Hausarbeit SME-PHY-B: Wahlthema 2 - Fitnesszähler

Joel Ewig

28. März 2021

Zusammenfassung

Es sollen sensorbasiert ähnlich einer Fitnessuhr verschiedene Sportübungen sowie deren Anzahl erkannt werden. Benutzt werden sollen der Beschleunigungssensor und das Gyroskop des IMU-6050.
Mein Lösungsansatz gliedert sich in mehrere Stufen: die Rohdaten aus dem IMU-6050 sollten durch einen mehrdimensionalen Kalman-Filter bereinigt werden. Die bereinigten Daten werden mit dem K-Means Algorithmus geclustert um daraus sogenannte „Beobachtungen“ zu machen. In der Lernphase wird mit diesen Beobachtungen pro Übung ein Hidden Markov Model erlernt. In der folgenden Detektionsphase werden die empfangenen Daten ebenfalls geclustert und die Beobachtungen an alle Hidden Markov Modelle weitergegeben. Das Hidden Markov Modell, welches die höchste Wahrscheinlichkeit für die Emittierung der beobachteten Folge bestimmt, gilt als Erkenner und die zugehörige Übung wird als ausgeführt behandelt.

Inhaltsverzeichnis

1 Konzept **4**
1.1 Lernphase 4
1.2 Detektionsphase 5
1.3 Beispiel 5

2 Sensorik **7**

3 Mehrdimensionaler Kalman-Filter **7**

4 Baum-Welch Algorithmus **8**

5 Evaluierung **10**

Bibliographie **14**

1 Konzept

Hier soll die Idee, nach der vorgegangen wird, erläutert werden. Genauere Angaben zur Sensorik sind in Kapitel 2. Die Umsetzung eines Kalman-Filters und warum dieser hier nicht umgesetzt wird, sind in Kapitel 3 zu finden. Eine Erklärung zur Funktionsweise des Baum-Welch Algorithmus findet sich in Kapitel 4. Abschließend wird eine Evaluierung auf Grundlage von Experimenten in Kapitel 5 gezeigt.
Der hier betrachtete Ansatz nutzt in der Praxis zwei Phasen. In der ersten Phase, der Lernphase, wird die Anzahl der zu erkennenden Übungen sowie für jede Übung eine Anzahl an Ausführungen festgelegt. Nachdem diese durchgeführt wurden und fertig verarbeitet wurden, ist die Lernphase beendet. In der Detektionsphase wird eine Anzahl an zu erkennenden Ausführungen festgelegt. Nachdem diese durchgeführt wurden, wird das Ergebnis der Detektion ausgegeben und die Detektionsphase ist beendet.

1.1 Lernphase

Die einkommenden 6-dimensionalen Daten werden wie oben beschrieben mit einem K-Means geclustert. Der K-Means ([Skl, 2021]) wird auf den kompletten Trainingsdaten initialisiert, also unabhängig von den ausgeführten Übungen. Jeder Punkt bekommt dadurch einen Clusternamen zugewiesen, umgesetzt als Nummer, welche weitergegeben werden. Die Clusternummern sind die diskreten Beobachtungen die als Eingabe für das Lernen der Hidden Markov Modelle dienen.
Genutzt wird ein Hidden Markov Modell (π, p, q) nach [Rabiner, 2001] mit:

1. $\pi : Z \to [0, 1]$ für die initialen Wahrscheinlichkeiten der Zustände in Z
2. $p : Z \times Z \to [0, 1]$ für die Übergangswahrscheinlichkeiten zwischen den Zuständen in Z
3. $q : Z \times B \to [0, 1]$ für die Beobachtungswahrscheinlichkeiten für Beobachtung B in den Zuständen in Z

Ein Hidden Markov Modell für eine Übung wird wie folgt trainiert:
Zuerst werden die Emissionswahrscheinlichkeiten und die Übergänge zwischen den versteckten Zuständen zufällig initialisiert, konkret wird also π mit Zufallszahlen im Intervall $[0, 1]$ gefüllt. Anschließend werden die genannten Werte mittels des Baum-Welch Algorithmus an die in der Lernphase aufgenommenen Sequenzen angepasst (mehr dazu in Kapitel 4). Die Hidden Markov Modelle werden mehrfach initialisiert, um die Wahrscheinlichkeit nicht über ein lokales Optimum hinauszukommen zu vermindern. Es wird das Modell gewählt, welches die größte Wahrscheinlichkeit bei der Erkennung der bekannten Übungen aufweist. Durch die geringe Anzahl an Lerndaten wird hier keine Mindestwahrscheinlichkeit gesetzt, da dazu zu wenig Trainingsdaten vorhanden sind. In den durchgeführten Experimenten hat sich kein Wert als stabil erwiesen, da hier

versucht wurde mit möglichst wenigen Ausführungen auszukommen. Dadurch bleibt der Trainingsdatenumfang relativ gering (mehr dazu in Kapitel 5).

1.2 Detektionsphase

In der Detektionsphase werden die empfangenen Daten mit dem K-Means aus der Lernphase geclustert. Der K-Means ist eingefroren, das heißt die Clustermittelpunkte bleiben so wie sie in der Lernphase bestimmt wurden.
Die Beobachtungen einer Durchführung werden allen Hidden Markov Modelle zugeführt, diese geben die Emissionswahrscheinlichkeit für die beobachtete Sequenz aus. Es wird die Übung erkannt, dessen zugehöriges Hidden Markov Modell die höchste Wahrscheinlichkeit für die Emission der beobachteten Sequenz ausgibt.

1.3 Beispiel

Es folgt ein Beispiel für eine typische Ausführung in der Detektionsphase:
Während der Ausführung einer Übung werden die Daten des Gyroskops und des Beschleunigungssensors erfasst. Für 5 Zeitschritte haben die Gyroskopdaten folgende Form:

$$\begin{bmatrix} g_1^1 & g_1^2 & g_1^3 & g_1^4 & g_1^5 \\ g_2^1 & g_2^2 & g_2^3 & g_2^4 & g_2^5 \\ g_3^1 & g_3^2 & g_3^3 & g_3^4 & g_3^5 \end{bmatrix}$$

und die des Beschleunigungssensors folgende Form:

$$\begin{bmatrix} a_1^1 & a_1^2 & a_1^3 & a_1^4 & a_1^5 \\ a_2^1 & a_2^2 & a_2^3 & a_2^4 & a_2^5 \\ a_3^1 & a_3^2 & a_3^3 & a_3^4 & a_3^5 \end{bmatrix}$$

Diese werden mit einem mehrdimensionalen Kalman-Filter (Kapitel 3) bereinigt und wie folgt zusammengeführt:

$$\begin{bmatrix} a_1^1 & a_1^2 & a_1^3 & a_1^4 & a_1^5 \\ a_2^1 & a_2^2 & a_2^3 & a_2^4 & a_2^5 \\ a_3^1 & a_3^2 & a_3^3 & a_3^4 & a_3^5 \\ g_1^1 & g_1^2 & g_1^3 & g_1^4 & g_1^5 \\ g_2^1 & g_2^2 & g_2^3 & g_2^4 & g_2^5 \\ g_3^1 & g_3^2 & g_3^3 & g_3^4 & g_3^5 \end{bmatrix}$$

Die Spaltenvektoren dienen als Eingabe für den K-Means Clustering Algorithmus. Dieser gibt für jeden Spaltenvektor das zugehörige Cluster c aus:

$$\begin{bmatrix} a_1^1 \\ a_2^1 \\ a_3^1 \\ g_1^1 \\ g_2^1 \\ g_3^1 \end{bmatrix} \rightarrow c_1$$

Die Sequenz der Clusternamen dient als Eingabe in n Hidden Markov Modelle, n ist dabei die Anzahl der Übungen, die erkannt werden sollen. Die Hidden Markov Modelle geben eine Emissionswahrscheinlichkeit für die beobachtete Folge an:

$$[c_1, c_2, c_3, c_4, c_5] \rightarrow p_1, ..., p_n$$

Aus den Emissionswahrscheinlichkeiten p wird die größte gewählt. Die zum gewählten Hidden Markov Modell gehörende Übung gilt als erkannt.

2 Sensorik

Auf dem Arduino werden die Daten des Beschleunigungssensors und des Gyroskop aus dem IMU ausgelesen. Das Gyroskop sowie der Beschleunigungssensor werden dem im Datenblatt beschriebenem Selbsttest unterzogen. Dieser soll feststellen, ob die Sensoren noch funktionsfähig sind. Die zugehörigen Daten werden über die serielle Schnittstelle an ein Pythonskript übermittelt, welches die zugehörigen Rechnungen vornimmt und bei Nichtbestehen des Selbsttests das Programm beendet.

3 Mehrdimensionaler Kalman-Filter

Hier wird aus Zeitgründen auf einen Kalman-Filter verzichtet, da dieser die Ergebnisse zwar verbessern würde, allerdings nicht unbedingt notwendig ist. Mit einem solchen Kalman-Filter wäre die Erkennung vermutlich robuster, da mit bereinigten Daten „engere“ Cluster zu erwarten wären. Ein solcher Filter würde wie folgt aussehen:

$$X^{[t+1]} = A^{[t]}X^{[t]} + B^{[t]}u^{[t]} + C^{[t]} + \epsilon^{[t]}$$

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ g_1 \\ g_2 \\ g_3 \end{bmatrix}^{[t+1]} = A^{[t]} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ g_1 \\ g_2 \\ g_3 \end{bmatrix}^{[t]} + B^{[t]} \begin{bmatrix} ra_1 \\ ra_2 \\ ra_3 \\ rg_1 \\ rg_2 \\ rg_3 \end{bmatrix}^{[t]} + C^{[t]} + \epsilon^{[t]}$$

Dabei müssen $A^{[t]}$, $B^{[t]}$ und $C^{[t]}$ gesetzt und experimentell überprüft werden. Hier modelliert $\epsilon^{[t]}$ das Messrauschen und ist damit bei der Implementierung nicht zu beachten. Es darf allerdings nicht vergessen werden, dass der Kalman-Filter nur ein Zustands**schätzer** ist.

4 Baum-Welch Algorithmus

Der Baum-Welch Algorithmus wendet den Maximum-Likelihood Algorithmus auf Hidden Markov Modelle an. Genutzt wird ein Hidden Markov Modell (π, p, q) mit:

1. $\pi : Z \to [0, 1]$ als inititale Wahrscheinlichkeiten für die Zustände Z
2. $p : Z \times Z \to [0, 1]$ als Übergangswahrscheinlichkeiten zwischen den Zuständen Z
3. $q : Z \times B \to [0, 1]$ als Beobachtungswahrscheinlichkeiten für die Beobachtung B in den Zuständen Z

In der hier genutzten Implementierung ([Hmm, 2021]) sind p und q als Matrizen umgesetzt. Diese werden zu Beginn mit zufälligen Werten initialisiert. Im ersten Schritt werden die Wahrscheinlichkeiten mit den aktuellen Parametern ermittelt. Die Idee hinter dem Baum-Welch Algorithmus ist es, die Übergänge, die der wahrscheinlichste Pfad durch das Hidden Markov Modell nutzt, wahrscheinlicher zu machen.
Die nachfolgende Passage orientiert sich an der Erklärung in [Wunsch, 2001]. Für eine beobachtete Sequenz $O = o_1, .., o_T$ heißt das, dass π in der nächsten Iteration zu $\tilde{\pi}$ wird und berechnet sich wie folgt:

$$\tilde{\pi_i} = \gamma_i(1)$$

$\gamma_i(t)$ ist die erwartete Häufigkeit, sich in Zustand i zum Zeitpunkt t aufzuhalten. Diese ergibt sich aus:

$$\gamma_i(t) = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N} \alpha_j(t)\beta_j(t)}$$

α und β können rekursiv ermittelt werden über:

$$\begin{aligned}
\alpha_i(1) &= \pi_i q_i(o_1) \\
\alpha_j(t+1) &= \sum_{i=1}^{N} \alpha_i(t) p(i,j) q_j(o_{t+1}) \\
\beta_i(T) &= 1 \\
\beta_i(t) &= \sum_{j=1}^{N} p(i,j) q_j(o_{t+1}) \beta_j(t+1)
\end{aligned}$$

Dann werden p und q angepasst mit:

$$p(i,j) = \frac{\text{Anzahl der Übergänge von i nach j}}{\text{Anzahl der Übergänge von i zu einem beliebigen Zustand}}$$

$$q_i(k) = \frac{\text{Anzahl der Zeitschritte in i in denen k beobachtet wurde}}{\text{Anzahl der Zeitschritte in i}}$$

Diese Operationen werden so lange ausgeführt bis sich keiner der Werte um mehr als ein Mindestwert (hier: 0,01) verändert oder die festgelegte Höchstanzahl an Iterationen (hier: 100) erreicht ist.

5 Evaluierung

Im Folgenden werden vor allem Experimente mit dem bestehenden Ansatz und verschiedenen Parametern betrachtet. Es wurde beobachtet, dass Übungen, die keine oder wenig Bewegung des Arduinos umfassen, die Erkennungsleistung maßgeblich verschlechtern. Es liegt die Vermutung nahe, dass die meisten Übungen im Stillstand beginnen, enden oder einen „Umkehrpunkt“ beinhalten. Damit wird meist ein Zeitschritt erfasst, welcher eine Beschleunigung nahe null aufweist. Man kann diesen Sprung in den folgenden Diagrammen auch beobachten: der Sprung in der Erkennungsleistung von Abbildung 2 auf Abbildung 3 liegt unter anderem daran, dass die vierte Übung viel Stillstand beinhält.
Die folgenden Diagramme wurden mit verschieden vielen Übungen, die erkannt werden sollen erstellt. Auf der Hochachse ist der Anteil an richtig erkannten Übungen aufgetragen. Auf den anderen Achse jeweils die Clusteranzahl und die Ausführungsanzahl während der Trainingsphase. Hier werden nicht die üblichen Maße Precision und Recall genutzt, da wie oben beschrieben keine feste Mindestwahrscheinlichkeit festgelegt werden konnte und damit ein Nichterkennen nicht scharf definiert werden kann, da das entsprechende Hidden Markov Modell zwar eine hohe Wahrscheinlichkeit aufweisen kann, ein Fehlerhaftes jedoch eine noch größere, was nichts über das beobachtete Modell aussagt.
Die Daten zu den Schaubildern bauen aufeinander auf, die Ausführungen für die Übungen sind immer dieselben. Es werden nur mehr Übungen oder Ausführungen betrachtet. Die gesehenen Ausführungen sind also entlang einer Linie immer die selben. Die Hidden Markov Modelle haben jeweils 5 versteckte Zustände, was sich in Experimenten als bester Kompromiss zwischen Lernzeit und Erkennungsleistung herausgestellt hat.

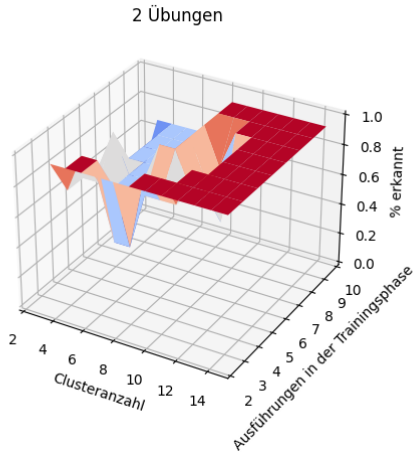

Abbildung 1: Diagramm für 2 verschiedene Übungen

Bei der Unterscheidung von 2 Übungen (Abbildung 1) ist die Erkennungslei-

stung im Allgemeinen sehr hoch, aber ein höheres k, also eine höhere Anzahl an Clustern, trennt hier die Übungen schärfer voneinander. Die Anzahl an Ausführungen spielt in diesem Fall dagegen eine eher untergeordnete Rolle für die Erkennungsleistung.

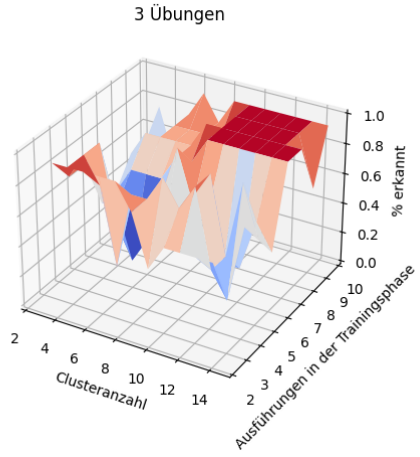

Abbildung 2: Diagramm für 3 verschiedene Übungen

In Abbildung 2 wird deutlich, dass die Anzahl an Ausführungen mehr ins Gewicht fällt. Auch hier ist wieder zu erkennen, dass ein etwas größeres k bessere Ergebnisse erzielt. Dieses sollte man jedoch auch nicht zu hoch wählen, da sonst einzelne, tatsächliche Cluster aufzuteilen welche das Hidden Markov Modell dann als gleichwertig erlernen muss. Außerdem wird damit auch das Hidden Markov Modell umfangreicher und mehr Ausführungsdaten werden benötigt um gute Ergebnisse zu erzielen.

Für die Unterscheidung von 4 Übungen (Abbildung 3) ist der oben erwähnte negative Sprung zu sehen, da die vierte Übung weniger Bewegung umfasst. Der bisher ersichtliche Vorteil von mehr Clustern scheint sich zu verkleinern, ist aber noch erkennbar. Eine höhere Ausführungsanzahl scheint hier den klaren Vorteil zu bringen.

Mit 5 Übungen als Grundlage (Abbildung 4) verfestigt sich der bisherige Trend, dass mehr Ausführungen zu deutlich besseren Ergebnissen führt. Die Clusteranzahl bietet keinen deutlichen Vorteil mehr.
Abschließend lässt sich feststellen, dass mehr Ausführungen bei mehr Übungen die Erkennungsleistung deutlich steigern. Darüber hinaus sollte auch immer ein k gewählt werden, das nicht zu klein ist, vor allem für wenige Übungen.

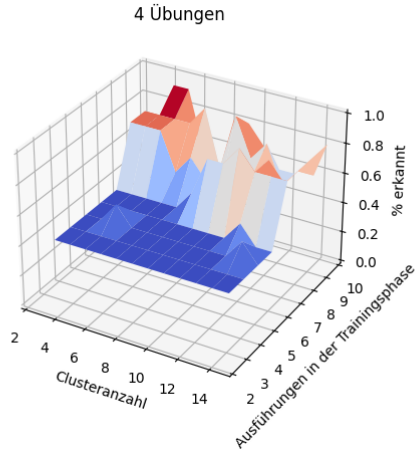

Abbildung 3: Diagramm für 4 verschiedene Übungen

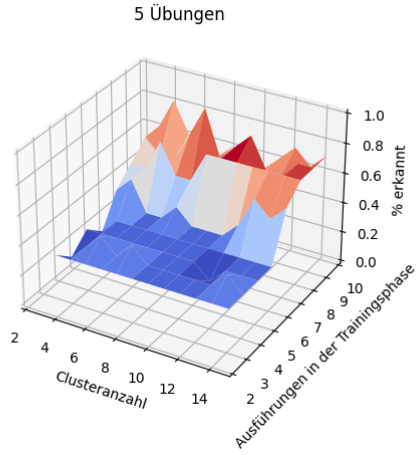

Abbildung 4: Diagramm für 5 verschiedene Übungen

Als Ausgangspunkt empfiehlt es sich $k = 5$ zu setzen, 5 versteckte Zustände und 5 Ausführungen zu nutzen. Diese Parameter können dann natürlich je nach Bedarf angepasst werden.

Literatur

[Hmm, 2021] (2021). Hmmlearn/hmmlearn. hmmlearn.

[Skl, 2021] (2021). Sklearn.cluster.KMeans — scikit-learn 0.24.1 documentation. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html.

[Rabiner, 2001] Rabiner, L. (2001). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 89(8):0200–0257.

[Wunsch, 2001] Wunsch, H. (2001). Der Baum-Welch Algorithmus fur Hidden Markov Models, ein Spezialfall des EM-Algorithmus. page 34.