Vorverarbeitung von Sensordaten

- Sensordaten sind oft verrauscht oder fehlerbehaftet.
- In roher Form sind die Sensordaten für die Anzeige oder die Weiterverarbeitung in der Regel ungeeignet.
- daher notwendig: Datenvorverarbeitung
- typische Aufgaben: Fehler, Ausreißer und Rauschen erkennen und behandeln, Aufbereitung der Daten, Standardisierung
- wichtige Methode: Filter

Fehler in Sensordaten

Wir unterscheiden:

- zufällige Fehler

  z.B. Mess- und Übertragungsfehler, können als statistisches Rauschen modelliert werden und gut durch Filter eliminiert werden.

- systematische Fehler

  z.B. falsche Kalibrierung, Drifteffekte, Verarbeitungsfehler, führen zu wesentlich gravierenden Verfälschungen.

Bedeutung der Vorverarbeitung

- Die Vorverarbeitung bzw. Filterung wird insbesondere dann wichtig, wenn wir die uns interessierende Größe nicht direkt messen können,
- sondern erst durch mehrfache Integration oder Ableitung erhalten.
- Beispiel: Höhenmeter aus Höhenmessung (für Läufer, Radfahrer). Für Höhe $h(t)$ folgt für die Höhenmeter hm zwischen Zeitpunkten t_0 und t_1:

$$hm = \int_{t_0}^{t_1} |h'(x)| dx$$

- In der Ebene gilt $h(t) = c$, woraus $hm = 0$ folgt.

- Sind aber unsere Höhenmessungen verrauscht, erhalten wir durch simples Summieren der Beträge der Höhendifferenzen $\mathrm{hm} > 0$ und messen nur das Rauschen.

Filterung

- Ziel: Den Einfluss von Rauschen und Ausreißern weitgehend zu eliminieren und die eigentlichen Werte zu erkennen.
- Wir gehen von gemessenen Sensorwerten $x_i, i = 0, 1, \ldots$ zu den Zeitpunkten $t_i, i = 0, 1, 2, \ldots$ aus.
- Ein Filter bildet die Folge (x_i) der Sensorwerte auf eine Folge (y_i) ab.

Gleitender Mittelwert

- Zu den einfachsten Filtermethoden gehören gleitende statistische Maße.
- Hierbei wird für jeden Wert x_i die unmittelbare Nachbarschaft dieses Wertes betrachtet und
- ein statistisches Maß dieser Wertemenge als gefilteter Wert y_i ermittelt.
- Symmetrische gleitende statistische Maße ungerader Ordnung nutzen hierzu die Nachbarschaftsmengen

$$N_{iq} := \{x_k | k = i - (q-1)/2, \ldots, i + (q-1)/2\} \qquad q = 3, 5, 7, \ldots$$

- Gleitender Mittelwert der ungeraden Ordnung q:

$$y_i = \frac{1}{q} \sum_{k=i-\frac{q-1}{2}}^{i+\frac{q-1}{2}} x_k$$

- Problem: Diese Definition ist nur für Analysen der Messreihen geeignet (Zeitreihenanalyse). In der Online-Verarbeitung sind die Werte $x_{i+1}, \ldots, x_{i+(q-1)/2}$ zum Zeitpunkt t_i nicht bekannt.

- In der Online-Verarbeitung nimmt man daher die letzten q Sensorwerte:

$$y_i = \frac{1}{q} \sum_{k=i-q+1}^{i} x_k$$

- Hier kann q sowohl gerade als auch ungerade Werte annehmen.
- Je größer q ist, desto besser wird Rauschen herausgefiltert.
- Probleme:
  - Es werden q Messwerte benötigt, bevor der erste y-Wert berechnet werden kann.
  - Kurzzeitige reale Spitzen in den x_i Werten werden abgeflacht.

Gleitender Mittelwert für mobile Geräte

- Bei mobilen Geräten treffen die Sensorwerte in der Regel nicht zeitlich äquidistant ein.
- Daher: Zeitfenster der Größe q und Berücksichtigung aller Sensorwerte im Zeitfenster.
- Für eine Instanz x vom Typ `SensorEvent` sei $x.t$ der Zeitpunkt, für den x ermittelt wurde (Instanzvariable `timestamp`) und v sei der zugehörige Wert (Komponente von `values`).

$$N_{iq} = \{k \mid x_k.t \in [x_i.t - q, x_i.t]\}$$

$$
\begin{aligned}
y_i.t &= x_i.t \\
y_i.v &= \frac{1}{|N_{iq}|} \sum_{k \in N_{iq}} x_k.v
\end{aligned}
$$

Effiziente Berechnung gleitender Mittelwerte (1)

- naiv und ineffizient: Für jedes Eintreffen eines `SensorEvent` (in der Methode `onSensorChanged()`) die Events der Menge $\{k|x_k.t \in [x_i.t - q, x_i.t]\}$ ermitteln und dann die Summenbildung durchführen.
- Aufwand: abhängig von der Größe q, proportional in der Anzahl der Elemente, die sich in einem Zeitfenster der Größe q befinden.
- Ziel: Verarbeitung der `SensorEvents` in amortisierter konstanter Zeit, unabhängig von der Größe von q

Effiziente Berechnung gleitender Mittelwerte (2)

- Speichere die `SensorEvents` in einer (doppelt) verketteten Liste l (Typ: `LinkedList<SensorEvent>`).
- Die Liste l wird als FIFO organisiert.
- Variable sum sei die Summe aller Sensorwerte in der Liste.
- Wenn ein neuer `SensorEvent` x eintrifft:
  - Hänge x an das Ende von l, $sum += x.v$.
  - Solange für das erste Element x' von l gilt $x'.t < x.t - q$ lösche x' und führe $sum -= x'.v$ aus.
  - $y.t = x.t$ und $y.v = sum / l.\texttt{count}()$

Diskussion Mittelwert

- Unter üblichen Voraussetzungen sind gleitende Mittelwerte gut geeignet, um statistisches Rauschen zu filtern (z.B. erwartungstreu).
- gleitende Mittelwerte sind aber empfindlich (nicht robust) gegenüber anderen Fehlern, z.B. Übertragungs- und Verarbeitungsfehlern.
- Beispiel Tafel
- Ein robusterer Schätzer ist der Median.

Median

- Für eine geordnete Stichprobe $(x_1, \ldots, x_n)$ von n Messwerten ist der Median med definiert durch

$$\mathrm{med} = \begin{cases} x_{\frac{n+1}{2}} & \text{falls } n \text{ ungerade} \\ \frac{1}{2}(x_{\frac{n}{2}} + x_{\frac{n}{2}+1}) & \text{falls } n \text{ gerade} \end{cases}$$

- Für die Sensordatenverarbeitung bei mobilen Geräten nutzen wir den Median wieder in einer gleitenden Variante.

Gleitender Median

- Für ungerades q ist der Median med zum Zeitpunkt t_i der Messwert, für den gilt:

$$|\{x_k \in \{x_{i-q+1}, \ldots, x_i\} | x_k \leq med\}| = |\{x_k \in \{x_{i-q+1}, \ldots, x_i\} | x_k \geq med\}|$$

- Treten die Sensorwerte nicht zeitlich äquidistant ein, wird wieder mit einem entsprechenden Zeitfenster gearbeitet.

Effiziente Rang- und Medianberechnung (1)

- Um Minimum (oder Maximum) einer n-elementigen Menge zu bestimmen, benötigen wir ohne Vorverarbeitung (und zusätzliche Voraussetzungen) Zeit $O(n)$.
- Für festes r können wir das r-größte Element ebenfalls in Zeit $O(n)$ bestimmen.
- Allerdings nimmt der konstante Aufwand mit zunehmenden r ebenfalls zu. (Beispiel: zweitgrößtes Element)
- Wie können wir vorgehen, wenn der Rang r Teil der Eingabe (und nicht fest) ist?

- Erste Idee: Sortieren, dann direkter Zugriff auf das r-größte Element
- Zeitaufwand mit effizientem Sortieralgorithmus: $O(n \log n)$
- Kann das Problem in linearer Zeit gelöst werden?

Effiziente Rang- und Medianberechnung (2)

- Ansatz: Teile-und-herrsche
- Wähle beliebiges Pivotelement aus den Werten $x_1, \ldots, x_n$, z.B. x_1.
- O.B.d.A. seien die x_i alle verschieden.
- Bilde zwei Teilmengen der Menge $M = \{x_2, \ldots, x_n\}$:

$$\begin{aligned} L &= \{x_k \in M | x_k < x_1\} \\ U &= \{x_k \in M | x_k > x_1\} \end{aligned}$$

- Unterscheide drei Fälle:

  1. $|L| = r - 1$
     Dann ist x_1 das r-größte Element. STOP!
  2. $|L| \geq r$
     Dann befindet sich das r-größte Element in L. Wende das Verfahren rekursiv auf L an.
  3. $|L| < r - 1$
     Dann ist das r-größte Element in U und dort das $(r - |L| - 1)$-größte. Wende das Verfahren rekursiv auf U an mit $r' = r - |L| - 1$ statt r.

- Zeitaufwand: im Mittel $O(n)$

- weitere Verbesserungen garantieren Zeit $O(n)$ im Worst-Case

Effiziente Rang- und Medianberechnung

- Weitere Effizienzsteigerungen sind mit angepassten ausgeglichenen Bäumen möglich.
- Beispiel: AVL-Bäume, Knoten erweitert um die Anzahl der Werte, die in dem Unterbaum mit dem Knoten als Wurzel gespeichert werden.
- Bei Ausgleichsoperationen muss diese Anzahl mit angepasst werden.
- zusätzliche Datenstruktur für die zeitlichen Aspekte notwendig, denn Löschen/Einfügen passiert auf Basis der Zeit
- Aufwand: amortisierte Zeit $O(\log n)$
- siehe auch: Order Statistic Tree

Statistische Eigenschaften des Medians

- Vorteil: robuster gegenüber Datenfehlern
- Nachteil: Der Median (Stichprobenmedian) ist ein Schätzer für das 50%-Quantil (Median) einer Verteilung, nicht für deren Erwartungswert.
- Bei symmetrischen Verteilungen wie der Normalverteilung sind Median und Erwartungswert identisch.
- Bei anderen Verteilungen muss dies nicht der Fall sein.

- Beispiel: X sei exponentialverteilt mit Dichtefunktion

$$f(x) = \begin{cases} \lambda \cdot e^{-\lambda x} & \text{für } x \geq 0 \\ 0 & \text{sonst} \end{cases}$$

- $E(X) = \frac{1}{\lambda}$, aber Median $Q_{0.5} = \frac{\log 2}{\lambda}$

- Herleitung ✎

- Wenn wir den Parameter λ aus den Messwerten schätzen wollen, müssen wir dementsprechend unterschiedliche Berechungsmethoden für Mittel und Median verwenden.

Exponentielle Filter

- gleitende gewichtete Mittelwerte
- neuere Daten gehen stärker in die Berechnung für y_i ein:

$$y_i = \frac{\sum_{k=1}^{i} \beta_k \cdot x_k}{\sum_{i=1}^{k} \beta_k}$$

mit

$$\beta_k = e^{-\lambda(t_i - t_k)}$$

- Der Parameter $\lambda > 0$ bestimmt, wie schnell vergessen wird.

- Für $\lambda \to \infty$ wird der Effekt der Filterung geringer.

- Üblicherweise wieder Beschränkung auf ein Zeitfenster

$$N_{iq} := \{k | x_k.t \in [x_i.t - q, x_i.t]\}.$$

  um nicht alle alten Werte in der Berechnung berücksichtigen zu müssen.

- Statt exponentiell (fallender) Gewichtung sind auch andere Gewichtsfunktionen möglich.

- Lineares Gedächtnis:

$$\beta_k = \begin{cases} 1 - \frac{1}{q}(x_i.t - x_k.t) & \text{für } k \in N_{iq} \\ 0 & \text{sonst} \end{cases}$$

- Polynomiales Gedächtnis:

$$\beta_k = \begin{cases} \left(1 - \frac{1}{q}(x_i.t - x_k.t)\right)^{\alpha} & \text{für } k \in N_{iq} \\ 0 & \text{sonst} \end{cases}$$

  mit $\alpha \geq 1$.

- Für das lineare Gedächtnis ist eine effiziente Implementierung möglich.