Masterarbeit

Computer Science

Sensorbasierter Orientierungssinn mit künstlichen neuronalen Netzen und Entscheidungsbäumen

von

Tom Dymel

Juli 2021

Erstprüfer | Prof. Dr. Volker Turau
Institute of Telematics
Hamburg University of Technology

Zweitprüfer | Dr. Marcus Venzke
Institute of Telematics
Hamburg University of Technology

Institut für Telematik
Technische Universität Hamburg
TUHH

Inhaltsverzeichnis

Kapitel 1

Einleitung

Lokalisation is der Prozess die Position von einem Objekt zu bestimmen. Die Positionsbestimmung ist integral für viele technische System, z. B. Tracking, Navigation oder Überwachung. Ein sehr bekanntes und vielseitig genutztes Positionsbestimmungssystem ist das Global Positioning System (GPS). GPS trianguliert die Position des anfragenden Geräts mit Hilfe von mehreren Satelliten (TODO Quelle). Im freien ist eine Genauigkeit von X(TODO) m möglich. Für die Positionsbestimmung innerhalb von Gebäuden ist die Genauigkeit aber nicht ausreichend, außerdem wird sie erschwert durch die dicke Wände und Interferenzen. Dadurch ist GPS oft nicht ausreichend für Trackingsysteme innerhalb von Gebäuden, z. B. Lagerhallen. Aus diesem Grund werden andere Systeme für diesen Zweck verwendet. Je nach Bedarf der Genauigkeit können Objekte mit Sendern, RFID Tags oder Barcodes markiert werden (TODO Quellen). Diese Ansätze bedürfen eine Infrastruktur, die in den Gebäuden installiert und gewartet werden muss.

In dieser Arbeit wird die diskrete Positionsbestimmung basierend auf Sensordaten untersucht. Dabei soll eine bestimmte Anzahl an Orten anhand von verschiedenen Sensorwerten unterschieden werden. Dies ist Vergleichbar mit dem Orientierungssinn von Tieren und Menschen. Zum Beispiel navigieren Honigbienen auf Basis von gelernten Orientierungspunkten, um Nahrungsquelle und Nest zu finden (TODO Quelle). Mit Hilfe maschinellen Lernens sollen künstliche neuronale Netze (KNN) und Entscheidungsbäume trainiert werden. Als Eingabedaten werden aus den gesammelten Sensordaten Features extrahiert, d. h. Attribute und Eigenschaften dieser Daten. Dieses System bedarf keine Infrastruktur muss aber für jedes Gebäude individuell trainiert werden.

TODO: Was wurde in dieser Arbeit gemacht.

TODO: Kapitelübersicht

Kapitel 2

Entscheidungsbäume

Ein Entscheidungsbaum ist ein Baum mit dem Entscheidungen getroffen werden [Qui90]. Das geschieht, indem der Baum von der Wurzel zu einem Blatt traversiert wird. Dabei bestimmt ein Test in jedem inneren Knoten, mit welchem Kindknoten fortgefahren wird. Jedes Blatt entspricht einer Entscheidung des Entscheidungsbaums. Es wird unterschieden zwischen Bäumen, die versuchen eine der vordefinierten Klassen zu klassifizieren (Klassifizierer), und solchen, die versuchen den nächsten Wert vorherzusagen (Regressoren).

Die Konstruktion eines optimalen binären Entscheidungsbaumes ist NP-Vollständig [LR76]. Den optimalen Klassifizierer zu finden ist folglich sehr aufwendig. Aus diesem Grund werden bei der Konstruktion Heuristiken verwendet, die nur lokal die beste Entscheidung treffen. Zudem werden einzelne Entscheidungsbäume in einem Ensemble zu einen Entscheidungswald zusammengefasst, um den Fehler eines einzelnen Baumes zu reduzieren [Ent20b].

2.1 Scikit-Learn

Diese Arbeit verwendet die Python ML-Bibliothek *Scikit-Learn*. Scikit-Learn bietet verschiedene ML Algorithmen an mit einem High-Level Interface [PVG+11]. Zur Konstruktion der Entscheidungsbäume wird der Algorithmus von CART verwendet [Ent20a]. Die Bibliothek kann Klassifizierer und Regressoren generieren.

Relevant ist jedoch lediglich der Klassifizierer, da in dieser Arbeit ein Klassifizierungsproblem gelöst werden soll. Dieser bietet zahlreiche Hyperparameter an, um die Konstruktion des Entscheidungsbaumes zu steuern. In dieser Arbeit wird lediglich der Hyperparameter `max_depth` verwendet. Dieser Hyperparameter beschränkt die maximale Baumhöhe und begrenzt somit den Programmspeicherverbrauch.

Scikit-Learn bietet Ensemble-Methoden an, um Entscheidungswälder zu trainieren. Ein wichtiger Parameter diese Methoden ist `n_estimators`. Dieser steuert die Größe des Ensembles bzw. Waldgröße. Somit hat auch dieser Parameter Einfluss auf den Programmspeicherverbrauch.

2.2 Einzelne Entscheidungsbäume

Der einzelne Entscheidungsbaum ist eine rekursive Datenstruktur, um Entscheidungsregeln darzustellen [Qui90]. Jeder innere Knoten ist ein *Test*, welcher eine arbiträre Anzahl von sich gegenseitig ausschließenden Ergebnissen hat. Das Ergebnis eines Tests bestimmt mit welchem Kindknoten fortgefahren wird. Die Blätter des Baumes stellen die Entscheidungen dar bzw. die Klassen des Entscheidungsbaumklassifizierers. Abbildung 2.1 zeigt einen binäreren Entscheidungsbaum, in dem jeder Test zwei mögliche Ergebnisse hat. Das Trainieren von

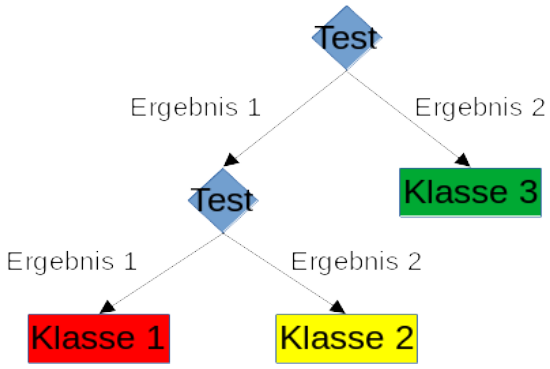

Abbildung 2.1: Beispiel eines binären Entscheidungsbaums mit 3 möglichen Ergebnissen.

Entscheidungsbäumen ist eine Art von *Supervised Learning*, d. h. aus einer beschrifteten Trainingsmenge werden Regeln abgeleitet, um das korrekte Mapping von Input zu Output abzubilden [GL09]. Die Trainingsmenge besteht aus Feature-Mengen, die mit Klassen beschriftet sind [Ste09]. Die Generalisierungsfähigkeit ist abhängig von der Trainingsmenge. Zum einen, sollte die Trainingsmenge möglichst repräsentativ sein für die Aufgabe, die gelernt werden soll. Zum anderen, sollten die verwendeten Features eine Partitionierung aller Klassen ermöglichen [PGP98].

Entscheidungsbäume werden heuristisch konstruiert, da die Konstruktion eines optimalen Entscheidungsbaumes NP-Vollständig ist [LR76]. Zu diesen Algorithmen gehören zum Beispiel `ID3` [Qui86], `C4.5` [Qui14] oder `CART` [BFSO84]. Die Aufgabe ist durch gezielte Trennungen eine Partitionierung der Trainingsmenge zu erzeugen, sodass möglichst nur Einträge mit der gleichen Beschriftung in einer Partitionierung enthalten sind. Die Algorithmen unterscheiden sich in ihrer Strategie [Qui86].

Scikit-Learn implementiert eine optimierte Version des *CART* (**C**lassification **A**nd **R**egression **T**rees) Algorihmus [Ent20a]. CART partitioniert die Trainingsmenge indem lokale immer die beste Teilung ausgewählt wird, d. h. es wird immer lokal die Beste Teilungsregel ausgewählt. Dieser Vorgang wird rekursiv mit jeder Teilmenge wiederholt, bis keine weitere Teilung mehr möglich ist oder alle Einträge einer Partitionierung die gleiche Beschriftung tragen [Ste09].

2.3 Ensemble-Methoden

TODO

2.4 Cherry-Picking

- Gehe hier auf Monte Carlo etc ein.

2.5 Ressourcenbedarf auf dem Mikrocontroller

- Energieverbrauch
- Speicherverbrauch
- RAM verbrauch
- Siehe Forschungsprojekt

Kapitel 3

Künstliche Neuronale Netze

TODO: Schau mal bei Kubik und Co. rein, was die da so geschrieben haben

3.1 Keras

TODO

3.1.1 Aktivierungsfunktionen

TODO

3.1.2 Verlustfunktionen

TODO

3.1.3 Optimierer

TODO

3.1.4 Regularisierung

TODO

3.2 Feed Forward Neuronale Netze

TODO

3.3 Training von Neuronalen Netzen

TODO

3.4 Ressourcenbedarf auf dem Mirkocontroller

- Energieverbrauch
- Speicherverbrauch
- RAM verbrauch
- Siehe Kubik und Co.

Kapitel 4

Standortbestimmung

- Location Awareness
- Anwendungen?

4.1 Lokalisierung

- Indoor
- Outdoor
- Network vs. Device based

4.2 Anwendung von Lokalisation

- GPS
- Triangulierung
- Navigation, Monitoring, Tracking

4.3 Standortbestimmung mit maschinellen Lernen

- Vision Based

4.4 Orientierungssinn

- Beispiel Honigbienen

Kapitel 5

Trainings- und Validationsdaten

- Daten aufgenommen mit CoppeliaSim(?)

5.1 Simulierten Sensordaten

- Wie wurde es aufgenommen?
- Welche Sensoren
- Aufgenommene Routen

5.2 Künstlichen Sensordaten

- Motivation: Warum ist das nötig?

5.2.1 Magnetfeld

- Welchen Sensor spiegelt das wieder?
- Wie funktioniert das Modell?
- Was und Wie wurden Daten ergänzt?

5.2.2 Temperatur

- Welchen Sensor spiegelt das wieder?
- Wie funktioniert das Modell?
- Was und Wie wurden Daten ergänzt?

5.2.3 Lautstärke

- Welchen Sensor spiegelt das wieder?
- Wie funktioniert das Modell?
- Was und Wie wurden Daten ergänzt?

5.2.4 WLAN Zugangspunkte

- Welchen Sensor spiegelt das wieder?
- Wie funktioniert das Modell?
- Was und Wie wurden Daten ergänzt?

5.3 Simulation von Interrupts

- Motivation => Energieverbrauch, Spiegelung der echten Datenaufnahme, Reduzierung der Trainingsdaten
- Wie funktioniert ist?
- Wie und Wann bei der Datenverarbeitung wird es gemacht?

5.4 Standortenkodierung

- Wie werden Orte modelliert?
- Warum wurden die Orte so modelliert?
- Wie werden Pfade zwischen Orten modelliert?
- Sollte das Modell ausgeben können, dass kein Ort erkannt wurde?
- Sollten Pfade erkannt werden können?

5.5 Feature-Extrahierung

- Datenfenster: Realzeit vs. Diskret mittels Wakeups
- Relevanz von Zeit
- Welche Feature werden genutzt?

- Wie werden diese extrahiert?
- Welchen Mehrwert verschaffen diese Features?
- Welchen Einfluss haben Sie im Hinblick auf Ressourcenbedarf(?), Klassifizierungsgenauigkeit(?), Fehlertoleranz(?)

5.6 Aufteilung der Daten

- Kurz und knapp wie und warum werden die Daten aufgeteilt.
- Sollten Trainingsdaten um synthetische Daten ergänzt werden?
  - Fault Daten, um das Modell Robuster zu machen
  - Synthetische Routen
- Wie viele Trainingsdaten werden benötigt?
  - Um KNN zu trainieren?
  - Um Entscheidungsbaum zu trainieren?
  - Ggf. Unterschiede klären

Kapitel 6

ML-Modelle

TODO

6.1 Entscheidungswald

- Wie wird automatisch das beste Modell gefunden?
- Warum trainieren wir es so => Um die beste Konfiguration zu finden unter den bestehenden Restriktionen.
- Welche Restriktionen werden eingefordert?
- Warum gibt es diese Restriktionen?
- Welchen Einfluss hätte es, gäbe es diese nicht?

6.2 Feed Forward Neuronales Netzwerk

- Wie viele Hidden Layers und Neuronen sollte es haben?
- Welche Aktivierungsfunktionen werden verwendet?
- Braucht man mehr Neuronen/Hidden Layer mit steigender Ort Anzahl?

6.3 Feedback Kanten

- Was ist das?
- Wofür ist das Relevant?
- Wie funktionier es?

6.4 Training der Modelle

- Trainieren in Phasen => Erklären wie es genau funktioniert?
- Warum trainieren wir in Phasen?
- Sag das wir die Zyklen der einzelnen Pfade nutzen
- Werden mehr Trainingsdaten benötigt mit steigender Ort Anzahl? Wenn ja wie viel?

Kapitel 7

Evaluation

TODO

7.1 Klassifizierungsgenauigkeit

- Metriken
- Wie groß ist die Wahrscheinlichkeit, dass die Orte korrekt erkannt werden?
- Entscheidungsbaum vs KNN
- Skalierung mit Anzahl der Orte
- Signifikanz der Features
- Einfluss von einzelnen Features für die Klassifizierungsgenauigkeit(?)
- Ist es sinnvoll für jeden Ort ein Feature zu haben, dass auf eins gesetzt wird, wenn der Ort erkannt wurde und ansonsten exponentiell abfällt. Wie schnell sollte es fallen, wenn ja?

7.2 Fehlertoleranz

- Metriken => Insbesondere continued_predict hier erwähnen
- Wenn falscher Ort erkannt wurde, wie lange dauert es um wieder den korrekten Ort zu finden?
- Was passiert wenn Sensoren ausfallen?
- Transportbox nicht dem trainierten Pfad folgt?

- Änderungen der Fabrik, e.g. Licht, Wärme, Magnet, Schnelligkeit der Fließbänder
- Einfluss von einzelnen Features für die Fehlertoleranz(?)

7.3 Ressourcennutzung

- Metriken(?)
- Entscheidungsbaum Ausführung
- FFNN Ausführung
- Feature extrahierung
  - Verhältnis von Kosten zu Nutzen
- Daten Sammlung => Interrupts (Wakeups)
- Einfluss von einzelnen Features für den Ressourcenverbrauch(?)

Kapitel 8

Diskussion

TODO

Kapitel 9

Schlussfolgerungen

TODO

- Kurze, knappe und gut formulierte Schlussfolgerung
  - Was ist besser Entscheidungsbaum oder Entscheidungswald? Welche Vor- und Nachteile?
  - Sollten Entscheidungswälder verwendet werden?
  - Sollten KNNs verwendet werden?
  - Is die Realzeit relevant für die Ortung(?)
- Kurze Zusammenfassung wichtigster Dinge
- Zukünftige Arbeit

Inhalt des USB-Sticks

Literaturverzeichnis

[BFSO84] BREIMAN, Leo ; FRIEDMAN, Jerome ; STONE, Charles J. ; OLSHEN, Richard A.: *Classification and regression trees*. CRC press, 1984

[Ent20a] ENTWICKLER scikit-learn: *1.10.6. Tree algorithms: ID3, C4.5, C5.0 and CART*. `https://scikit-learn.org/stable/modules/tree.html#tree-algorithms`. Version: 2020

[Ent20b] ENTWICKLER scikit-learn: *1.11. Ensemble methods*. `https://scikit-learn.org/stable/modules/ensemble.html#ensemble`. Version: 2020

[GL09] GHOSH, Joydeep ; LIU, Alexander: K-Means. In: *The top ten algorithms in data mining* 9 (2009), S. 21–22

[LR76] LAURENT, Hyafil ; RIVEST, Ronald L.: Constructing optimal binary decision trees is NP-complete. In: *Information processing letters* 5 (1976), Nr. 1, S. 15–17

[PGP98] PEI, M ; GOODMAN, ED ; PUNCH, WF: Feature extraction using genetic algorithms. In: *Proceedings of the 1st International Symposium on Intelligent Data Engineering and Learning, IDEAL* Bd. 98, 1998, S. 371–384

[PVG+11] PEDREGOSA, F. ; VAROQUAUX, G. ; GRAMFORT, A. ; MICHEL, V. ; THIRION, B. ; GRISEL, O. ; BLONDEL, M. ; PRETTENHOFER, P. ; WEISS, R. ; DUBOURG, V. ; VANDERPLAS, J. ; PASSOS, A. ; COURNAPEAU, D. ; BRUCHER, M. ; PERROT, M. ; DUCHESNAY, E.: Scikit-learn: Machine Learning in Python. In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830

[Qui86] QUINLAN, J. R.: Induction of decision trees. In: *Machine learning* 1 (1986), Nr. 1, S. 81–106

[Qui90] QUINLAN, J R.: Decision trees and decision-making. In: *IEEE Transactions on Systems, Man, and Cybernetics* 20 (1990), Nr. 2, S. 339–346

[Qui14] QUINLAN, J R.: *C4. 5: programs for machine learning*. Elsevier, 2014

[Ste09] STEINBERG, Dan: CART: classification and regression trees. In: *The top ten algorithms in data mining* 9 (2009), S. 179–201