised Learning, Semi-Supervised Learning, Reinforcement Learning)
- Ein Algorithmus lernt aus Beispielen von Eingängen und Ausgängen, d. h. es sind Eingangsdaten und zugehörige Ausgangsdaten für das Lernen vorhanden. Man sagt, dass die Eingangsdaten gelabelt sind, d. h. sie haben einen zugehörigen Ausgang. Ziel ist es, Vorhersagen für neue Eingangsdaten zu machen.

KI-Algorithmus
Erläuterung: Grundlagen des Maschinellen Lernens

- **Ziel:** Algorithmus zur Bestimmung einer beschreibenden Größe eines Systems, basierend auf im Vorfeld beobachteten Merkmalen dieses Systems, zu finden. Der Algorithmus soll dann auf Fälle angewandt werden, bei denen die Werte der zu beschreibenden Größe unbekannt sind
- Wenn pro Beobachtung N Merkmale vorliegen, so werden diese im Vektor x zusammengefasst

$$x = [x_1, ..., x_N]^T \tag{9}$$

- Die zu beschreibende Größe wird mit y notiert, die durch den Algorithmus berechnete mit $\hat{y} = f(x)$.

Regression

- Falls y kontinuierliche Werte annimmt, also $y \in \mathbb{R}$, werden die Eingangswerte x auf eine Ausgabe $\hat{y}$ regressiert
- Ziel ist, dass die Lösung y möglichst gut durch $\hat{y}$ approximiert wird.

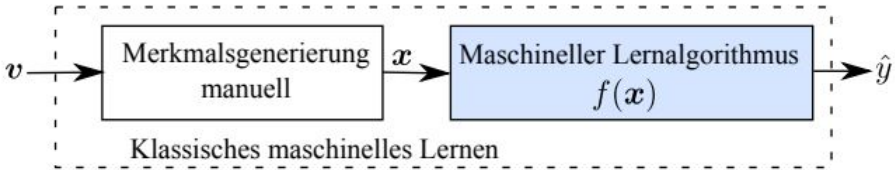


Abbildung: Beispiel für Regression

Training:

- Das Lernen basiert auf einer Verlustfunktion und dessen Gradienten.
- Diese erfasst den Unterschied zwischen dem Ausgang $\hat{y}$ und dem Zielwert y quantitativ.
- Das Ziel des Trainings ist damit, die Funktion f zu finden, bei der dieser Verlust möglichst gering ist.
- Eine häufig verwendete quadratische Verlustfunktion für lautet beispielsweise:

$$\mathcal{L}(y, f(x)) = (y - f(x))^2 \tag{10}$$

Funktion f(x):

- Je komplizierter die Funktion, desto mehr Parameter müssen im Training erlernt werden.
- Gleichzeitig gilt aber auch: Je mehr Parameter erlernt werden können, desto kompliziertere Zusammenhänge können auch dargestellt werden.
- Die Komplexität der Funktion spielt eine entscheidende Rolle für Over- bzw. Underfitting

Over- und Underfitting:

- Beachte: Auch der verwendete Trainingsdatensatz unterliegt der Statistik (Welcher Teil der Realität wird dargestellt?, Messfehler?, Rauschen?, etc.)
- Bei zu hoher Freiheit, kann der Verlust im Training sehr gering werden, obwohl der Algorithmus für neue, unbekannte Daten weniger gut approximieren würde. ist. Grund dafür ist, dass der Algorithmus sich zu stark an den eingeschränkten Trainingsdatensatz anpasst. Man spricht von Overfitting.
- Bei zu geringer Freiheit, kann die Funktion den geforderten Datensatz nicht darstellen. Man spricht von Underfittng.

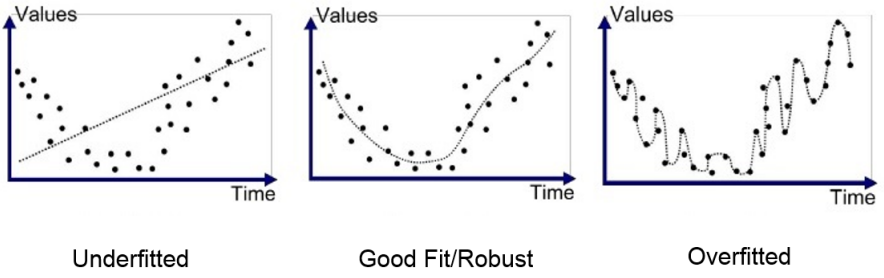


Abbildung: Beispiel für Overfitting und Underfitting [Bhande.2018]

Umgang mit großen Datenmengen:

- Falls M, die Anzahl der Datenpunkte, sehr groß wird, kann es sinnvoll sein den Datensatz im Training in sogenannte Batches aufzuteilen und sich über ein Gradientenabstiegsverfahren sequentiell der Lösung anzunähern.

$$\Theta^{(l+1)} = \theta^{(l)} - \alpha \frac{\delta}{\delta\theta} \Big(\mathcal{L}(y_m, f_\theta(x_m)) \Big) \Big|_{\theta=\theta^{(l)}} \tag{11}$$

  Dabei ist l der Index der Iteration und α die sogenannte Lernrate, die die Schrittweite pro Iteration festlegt.

- Nutzt man für die Aktualisierung der Parameter den gesamten Datensatz $\mathcal{D}$, so nennt man dies eine Epoche.

Normierung

- Bei der Minimierung des quadratischen Fehlers haben Merkmale x_n, deren Wertebereich auch Zahlen mit großem Betrag enthält, einen größeren Einfluss auf die Lösung, als Merkmale, deren Wertebereich nur Zahlen mit kleinem Betrag besteht.
- Um dies zu neutralisieren, werden die Wertebereiche aller Merkmale auf das Intervall zwischen 0 und 1 abgebildet

$$x_{n,m}^{norm} = \frac{x_{n,m} - x_{n,min}}{x_{n,max} - x_{n,min}} \tag{12}$$

Motivation

- Die Berechnung der kritischen Zeit ist auch aus Fahrdynamikgleichungen ableitbar und setzt deswegen keine Notwendigkeit für KI-Algorithmen voraus.
- Die dadurch gegebene Intuition für das Ergebnis ermöglicht aber ein gutes Verständnis für die Möglichkeiten und Grenzen von KI-Algorithmen.
- Ziel ist die kritische Zeit bis zum Auffahrunfall zu berechnen, um z. B. eine Notbremsung zu entwerfen

Ein Maß für die Kritikalität eines Verkehrsszenarios ist die sogenannte **„Time-To-Collision“ (TTC)**

Erläuterung:

- Je kleiner der Wert der TTC, desto kritischer das Szenario
- Für Auffahrunfälle: TTC kann aus der Distanz d der beiden Fahrzeuge und der relativen Geschwindigkeit $v_{x,rel}$ bestimmt werden

$$TTC = \begin{cases} \frac{d}{|v_{x,rel}|} \text{ für } v_{x,rel} < 0 \\ \tau \text{ für } v_{x,rel} \geq 0, \end{cases} \tag{13}$$

wobei τ einen großen Wert besitzt, der indiziert, dass das Szenario unkritisch ist.

1. Generieren Sie einen Datensatz $\mathcal{D}_{train}$ mit den Eingangsvektoren x_m, der die Distanz und relativ Geschwindigkeit zwischen Ego- und vorausfahrendem Fahrzeug enthält. Erzeugen sie zudem das zugehörige Label, die kritische Zeit bis zur Kollision. Damit sind die Zufallsvariablen für Ein- und Ausgang:

$$x = \begin{bmatrix} d & v_{x,rel} \end{bmatrix}^T \in \mathbb{R} \text{ und } y = TTC \in \mathbb{R} \tag{14}$$

Vorgaben bei der Erzeugung des Trainingsdatensatzes $\mathcal{D}_{train}$:

- Distanz d: Schrittweite $1m$; Intervall $[1m, 30m]$
- Relativgeschwindigkeit $v_{x,rel}$: Schrittweite $5km/h$; Intervall $[-60km/h, -5km/h]$
- $\tau = 60s$

KI-Algorithmus
Implementierung: lineare Regression

2. Erzeugen Sie ein Testdatenset $\mathcal{D}_{text}$ mit folgenden Vorgaben:
   - Distanz d: Schrittweite $0.5m$; Intervall $[1m, 40m]$
   - Relativgeschwindigkeit $v_{x,rel}$: Schrittweite $3km/h$; Intervall $[-70km/h, -7km/h]$
3. Wenden sie als Methode eine lineare Regression an. Bestimmen Sie den Parameter θ mit Hilfe des Trainingsdatensatzes $\mathcal{D}_{train}$. Vergessen Sie dabei die Normierung des Eingangsvektors nicht.
4. Testen sie den trainierten Algorithmus mit dem Testdatensatz $\mathcal{D}_{test}$, indem Sie die Ausgaben $\hat{y}$ mit den korrekten Ergebnissen $y = TTC$ vergleichen
5. Visualisieren Sie das das Mapping der linearen Regressios zwischen d, $v_{x,rel}$ und $\hat{y}$. Visualisieren Sie im selben Plot auch den Datensatz $\mathcal{D}_{test}$
6. Integration in die ROS-Pipeline zur Bestimmung der kritischen Zeit TTC in Echtzeit

Die lineare Regression ist eine einfache Methode des maschinellen Lernens, die eine gute Veranschaulichung ermöglicht.

- Es wird angenommen, dass die Datenquelle die Zielwerte y entsprechend der Rechenvorschrift

$$y = w^T x + t + \eta \text{ mit } \eta \backsim \mathcal{N}(0, \sigma^2), \tag{15}$$

  wobei w und t Parameter sind, und x die Dimension N hat, generiert.
- Für eine kompaktere Schreibweise lassen sich w und t in den Parametervektor $\Theta = [w^T, t]^T$ der Dimension $(N + 1)$ zusammenfassen und der Eingang x erweitern zu

$$\tilde{x} = [x^T, 1]^T, \tag{16}$$

- Damit gilt:

$$y = \Theta^T \tilde{x} + \eta \text{ mit } \eta \backsim \mathcal{N}(0, \sigma^2), \tag{17}$$

- Die Lösung für den gesuchten Parameter θ ergibt sich aus dem Minimum der Verlustfunktion. Für die quadratische Verlustfunktion bedeutet dies:

$$\theta = \underset{\theta}{\operatorname{argmin}} \left\{ \frac{1}{M} \sum_{m=1}^{M} (y_m - \theta^T \tilde{x}_m)^2 \right\} \tag{18}$$

- Die Ableitung der quadratischen Funktion hat eine Nullstelle im Minimum

$$\frac{\delta}{\delta \theta^T} \Big(\frac{1}{M} \sum_{m=1}^{M} (y_m - \theta^T \tilde{x}_m)^2 \Big) = \sum_{m=1}^{M} -(y_m - \theta^T \tilde{x}_m) \tilde{x}_m^T \stackrel{!}{=} 0 \tag{19}$$

Die Lösung ist:

$$\theta = \Big(\sum_{m=1}^{M} \tilde{x}_m \tilde{x}_m^T\Big)^{-1} \Big(\sum_{m=1}^{M} \tilde{x}_m y_m\Big) \tag{20}$$

Schreibt man dieses Ergebnis in Matrix-Vektor-Notation, so werden alle Eingänge und Zielwerte aus dem Datensatz $\mathcal{D}$ in der Matrix $\tilde{X}$ und dem Vektor y sortiert.

$$\theta = (\tilde{X}^T \tilde{X})^{-1} \tilde{X}^T y \quad \text{, mit } \tilde{X} = \begin{bmatrix} \tilde{x}_1^T \\ \tilde{x}_2^T \\ \vdots \\ \tilde{x}_M^T \end{bmatrix}, y = \begin{bmatrix} \tilde{y}_1 \\ \tilde{y}_2 \\ \vdots \\ \tilde{y}_M \end{bmatrix} \tag{21}$$

Motivation

- Die Einschränkung durch Linearität ermöglicht es dem Algorithmus nicht die TTC fehlerfrei darzustellen.
- Ein klassischer Fall von Underfitting.
- Eine entscheidende Möglichkeit kompliziertere Zusammenhänge darzustellen, ist Nicht-Linearität in die Funktion zu bringen
- Eine Möglichkeit sind nicht-lineare Basisfunktionen.

1. Verwenden Sie die Datensätze der vorhergehenden Aufgabe.
2. Analog zum beschriebenen Verfahren x soll eine nichtlineare Basisfunktion $\varphi(x)$ verwendet werden. Folgende wird in diesem Fall gegeben:

$$\varphi(x) = \begin{bmatrix} \varphi_1(x) & \varphi_2(x) & \varphi_3(x) & \varphi_4(x) & \varphi_5(x) & \varphi_6(x) & \varphi_7(x) & \varphi_8(x) & \varphi_9(x) \end{bmatrix}$$

mit

$$\varphi_1(x) = x_1, \;\; \varphi_2(x) = x_2, \;\; \varphi_3(x) = x_1^2, \;\; \varphi_4(x) = x_2^2, \;\; \varphi_5(x) = x_1 x_2,$$
$$\varphi_6(x) = x_1^3, \;\; \varphi_7(x) = x_2^3, \;\; \varphi_8(x) = x_1^2 x_2, \;\; \varphi_9(x) = x_1 x_2^2$$

KI-Algorithmus

Implementierung: Nicht-lineare Basisfunktionen

3. Bestimmung von θ für das neue Regressionsmodell. Test der Ausgabe $\hat{y}$ für den Testdatensatz $\mathcal{D}_{test}$.
4. Visualisierung des Zusammenhangs von d und v_x, rel zu $\hat{y}$ für das berechnete Regressionsmodell. Visualisierung des Datensatzes $\mathcal{D}_{test}$ im selben Plot.
5. Integration in die ROS-Pipeline zur Bestimmung der kritischen Zeit TTC in Echtzeit
6. Zusatzaufgabe: Untersuchen sie die Folgen einer Änderung des Trainingsdatensatzes oder der Basisfunktion

- Basiert auf der linearen Regression, wobei x durch $\phi(x)$ ersetzt wird.

$$y = w^T \varphi(x) + t + \eta \text{ mit } \eta \backsim \mathcal{N}(0, \sigma^2) \text{ und } \varphi(x) = [\varphi_1(x), \varphi_2(x), \ldots, \varphi_H(x)]^T, \quad (22)$$

wobei die Funktionen $\varphi(x)$ nichtlinear sein können

- Analog zur vorherigen Herleitung ergibt sich:

$$\tilde{\varphi}(x_m) = \begin{bmatrix} \varphi(x_m)^T & 1 \end{bmatrix}^T \quad (23)$$

$$\theta = \Big(\sum_{m=1}^{M} \tilde{\varphi}(x_m) \tilde{\varphi}(x_m)^T \Big)^{-1} \Big(\sum_{m=1}^{M} \tilde{\varphi}(x_m) y_m \Big) \quad (24)$$

- In Matrixschreibweise ergibt sich:

$$\theta = \left(\tilde{\phi}^T \tilde{\phi}\right)^{-1} \tilde{\phi}^T y \tag{25}$$

mit

$$\tilde{\phi} = \begin{bmatrix} \tilde{\varphi}(x_1)^T \\ \tilde{\varphi}(x_2)^T \\ \vdots \\ \tilde{\varphi}(x_M)^T \end{bmatrix}, \; y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{bmatrix} \tag{26}$$

Quellen

Der Aufbau und Inhalt der Folien ist hauptsächlich angelehnt an [Botsch.2020]. Weitere genutzte Quellen sind:

- 1) [Wik.2022]
  Wikipedia. Lidar. https://de.wikipedia.org/wiki/Lidar. zuletzt abgerufen am 13.05.2022
- 2) [Rummelhard.2017]
  Lukas Rummelhard, Anshul Paigwar, Amaury Nègre, Christian Laugier. Ground Estimation
- 3) [GeneSys.2022]
  ADMA Automotive Dynamic Motion Analyzer with 1000 Hz. https://www.leaneautomotive.co
- 4) [Miller.2022]
  https://www.wired.com/2014/12/nokia-here-autonomous-car-maps/. Autonomous Cars Will

- 5) [Breuer.2015]

  Stefan Breuer, Andrea Rohrbach-Kerl. Fahrzeugdynamik.Mechanik des bewegten Fahrzeug

- 6) [Hyin.2018]

  William Hyin. Lidar Obstacle Detection. https://www.codetd.com/en/article/10631576. 2018.

- 7) [Botsch.2020]
  M. Botsch, W. Utschick. Fahrzeugsicherheit und automatisiertes Fahren. Methoden des Sig
- 8) [Caesar.2020] Caesar et al.. nuScenes: A multimodal dataset for autonomous driving. 2020. arXiv:1903.11027v5
- 9) [Velodyne.2020] Landis Communications Inc.. Velodyne-Lidar-Sensoren ermöglichen 3D-Datenerfassung im neuen NavVis VLX-Kartierungssystem. 2020. zuletzt abgerufen am 13.05.2022
- 10) [Bhande.2018] Anup Bhande. What is underfitting and overfitting in machine learning and how to deal with it. https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76. 2018. zuletzt besucht am 13.05.2022