Hausarbeit SME-PHY-B: Wahlthema 2 - Fitnesszähler

Joel Ewig

26. März 2021

Zusammenfassung

Sensorbasiert sollen ähnlich einer Fitnessuhr verschiedene SportÜbungen erkennt sowie die Anzahl wie oft diese jeweils ausgeführt wurden. 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 HMM 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

1. Accelerometer und Gyrometer mittels mehrdimensionalenm Kalman-Filter bereinigt
2. gefilterte Daten werden mittels K-Means geclustert
3. clusterzugehörigkeit als beobachtung in ein HMM zur Erkennung von Aktivität und Zählen

Es folgt ein Beispiel für eine typische Ausführung während der Detektionsphase: Während der Ausführung einer Übung werden die Daten des Gyroskops und des Beschleunigungssensors erfasst. Für 5 Zeitschritte haben die Gyroskopdaten die 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 die 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 bereinigt und zusammengeführt, das sieht dann so aus:

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

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

$$\begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \\ g_{11} \\ g_{21} \\ g_{31} \end{bmatrix} \rightarrow c_1$$

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:

$$[c_1, c_2, c_3, c_4, c_5] \rightarrow p_1, p_2, ..p_n$$

Aus den Emissionswahrscheinlichkeiten p wird die größte gewählt. Die zum gewählten Hidden Markov Model 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, dass 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

Der mehrdimensionale Kalman-Filter soll hier vernachlässigt werden, da er nicht unbedingt nötig ist für die Erkennung. Mit einem solchen Kalman-Filter wäre die Erkennung vermutlich robuster, da mit bereinigten Daten „engere “ Cluster zu erwarten wären. Hier wird aus Zeitgründen auf einen Kalman-Filter verzichtet, da dieser die Ergebnisse zwar verbessern würde, allerdings nicht unbedingt notwendig ist.
Ein solcher Filter würde wie folgt funktionieren;

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

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

Dabei müssen $A^{[t]}$, $B^{[t]}$ und $C^{[t]}$ gesetzt und experimentell überprüft werden. $\epsilon^{[t]}$ modelliert 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 Phasen

Der Ansatz nutzt in der Praxis zwei Phasen. In der Ersten, der Lernphase, wird die Anzahl der zu erkennenden Übungen sowie für jede Übung eine Anzahl an Ausführungen festgelegt. Nachdem diese ausgeführt wurden und fertig verarbeitet wurden ist die Lernphase beendet. In der Detektionsphase wird eine Anzahl an zu erkennenden Ausführungen festgelegt. Nachdem diese ausgeführt wurden, wird das Ergebnis der Detektion ausgegeben und die Detektionsphase ist beendet.

4.1 Lernphase

Die einkommenden 6-dimensionalen Daten werden wie oben beschrieben mit einem K-Means geclustert. Der K-Means wird auf den kompletten Trainingsdaten, also unabhängig von den Übungen, initialisiert. Jeder Punkt bekommt dadurch einen Clusternamen zugewiesen, umgesetzt als Nummern, welche weitergegeben werden. Die Clusternamen sind die Beobachtungen die als Eingabe für das Lernen der HMMs dienen.
Genutzt wird ein Hidden Markov Model (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

Ein HMM 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 5. Die HMMs werden mehrfach initialisiert um zu die Wahrscheinlichkeit zu vermindern, dass man nicht über ein lokales Optimum hinauskommt. Es wird das HMM gewählt, welche 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. Das liegt an dem Ziel, mit möglichst wenigen Übungen in der Lernphase auszukommen, dadurch bleibt der Trainingsdatenumfang relativ gering.

4.2 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 wie sie in der Lernphase festgestellt wurden.
Die Beobachtungen einer Durchführung werden allen HMMs zugeführt, diese geben die Emissionswahrscheinlichkeit für die beobachtete Sequenz aus. Es wird die Übung erkannt, dessen zugehöriges HMM die höchste Wahrscheinlichkeit für die Emission der beobachteten Sequenz ausgibt.

5 Baum-Welch Algorithmus

Der Baum-Welch Algorithmus wendet den Maximum-Likelihood Algorithmus auf Hidden Markov Modelle an. Genutzt wird ein Hidden Markov Model (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 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, die Übergänge, die der wahrscheinlichste Pfad durch das HMM nutzt, wahrscheinlicher zu machen.
Für eine beobachtete Sequenz $O = o_1, .., o_T$ heißt das, dass π_i in der nächsten Iteration zu $\tilde{\pi}$ und berechnet sich durch:

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

$\gamma_i(t)$ 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:

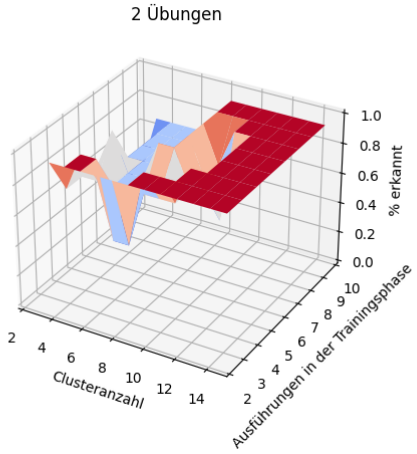
$$p(i,j) = \frac{\text{Anzahl der Übergänge von i nach j}}{\text{Anzahl der Übergänge von i zu einem beliebigen Zustand}}$$

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

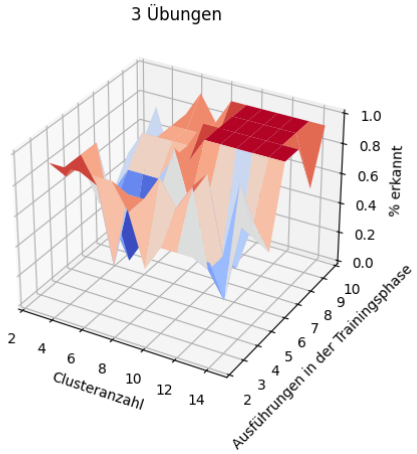
Diese Operationen werden ausgeführt bis sich keiner der Werte um mehr als ein Mindestwert (hier: 0,01) verändert oder die festgelegte Höchstanzahl an Iterationen (hier: 100) erreicht ist.

6 Evaluation

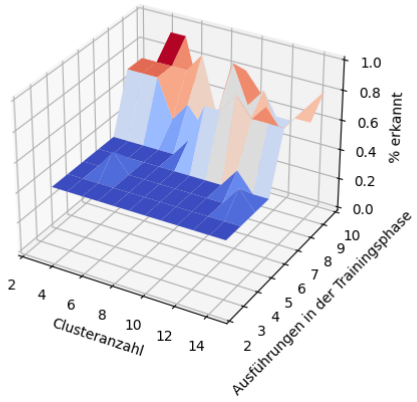
Experimentell wurde ermittelt, dass stabile Ergebnisse für alle $k \geq 3$ für einfache und wenige Übungen liefert. Ab $k \geq 5$ auch mit mehr beziehungsweise voneinander unterschiedlicheren Übungen, weshalb $k = 5$ als Default-Wert genutzt wird. Für sehr komplexe Übungen sollte k erhöht werden.



aa



4 Übungen



5 Übungen