technische universität
dortmund

Bachelorarbeit

Effizienteres k-means Clustering von Zeitreihen mit Dynamic Time Warping durch kaskadiertes Berechnen von unteren Schranken

Lukas Pfahler

Juli 2013

Betreuer:
Prof. Dr. Katharina Morik
Dipl.-Inform. Marco Stolpe

Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl für Künstliche Intelligenz (LS-8)
http://www-ai.cs.uni-dortmund.de/

Inhaltsverzeichnis

Kapitel 1

Einführung

Heutzutage werden in immer mehr Bereichen immer mehr Daten erfasst und immer größere Datenmengen mithilfe von Data-Mining Techniken computergestützt ausgewertet.

Data-Mining ist essentieller Bestandteil moderner Forschung in verschiedensten Disziplinen - sei es in der Medizin, in der Physik oder der Chemie, in der Industrie oder auch in der Soziologie und in den Wirtschaftswissenschaften, überall fallen entweder in Experimenten oder im Betriebsablauf Datenmengen an, die nur noch mit Methoden der Informatik oder Statistik analysiert werden können.

Diese Data-Mining Techniken oder Lernverfahren sind standardmäßig für Vektoren aus dem $\mathbb{R}^n$ definiert, in der Praxis soll aber auch mit Daten anderer Form gearbeitet werden, beispielsweise mit Texten, Bildern oder auch Audio- und Videodaten. In diesem Fall behilft man sich häufig damit, eine Reihe von reellwertigen Merkmalen aus den Daten zu extrahieren, um so eine Transformation der Daten in den $\mathbb{R}^n$ zu definieren. Dies kann im Fall von Textdaten beispielsweise über Häufigkeiten bestimmter Wörter geschehen, im Fall von Audiodaten können mit Methoden der Signalanalyse die hauptsächlich vorkommenden Frequenzen mit ihren zugehörigen Amplituden ermittelt werden.

Eine weitere relevante Form, in der Daten vorliegen können, sind Zeitreihen. Beispielsweise werden Sensordaten meistens nicht nur an einem Zeitpunkt erhoben, sondern über Zeitspannen. Die so entstehenden Listen von reellwertigen, zeitabhängigen Messwerten werden als Zeitreihen bezeichnet; ein wesentlicher Unterschied zu Vektoren aus dem $\mathbb{R}^n$ ist, dass zwei gemessene Zeitreihen nicht zwingend die gleiche Länge haben müssen.

Auch ein Stahlwerk eines führenden deutschen Stahlherstellers liefert Zeitreihendaten in Form von Messkurven, die an verschiedenen Stationen des Produktionsprozess proto-

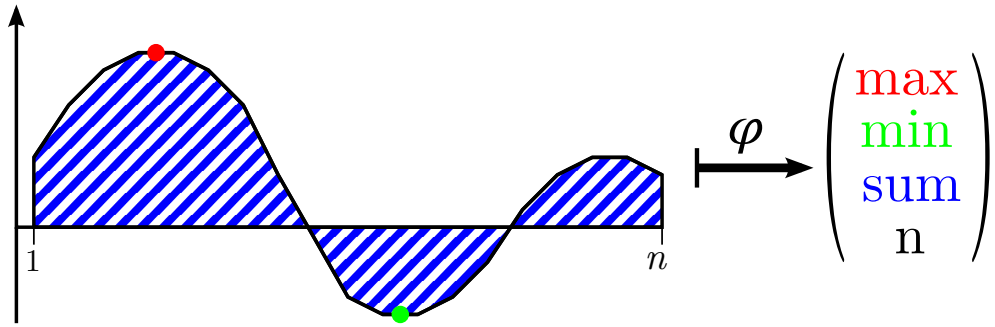


Abbildung 1.1: Beispiel für die Transformation einer Zeitreihe in den $\mathbb{R}^n$ durch Extraktion von 4 Merkmalen: Maximum, Minimum, Summe aller Messwerte und Länge der Zeitreihe. Offensichtlich kann leicht eine Zeitreihe konstruiert werden, die die selben Merkmale aufweist, jedoch eine völlig andere Form hat.

kolliert werden. Dabei handelt es sich um Temperaturen im Drehherdofen, Drehzahlen der Walzen und Druckmessungen der Walzen. Das Ziel ist, mithilfe der Messdaten schon während des Fertigungsprozesses die Qualität der Stahlblöcke vorherzusagen, um so fehlerhafte Stahlblöcke frühzeitig auszusortieren.

Es ist gängige Praxis (siehe [15, 13]) aus einer gegebenen Zeitreihe Merkmale wie die Länge, den maximalen oder minimalen Wert, den Durchschnitt aller Werte oder die Standardabweichung zu extrahieren. Für manche Aufgaben sind diese extrahierten Merkmale ausreichend, beispielsweise reicht der maximale Wert, wenn man eine Menge von Geschwindigkeitsmessungen nach Temposünder/Kein Temposünder klassifizieren will.

Jedoch ist eine solche Transformation immer mit einem Informationsverlust verbunden und insbesondere die Information der Form der Zeitreihe geht verloren. Deshalb soll in dieser Arbeit versucht werden, ein gängiges Lernverfahren des unüberwachten Lernens so zu modifizieren, dass es direkt auf Zeitreihen arbeitet.

1.1 Ziel der Arbeit

Aufgabe dieser Arbeit ist es, durch Anpassung eines gängigen Lernverfahrens des unüberwachten Lernens, dem k-means Clustering, einen Algorithmus zum effizienten Clustern von Zeitreihen mit Dynamic Time Warping Abstandsmaß zu entwickeln.

Das k-means Verfahren partitioniert eine gegebene Eingabemenge so in k Partitionen, sogenannte Cluster, dass ähnliche Elemente in einem Cluster liegen. Dabei werden die Cluster über ihre Mittelpunkte, die sogenannten Zentroide, identifiziert. Ein Clustering

wird berechnet, indem zwei Schritte iteriert werden: Die Elemente der Eingabemenge werden dem jeweils nächsten Zentroid zugewiesen. Dann werden zu den so entstehenden Clustern neue Zentroide berechnet.

Üblicherweise wird mit k-means eine Menge von reellwertigen Vektoren geclustert; um eine Menge von Zeitreihen zu clustern können reellwertige Merkmale aus den Zeitreihen extrahiert werden. Ziel dieser Arbeit ist es allerdings, ohne eine Merkmalsextraktion auszukommen und die Ähnlichkeit zweier Zeitreihen mit dem Dynamic Time Warping Abstandsmaß zu messen. Dynamic Time Warping beurteilt wie formähnlich zwei Zeitreihen sind; der angepasste Algorithmus soll also formähnliche Zeitreihen in Clustern zusammenfassen.

Im Laufe dieser Arbeit muss insbesondere das Problem gelöst werden, wie zu einer gegebenen Menge von Zeitreihen ein neues Zentroid berechnet werden kann.

Als Nachteil von Dynamic Time Warping wird häufig seine hohe, quadratische Laufzeit aufgeführt. Die von Keogh in [10] vorgestellten Ideen, den Umgang mit DTW unter anderem durch kaskadiertes Berechnen von unteren Schranken zu beschleunigen, sollen angewandt werden, um einen effizienteren Algorithmus zu entwickeln.

Der entwickelte Algorithmus soll als RapidMiner Operator in Java implementiert und bereitgestellt und bei der experimentellen Evaluation des Algorithmus verwendet werden.

1.2 Aufbau der Arbeit

Zunächst wird in Kapitel 2 der k-means Algorithmus in seiner allgemeinsten Form eingeführt. Anschließend wird ein Konvergenzbeweis für das k-means Clustering von reellwertigen Vektoren vorgeführt, aus dem wichtige Erkenntnisse für das Entwickeln eines Clustering Algorithmus für Zeitreihen gewonnen werden können.

In Kapitel 3 wird das Dynamic Time Warping Abstandsmaß zum Bestimmen der Ähnlichkeit zweier Zeitreihen eingeführt. Im k-means Algorithmus werden allerdings nicht nur einzelne Abstände berechnet, sondern gegeben eine Zeitreihe die ähnlichste aus einer Menge von Zentroiden gesucht. Diese Ähnlichkeitssuche wird als Query bezeichnet; es werden die in [10] entwickelten Ansätze zur Beschleunigung solcher Querys durch kaskadiertes Berechnen unterer Schranken präsentiert.

Das k-means Clustering und das Dynamic Time Warping werden in Kapitel 4 zu einem Algorithmus zum effizienten Clustern einer Menge von Zeitreihen kombiniert. Dabei wer-

den die in Kapitel 2 vorgestellten Schritte des k-means Algorithmus separat behandelt: Ich wende das effiziente Berechnen von Querys auf die Zuweisung der Elemente auf die nächsten Zentroide an und stelle zwei eigene Ideen und drei bestehende Ideen zum Ermitteln neuer Zentroide vor.

Mithilfe meiner in Kapitel 5 kurz vorgestellten Implementierung als RapidMiner-Operator können dann in Kapitel 6 insbesondere die verschiedenen Mittelungsverfahren auf Testdaten evaluiert werden. Anschließend wird der entwickelte Clustering-Algorithmus auf die vorliegenden Zeitreihendaten eines führenden deutschen Stahlherstellers angewandt.

Die Arbeit schließt mit einem Fazit sowie einem kleinen Ausblick in Kapitel 7.

Kapitel 2

Clustering

Clustering ist eine Data Mining Technik aus dem Bereich des unüberwachten Lernens, also dem 'Lernen ohne Lehrer'. Aus statistischer Sicht wird beim unüberwachten Lernen versucht, aus einer gegebenen Menge von Ausprägungen mehrdimensionaler Zufallsvariablen X Informationen über die zugrundeliegende Verteilung zu lernen. Anders als beim überwachten Lernen wird das Lernverfahren dabei nicht von einem 'supervisor' oder Lehrer mit weiteren Informationen über die Verteilung oder Fehlereinschätzungen versorgt (Siehe [5]).

Beim Clustering oder genauer beim partitionierenden Clustering wird versucht, eine Eingabemenge so in disjunkte Teilmengen, im Folgenden als Cluster bezeichnet, zu partitionieren, dass die Elemente eines Clusters sich maximal ähnlich sind und Elemente verschiedener Cluster maximal verschieden sind. Es wird also ein Ähnlichkeits- oder Abstandsmaß $d : \mathcal{F} \times \mathcal{F} \to \mathbb{R}^+$ benötigt, dass für zwei Elemente $x, y \in \mathcal{F}$ einen reellwertigen Abstand ermittelt. Die verbreiteste Variante des partitionierenden Clusterings, das k-means Clustering, soll im Folgenden vorgestellt werden.

2.1 k-means Clustering

k-means ist ein bekanntes Verfahren zum partitionierenden Clustern, welches zu einer gegebenen endlichen Menge $M \subseteq \mathcal{F}$, einem Abstandsmaß $d : \mathcal{F} \times \mathcal{F} \to \mathbb{R}^+$ und einem festen k eine Partitionierung von M in k Cluster berechnet. Dabei werden die Cluster über ihre Zentroide oder Mittelpunkte $z_1, ..., z_k \in \mathcal{F}$ beschrieben. Aus diesen kann eine Partition abgeleitet werden, indem jedes $x \in M$ dem Zentroid $\arg\min_{z_i} d(x, z_i)$ zugeordnet wird.

Bei k-means handelt es sich um ein Optimierungsproblem, bei dem k optimale Zentroide gesucht werden; formal soll die Gesamtkostenfunktion des Clusterings

$$cost(C_1, ..., C_k, z_1, ..., z_k) = \sum_{i=1}^{k} \sum_{c \in C_i} d(c, z_i) \tag{2.1}$$

wobei $z_1, ..., z_k$ Zentroide mit zugehörigen Clustern C_i sind, minimiert werden.
Zur Lösung des Clustering-Problems wird der folgende Algorithmus verwendet:

Definition 1 (k-means Algorithmus)**.** Der k-means Algorithmus, auch Lloyds Algorithmus, partitioniert eine Eingabemenge $M \subseteq \mathcal{F}$ mit Abstandsmaß $d : \mathcal{F} \times \mathcal{F} \to \mathbb{R}^+$ in folgenden Schritten:

1. Wähle zufällig initiale Zentroide $z_1, ..., z_k$. (I)
2. Solange sich die Gesamtkosten des Clusterings ändern:
3. Ordne die Elemente aus M dem jeweils bezüglich d nächsten Zentroid z_i zu. (II)
4. Bestimme neue Zentroide $z_1, ..., z_k$ der entstandenen Cluster $C_1, ..., C_k$. (III)
5. Gib die Zentroide $z_1, ..., z_k$ aus.

Klassischerweise wird der k-means Algorithmus für das Clustering von Vektoren aus dem $\mathbb{R}^n$ mit der quadrierten euklidischen Distanz $d(x, y) = ||x - y||^2$ beschrieben, dann können die Schritte I-III wie folgt spezifiziert werden:

(I) Wähle initiale Zentroide zufällig aus M

(II) Für alle $x \in M$: setze $x \in C_j$ mit $j := \arg\min_{1 \leq i \leq k} ||x_i - z_i||^2$

(III) Für alle $1 \leq i \leq k$ setze $z_i := \frac{1}{|C_i|} \sum_{x \in C_i} x$

Satz 1. Der k-means Algorithmus aus Definition 1 auf dem $\mathbb{R}^n$ terminiert in einem lokalen Optimum des Minimierungsproblems in Gleichung 2.1.

Beweis. (nach [3]) Seien $C_1^{(t)}, ..., C_k^{(t)}$ bzw. $z_i^{(t)}, ..., z_k^{(t)}$ die in der t-ten Iteration aktuellen Cluster bzw. Zentroide. Um Konvergenz zu zeigen, genügt es zu zeigen, dass die Gesamtkosten

$$cost(C_1, ..., C_k, z_1, ..., z_k) := \sum_{i=1}^{k} \sum_{x \in C_i} ||x - z_i||^2$$

des Clustering mit jeder Iteration monoton fallen, da es nur endlich viele Möglichkeiten gibt, die Menge M zu partitionieren. Es muss also

$$cost(C_1^{(t+1)}, ..., C_k^{(t+1)}, z_1^{(t+1)}, ..., z_k^{(t+1)}) \leq cost(C_1^{(t)}, ..., C_k^{(t)}, z_1^{(t)}, ..., z_k^{(t)}) \tag{2.2}$$

gezeigt werden. Dies wird nun in zwei Schritten bewiesen.

Durch die Zuteilung der Vektoren auf die jeweils nächsten Zentroide können die Gesamtkosten nicht erhöht werden, somit gilt

$$cost(C_1^{(t+1)}, ..., C_k^{(t+1)}, z_1^{(t)}, ..., z_k^{(t)}) \leq cost(C_1^{(t)}, ..., C_k^{(t)}, z_1^{(t)}, ..., z_k^{(t)}). \tag{2.3}$$

Im zweiten Schritt muss nun auch noch

$$cost(C_1^{(t+1)}, ..., C_k^{(t+1)}, z_1^{(t+1)}, ..., z_k^{(t+1)}) \leq cost(C_1^{(t+1)}, ..., C_k^{(t+1)}, z_1^{(t)}, ..., z_k^{(t)}), \tag{2.4}$$

gezeigt werden. Dazu muss zunächst gezeigt werden, dass durch die arithmetische Mittelung die Kosten eines Clusters C mit Zentroid z

$$cost(C, z) := \sum_{x \in C} ||x - z||^2 = \sum_{x \in C_i} \sum_{j=0}^{n} (x_j - z_j)^2$$

minimiert werden. Durch Nullsetzen der Ableitung erhält man

$$\frac{\partial}{\partial z_i} cost(C, z) = \sum_{x \in C} (-2x_i + 2z_i) = 0 \tag{2.5}$$

$$\Leftrightarrow z_i = \frac{1}{|C|} \sum_{x \in C} x_i, \tag{2.6}$$

folglich werden die Gesamtkosten der Cluster durch das Mitteln minimiert, $\frac{1}{|C|} \sum_{x \in C} x$ ist sogar globales Minimum der Kostenfunktion bei festen C.

Also folgt auch die Gültigkeit von Gleichung 2.4 und somit von Gleichung 2.2; die Gesamtkosten nehmen monoton ab. Da es nur endlich viele mögliche Partitionierungen der Eingabemenge gibt, konvergiert die Berechnung immer.

Insbesondere folgt, dass die Berechnung in einem lokales Minimum terminiert, da

$$cost(C_1, ..., C_k, z_1, ..., z_k) := \sum_{i=1}^{k} cost(C_i, z_i)$$

und somit folgt, dass die Ableitung der Gesamtkostenfunktion für die gefundenen Zentroide auch 0 ist. □

Der k-means Algorithmus im $\mathbb{R}^n$ terminiert also in jedem Fall, allerdings nicht zwangsläufig im globalen Optimum. Dabei hängt die Qualität des berechneten Clusterings stark von den zufällig ausgewählten initialen Zentroiden ab. Um eine höhere Konfidenz in die ausgegebenen Zentroide zu erhalten, bietet es sich an, den Algorithmus mehrfach auszuführen. Weiterhin ist die maximal benötigte Anzahl der Iterationen theoretisch exponentiell groß. Deshalb wird die Anzahl der Iterationen häufig künstlich auf einen konstanten Wert beschränkt.

In einer Iteration werden $k \cdot |M|$ Distanzberechnungen durchgeführt. Die Laufzeit einer Iteration beträgt also $\mathcal{O}(k \cdot |M| \cdot n)$, wobei n die in der Dimension lineare Laufzeit der quadrierten euklidischen Distanz ist. Ersetzt man die quadrierte euklidische Distanz durch ein Distanzmaß mit höherer Komplexität, beispielsweise das Dynamic Time Warping Abstandsmaß mit Laufzeit n^2, steigt die Laufzeit einer Iteration entsprechend an.

Außerdem wird deutlich, wie wichtig miteinander kompatible Abstandsmaße und Mittelungsverfahren für die Konvergenz und Optimalität der Lösung sind. Wenn Gleichung 2.4 gezeigt werden kann, also das durch ein Mittelungsverfahren berechnete Zentroid die Kosten eines Clusters gegenüber den Kosten des Clusters mit dem alten Zentroid verbessert, dann terminiert der k-means Algorithmus. Findet das Mittelungsverfahren ein lokal optimales Zentroid, terminiert der k-means Algorithmus auch in einem lokalen Optimum.

Betrachten wir ein weiteres Beispiel für eine kompatible Kombinationen von Distanzmaß und Mittelungsverfahren: Wird statt der quadrierten euklidischen Distanz die durch die L1-Norm definierte Manhattan-Distanz

$$d_{man}(x, z) = \sum_{i=1}^{n} |x_i - z_i|$$

verwendet, dann minimiert das Mittelungsverfahren, das jede Komponente z_i des neuen Zentroids z als den Median aller x_i mit $x \in C$ definiert, die Kostenfunktion $cost(C, z)$. Deshalb spricht man auch vom k-median Clustering.

Erstaunlicherweise ist es viel schwieriger bezüglich der unquadrierten euklidischen Distanz statt der quadrierten Distanz ein neues Zentroid zu berechnen. Das hierfür zu lösende Minimierungsproblem wird als Fermat-Weber Problem bezeichnet und bis heute gibt es kein exaktes Lösungsverfahren, eine Lösung kann jedoch iterativ approximiert werden. Im Allgemeinen ist es also kein einfaches Problem, zu einem Distanzmaß ein optimales Mittelungsverfahren zu finden.

Ein wesentlicher Teil dieser Arbeit ist es, ein zum im nächsten Kapitel vorgestellten Dynamic Time Warping Abstandsmaß kompatibles Mittelungsverfahren zu finden.

Kapitel 3

Dynamic Time Warping

Dynamic Time Warping, kurz DTW, ist eine Technik, die ihren Ursprung in der Spracherkennung hat [20]. Dort hatte man eine Datenbank mit Tonaufnahmen verschiedener Wörter erstellt und wollte nun für neue Aufnahmen bestimmen, welches Wort gesprochen wurde. Dabei liegen die Aufnahmen als Liste von diskreten Amplitudenwerten vor, die durch Abtastung des analogen Tonsignals in gleichmäßigen Abständen gewonnen werden.

Damit der Spracherkennungs-Algorithmus Wörter zuverlässig erkennen kann, muss er damit zurechtkommen, dass die Wörter auch unterschiedlich schnell gesprochen werden können. Dieses Problem löst das Dynamic Time Warping, indem es zwischen zwischen zwei Sequenzen ein optimales Alignment berechnet. Ein Alignment bezeichnet dabei die Menge der Paare von Zeitpunkten der Sprachaufnahmen, die verglichen werden. (Siehe Abbildung 3.1) Mithilfe dieser optimalen Alignments können auch Ähnlichkeiten zwischen verzerrten Sprachaufnahmen entdeckt werden; dies führt zu einer verlässlicheren Spracherkennung.

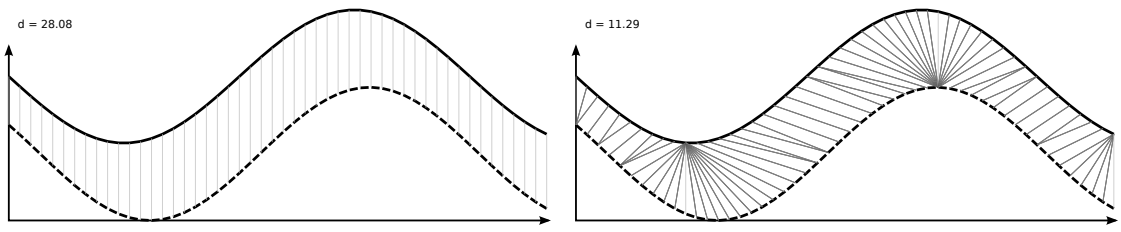


Abbildung 3.1: Zwei verschiedene mögliche Alignments zweier Zeitreihen. Links ist das vom euklidischen Abstand verwendete Alignment, rechts das von DTW berechnete Alignment.

3.1 Zeitreihe

Bevor das Dynamic Time Warping eingeführt wird, lösen wir uns allerdings von den Sprachaufnahmen und abstrahieren sie auf den Begriff der Zeitreihe, der auch in der Einleitung schon angedeutet wurde.

Definition 2 (Zeitreihe)**.** Unter einer Zeitreihe t wird eine geordnete reellwertige Liste $t = [t_1, t_2, ..., t_n]$, $t_i \in \mathbb{R}$ beliebiger Länge n verstanden. Diese werden häufig als Graph der Funktion $i \mapsto t_i$, $i \in [1, .., n]$ visualisiert.

Definition 3 (z-normalisiert)**.** Eine Zeitreihe t heißt z-normalisiert, wenn

$$\mu := \frac{1}{n}\sum_{i=1}^{n} t_i = 0 \text{ und } \sigma^2 := \frac{1}{n}\sum_{i=1}^{n}(t_i - \mu)^2 = 1.$$

Eine nicht z-normalisierte Zeitreihe t kann demnach in eine z-normalisiere Zeitserie t' transformiert werden:

$$t'_i = \frac{t_i - \mu}{\sigma} \text{ für } 1 \leq i \leq n.$$

Keogh et al. betonen in [10] ausdrücklich die Wichtigkeit des Normalisierens:

> In order to make meaningful comparisons between two time series, both must be normalized. While this may seem intuitive, and was explicitly empirically demonstrated a decade ago in a widely cited paper [19][1], many research efforts do not seem to realize this.

Sie demonstrieren die Wichtigkeit durch einen Test eines 1-NN Klassifizierers auf 45 Datensätzen. Durch Skalieren und Verschieben der Zeitreihen um einen Faktor von nur 5% verschlechtert sich die Vorhersagegenaugikeit auf jedem Datensatz um mindestens 50%.

3.2 Abstandsmaße für Zeitreihen

Abstandsmaße sind eine wichtige Komponente vieler Data-Mining-Algorithmen und Maschinenlernaufgaben. Auf dem normierten Raum $\mathbb{R}^n$ lassen sich Abstandsmaße beispielsweise über die Manhattan-Norm oder die Euklidische Norm definieren; und auch für gleichlange Zeitreihen lässt sich die eine Euklidische Distanz definieren.

[1] E. Keogh and S. Kasetty. 2003. On the need for time series data mining benchmarks: a survey and empirical demonstration. Data Mining and Knowledge. Discovery 7, 4, 349-371.

Definition 4 (Euklidische Distanz)**.** Die euklidische Distanz zweier Zeitreihen s, t gleicher Länge n ist definiert als:

$$d_{ed}(s,t) = \sqrt{\sum_{i=1}^{n}(s_i - t_i)^2}.$$

Wie schon am Beispiel der Spracherkennung festgestellt ist die Limitierung auf das Vergleichen von Zeitreihen selber Länge in der Praxis problematisch. Deshalb bietet es sich an, beim Vergleich zweier Zeitreihen ein mächtigeres Abstandsmaß zu verwenden, das Dynamic Time Warping, kurz DTW (siehe [1]).

Während die euklidische Distanz immer nur genau ein Element der Zeitreihe mit dem entsprechenden der anderen Zeitreihe aligniert (siehe Abbildung 3.1), weicht DTW diese Beschränkung auf und betrachtet auch andere mögliche Alignments, um ein günstigstes Alignment zu finden. Dabei beschränkt sich DTW auf eine bestimmte Klasse von Alignments, die sogenannten Warping-Pfade:

Definition 5 (Warping-Pfad)**.** Seien s und t Zeitreihen der Längen n beziehungsweise m. Ein Warping-Pfad $p_{s,t}$ ist eine Liste von Indexpaaren $[p_1, .., p_L]$, $p_i \in [1,n] \times [1,m]$, sodass:

- $p_1 = (1,1)$ und $p_L = (n,m)$ (Beschränkung auf globale Alignments)
- Wenn $p_i = (a,b)$, dann gilt $p_{i-1} = (a',b')$ mit $a - a' \geq 0$ und $b - b' \geq 0$ (Monotonie)
- Wenn $p_i = (a,b)$, dann gilt $p_{i-1} = (a',b')$ mit $a - a' \leq 1$ und $b - b' \leq 1$ (Stetigkeit)

Es gilt folglich $\max\{n,m\} \leq L \leq n + m$.
Die Gesamtkosten eines Warping-Pfades $p_{s,t}$ sind definiert als

$$cost(p_{s,t}) = \sum_{(i,j)\in p_{s,t}} (s_i - t_j)^2.$$

Nun kann das DTW-Abstandsmaß definiert werden; unter allen Warping-Pfaden wird derjenige gesucht, dessen Kosten minimal sind.

Definition 6 (Dynamic Time Warping)**.** Gegeben zwei Zeitreihen s, t der Längen $n, m > 0$ ist

$$d_{dtw}(s,t) = \sqrt{\min\{cost(p_{s,t}) \mid p_{s,t} \text{ ist Warping-Pfad}\}}.$$

$p^*_{s,t} := \arg\min\{cost(p_{s,t}) \mid p_{s,t} \text{ ist WP}\}$ wird als optimaler Warping-Pfad der Zeitreihen s und t bezeichnet.

Diese Berechnung des optimalen Warping-Pfades kann als dynamisches Programm formuliert werden: Da ein Warping-Pfad monoton und stetig ist, hat ein Indextupel $p_i = (a,b)$ mit $a, b > 1$ nur drei mögliche Vorgänger: $(a-1, b-1), (a-1, b)$ oder $(a, b-1)$. Ein optimaler Warping-Pfad kann also berechnet werden, indem rekursiv alle drei Kandidaten ausprobiert werden und nur der beste ausgegeben wird. Unabhängig davon erhöht das Indexpaar (a,b) die Gesamtkosten des Pfades um $(s_a - t_b)^2$.

Weiterhin gilt, dass es nur genau einen Warping-Pfad zwischen einer Zeitreihe der Länge 1 und einer beliebigen anderen Zeitreihe gibt. Diese Erkenntnisse können in folgender rekursiven Funktionsvorschrift zusammengefasst werden:

$$d_{dtw}(s,t) = \sqrt{dtw(s,t)}$$

$$dtw([\sigma],t) = \sum_{\sigma' \in t} (\sigma - \sigma')^2 \text{ mit } \sigma \in \mathbb{R}$$

$$dtw(s,[\sigma]) = \sum_{\sigma' \in s} (\sigma - \sigma')^2 \text{ mit } \sigma \in \mathbb{R}$$

$$dtw([s_1,\ldots,s_n],[t_1,\ldots,t_m]) = (s_n - t_m)^2 + \min \begin{cases} dtw(\quad [s_1,\ldots,s_n] \quad , \quad [t_1,\ldots,t_{m-1}] \quad) \\ dtw(\quad [s_1,\ldots,s_{n-1}] \quad , \quad [t_1,\ldots,t_m] \quad) \\ dtw(\quad [s_1,\ldots,s_{n-1}] \quad , \quad [t_1,\ldots,t_{m-1}] \quad) \end{cases}$$

Diese Rekurrenzen von dtw lassen sich jetzt als dynamisches Programm implementieren, in dem eine Matrix D der Größe $n \times m$ in $\mathcal{O}(n \cdot m)$ Laufzeit ausgefüllt wird. Aus dieser Matrix lässt sich in einem zweiten Schritt auch ein optimaler Warping-Pfad ableiten, dabei wird der Pfad rückwärts erzeugt:

Jeder Warping-Pfad muss auf (n,m) enden. Es gilt also $p_L = (n,m)$. Ausgehend von einem beliebigen Element des Pfades $p_l = (i,j) \neq (1,1)$ kann der Vorgänger bestimmt werden:

$$p_{l-1} = \begin{cases} (1, j-1) & \text{falls } i = 1 \\ (i-1, 1) & \text{falls } j = 1 \\ \arg\min\{D[i-1,j], D[i,j-1], D[i-1,j-1]\} & \text{sonst} \end{cases}$$

Jeder Warping-Pfad beginnt mit dem Tupel $(1,1)$.
Ein optimaler Warping-Pfad lässt sich also in $\mathcal{O}(L)$ und somit in $\mathcal{O}(n+m)$ ablesen.

3.3 Globale Constraints

Ein Nachteil von Dynamic Time Warping gegenüber dem Euklidischen Abstand ist die höhere, quadratische Laufzeit. Eine Möglichkeit, diese Laufzeit zu verringern, ist es, DTW um Globale Constraints zu erweitern, das heißt die Menge möglicher Warping-Pfade weiter zu beschränken: Anstatt alle Indizes aus $[1,n] \times [1,m]$ zuzulassen, werden Elemente eines Warping-Pfads auf eine Teilmenge $R \subseteq [1,n] \times [1,m]$ beschränkt; es gilt also $p_i \in R$.

Jede Beschränkung des Suchraums führt natürlich direkt zu einer Laufzeitverbesserung. Darüber hinaus können globale Constraints aber auch problemabhängig degenerierte Pfade ausschließen und so beispielsweise in Data-Mining Problemen zu besseren Resultaten führen (vgl. hierzu [17]).

Die beiden am häufigsten verwendeten Constraints sind das Sakoe-Chiba-Band sowie das Itakura-Parallelogramm:

Definition 7 (Sakoe-Chiba-Band)**.** Das Sakoe-Chiba-Band der Breite r schränkt die Menge möglicher Warping-Pfad-Elemente auf

$$p_i \in \left\{(x,y) \mid x \in [1,n],\ y \in \left[\frac{m-r}{n-r} \cdot (x-r), \frac{m-r}{n-r} \cdot x + r\right] \cap [1,m]\right\}.$$

Das Sakoe-Chiba-Band legt also um die Diagonale der DTW-Matrix einen erlaubten Bereich konstanter Breite (siehe Abbildung 3.2.a). Ein häufiger Literaturwert für die Breite ist 10% der Länge der Zeitreihen.

Definition 8 (Itakura-Parallelogramm)**.** Das Itakura-Parallelogramm der Steigung $s \geq 1$ beschränkt die Menge möglicher Warping-Pfad-Elemente auf

$$p_i \in \left\{(x,y) \mid \frac{1}{s} \leq \frac{y}{x} \leq s \text{ und } \frac{1}{s} \leq \frac{m-y}{n-x} \leq s\right\} \subseteq [1,n] \times [1,m].$$

Anders als beim Sakoe-Chiba-Band ist die Breite des zulässigen Bereichs nicht konstant; alle Elemente die von $(1,1)$ und (n,m) ausgehend einen Pfad mit einer Steigung haben, die zwischen s und $\frac{1}{s}$ liegt, sind zugelassen (siehe Abbildung 3.2.b).

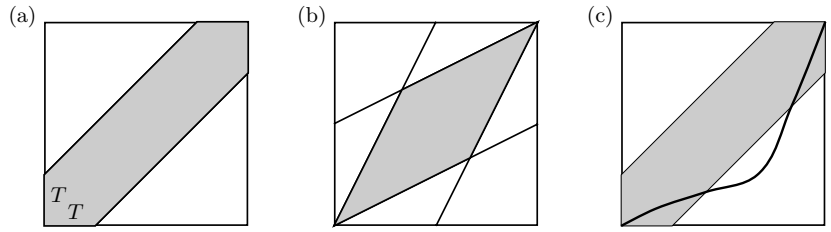


Abbildung 3.2: (a) Sakoe-Chiba-Band (b) Itakura-Parallelogramm (c) Der optimaler Pfad des unrestringierten DTW liegt nicht im zulässigen Bereich. (Quelle: [1])

3.4 Effizientere Suche nach ähnlichsten Zeitreihen mit DTW

Bei vielen Data-Mining Techniken müssen nicht nur einzelne Abstände berechnet werden, sondern aus einer Menge von Objekten das ähnlichste gesucht werden. Beim k-means Clustering muss beispielsweise für jedes Objekt das nächste Zentroid aus der Menge der Zentroide gefunden werden. Diese Ähnlichkeitssuche bezeichnet man als Anfrage oder Query.

Die Ähnlichkeitssuche auf Zeitreihen mit Dynamic Time Warping Abstand ist wie folgt

definiert: Gegeben eine Zeitreihe q und einer Menge von Zeitreihen C berechne

$$query(C, q) := \arg\min_{c \in C} d_{dtw}(c, q).$$

In [10] identifizieren Rakthanmanon et al. die Querys als Flaschenhals vieler Data-Mining Techniken:

> Most time series data mining algorithms use similarity search as a core subroutine, and thus the time taken for similarity search is the bottleneck for virtually all time series data mining algorithms.

Eine naive Implementierung muss für jedes Element aus C den Abstand zu q berechnen und kann dann die Zeitreihe mit dem niedrigsten Abstand ausgeben. Dies ist besonders durch die quadratische Laufzeit von DTW problematisch, es müssen $|C|$ DTW-Berechnungen ausgeführt werden und die Laufzeit liegt in $\mathcal{O}(|C|n^2)$, falls alle Zeitreihen die Länge n haben.

Keogh et al präsentieren in [10] eine Reihe von Ideen, die Berechnung eines Querys mit Dynamic Time Warping Abstandsmaß zu beschleunigen. Diese sollen im Folgenden vorgestellt werden.

3.4.1 Berechnung des quadrierten Abstands

Da die Wurzelfunktion monoton ist, ändern sich die relativen Abstände nicht, wenn man die Wurzelberechnung auslässt, man kann sich also die Rechenzeit für die Wurzelberechnung sparen. Im Folgenden ist mit d_{dtw} immer der quadrierte Abstand gemeint; dies vereinfacht auch das Differenzieren des Abstands enorm und macht somit komplexe Rechnungen unnötig.

3.4.2 Berechnung unterer Schranken

Um die Berechnung eines Querys zu beschleunigen, sollen günstiger berechenbare untere Schranken des DTW-Abstandes verwandt werden: Ist der bisher kleinste Abstand einer Query-Zeitreihe q zu einem Element $c_{best} \in C$

$$d_{dtw}(q, c_{best}) = bestSoFar$$

bekannt und gibt es für die Abstandsberechnung zu einem weiteren Element c_i eine untere Schranke $l_i \geq bestSoFar$, so braucht die teure Abstandsberechnung mit DTW nicht mehr

durchgeführt werden, da der tatsächliche Abstand $d_{dtw}(q, c_i) \geq l_i \geq bestSoFar$. Somit ließe sich eine aufwändige DTW-Berechnung sparen [10, 11].

Generell gilt: Je aufwändiger die Berechnung einer untere Schranke ist, desto genauer oder 'schärfer' sollte sie sein. In "umfangreichen Experimenten" wurden in [10] zwei besonders gute untere Schranken identifiziert: Die untere Schranke von Kim [11], welche in $\mathcal{O}(1)$ berechnet werden kann, sowie die untere Schranke von Keogh [9], die in $\mathcal{O}(n)$ berechnet werden kann.

3.4.3 Untere Schranke von Kim

In [11] wird die untere Schranke LB_{kim} definiert, die die Abstände zwischen den jeweils ersten, letzten sowie den maximalen und minimalen Elementen betrachtet und den größten solchen Abstand ausgibt. Durch das Auffinden des Maximums bzw. des Minimums entsteht jedoch eine lineare Laufzeit. Laut [10] sind auf z-normalisierten Zeitreihen die Differenzen der Maxima bzw. Minima jedoch meist vernachlässigbar klein, deshalb erreicht man eine vergleichbar gute Schranke, die sich in konstanter Zeit berechnen lässt, wie folgt:

Satz 2 (LB_{kim})**.** Für zwei Zeitreihen s, t der Längen n, m ist

$$LB_{kim}(s,t) = \max\left\{(s_1 - t_1)^2, (s_n - t_m)^2\right\}$$

eine untere Schranke für $d_{dtw}(s,t)$.

Beweis. Jeder Warping-Pfad p enthält die Elemente $p_1 = (1,1)$ und $p_L = (n,m)$. Somit entstehen also auf p mindestens die Kosten $(s_1 - t_1)^2$ bzw. $(s_n - t_m)^2$ und folglich insbesondere das Maximum der beiden. Somit ist LB_{kim} eine untere Schranke.

□

Wenn $n > 1$ oder $m > 1$, folgt analog auch, dass $(s_1 - t_1)^2 + (s_n - t_m)^2$ eine untere Schranke ist. Deshalb kann die untere Schranke nochmals modifiziert werden zu

$$LB_{kim'}(s,t) = \begin{cases} (s_1 - t_1)^2 & \text{falls } n = m = 1 \\ (s_1 - t_1)^2 + (s_n - t_m)^2 & \text{sonst} \end{cases}$$

3.4.4 Untere Schranke von Keogh

Die untere Schranke von Keogh [9] benutzt das für die DTW-Berechnung verwendete Global Constraint, um eine untere Schranke zu konstruieren. Dazu wird für zwei Zeitrei-

hen c, q der Längen n, m die DTW-Berechnung $d_{dtw}(c, q)$ und der zugehörige optimale Warping-Pfad betrachtet.

Aufgrund der Stetigkeit der Warping-Pfade muss für die DTW-Berechnung jedes c_i mit mindestens einem q_j aligniert werden. Das globale Constraint schränkt die Wahl von j allerdings weiter ein. Der Einfachheit halber nehmen wir an, dass j aus einem Intervall $[a_i, b_i]$ kommen muss. Dieses Intervall wird vom jeweiligen i abhängen, wie etwa beim Sakoe-Chiba Band oder beim Itakura-Parallelogramm.

Mithilfe dieses Intervalls werden nun zu c zwei Zeitreihen u und l der Länge n definiert; u steht für upper-band und l für lower-band:

$$u_i^{(q)} = \max(q_{a_i}, ..., q_{b_i}) \text{ für } 1 \leq i \leq n$$

$$l_i^{(q)} = \min(q_{a_i}, ..., q_{b_i}) \text{ für } 1 \leq i \leq n$$

Wenn jetzt $c_i < l_i^{(q)}$, also c_i insbesondere kleiner ist als alle zulässigen q_j, dann gilt auch

$$(c_i - l_i^{(q)})^2 \leq (c_i - q_j)^2$$

für alle erlaubten q_j. Jeder Warping-Pfad nimmt also beim Alignieren von c_i mindestens $(c_i - l_i^{(q)})^2$ an Kosten zu.

Analog steigen im Fall $c_i > u_i^{(q)}$ die Kosten jedes Warping-Pfades um mindestens $(c_i - u_i^{(q)})^2$.
Falls $l_i \leq c_i \leq u_i$, dann kann anhand von u bzw. l nicht abgelesen werden, wie teuer das Alignment von c_i mindestens ist, da nicht bekannt ist, für welches q_j die Kosten $(c_i - q_j)^2$ minimal sind. Die einzige gesicherte Aussage ist, dass $(c_i - q_j)^2 \geq 0$ ist.

Folgender Satz gilt also:

Satz 3 (LB_{keogh})**.** Für zwei Zeitreihen c, q der Längen n, m ist

$$LB_{keogh}(c, q) = \sum_{i=1}^{n} \begin{cases} (c_i - l_i^{(q)})^2 & \text{falls } c_i < l_i^{(q)} \\ (c_i - u_i^{(q)})^2 & \text{falls } u_i^{(q)} < c_i \\ 0 & \text{sonst} \end{cases}$$

eine untere Schranke für $d_{dtw}(c, q)$.

Beweis. Die Korrektheit der Aussage ergibt sich aus der obigen Herleitung. □

Wenn ein Query für eine Query-Sequenz q und eine Menge C von Zeitreihen selber Länge berechnet werden soll, bietet es sich an, die upper- und lower-band Sequenzen um Zeitreihe q zu berechnen, da dies so nur einmal geschehen muss.

Aber LB_{keogh} lässt sich auch für Zeitreihen unterschiedlicher Länge anwenden. Dabei müssen allerdings für die unterschiedlichen Längen von $c \in C$ auch unterschiedlich lange Zeitreihen u und l berechnet werden.

Um den Aufwand der Berechnung der Zeitreihen u und l im Zuge eines Querys zu minimieren, sollten die Zeitreihen aus C der Länge nach geordnet durchlaufen werden. So müssen nur dann Neuberechnungen durchgeführt werden, wenn die Länge einen Sprung macht.

Die LB_{keogh} ist nicht symmetrisch, somit ergibt sich durch Vertauschen der Rollen von q und c eine neue untere Schranke, insbesondere ist das Maximum beider Varianten eine untere Schranke. Im Zuge einer Query-Berechnung ist die Variante mit vertauschten Rollen jedoch rechenaufwändiger, da die Zeitreihen $u^{(c)}, l^{(c)}$ für jedes c neu berechnet werden müssen.

3.4.5 Early Abandoning

Betrachten wir noch einmal die LB_{keogh}: Sie ist definiert als Summe über n Elemente, dabei gilt, dass alle Summanden ≥ 0 sind. Im Verlauf einer Berechnung der LB_{keogh} mithilfe einer For-Schleife und einer Zählvariable steigt der Wert dieser Variablen also monoton an. Folglich gilt: Wenn der Wert einmal $bestSoFar$ überschreitet, ist auch das Endergebnis größer als $bestSoFar$ - ein vollständiges Ausführen der Berechnung ist ergo unnötig. Dieses frühzeitige Abbrechen der Berechnung bezeichnet Keogh in [10] als Early Abandoning.

Das selbe Prinzip lässt sich auch auf die Berechnung der DTW anwenden. Wie die LB_{keogh} ist auch die d_{dtw} als Summe über positive Summanden definiert. Jedoch wird die DTW berechnet, indem eine $n \times m$ Matrix D spaltenweise ausgefüllt wird. Es gilt aber: Sobald in einer Spalte von D alle Elemente $bestSoFar$ überschreiten, kann das Endresultat der Berechnung nicht kleiner $bestSoFar$ sein. Die Berechnung kann also auch frühzeitig abgebrochen werden.

Die DTW-Berechnung kann unter Zuhilfenahme der LB_{keogh} aber noch weiter optimiert werden, indem der noch nicht ausgeführte Teil der DTW-Berechnung mit der LB_{keogh} abgeschätzt wird.

Wenn die i-te Spalte der Matrix ausgefüllt wurde, gilt

$$\min_{1 \leq j \leq m} d_{ij} + LB_{keogh}([q_{i+1}, .., q_n], c) \leq d_{dtw}(q, c),$$

also ergibt sich durch das Abschätzen der verbleibenden DTW-Berechnung eine neue untere Schranke. Wenn diese untere Schranke $bestSoFar$ übersteigt, kann die Berechnung demnach auch frühzeitig abgebrochen werden. Dieses Vorgehen setzt natürlich vorraus, dass $LB_{keogh}(q, c)$ im Vorfeld berechnet wurde und also insbesondere die Zeitreihen $u^{(c)}$ und $l^{(c)}$ bestimmt wurden.

3.4.6 Multithreading

Der Vollständigkeit halber sollte auch Multithreading als wichtige Optimierung genannt werden. Ohne Anpassung des Query-Codes erreicht man in vielen Anwendungen schon große Geschwindigkeitsverbesserungen, in dem einfach multiple Queries parallel ausgeführt werden. Auch ein einziges Query lässt sich parallelisieren, indem die Menge C in disjunkte Teilmengen zerlegt wird, für die dann parallel kleinere Querys berechnet werden; durch Selektieren des minimalen Resultats können die Teilquerys dann wieder kombiniert werden.

3.4.7 Die UCR-Suite

Keogh und Rakthanmanon [10] kombinieren die vorgestellten Ideen zu einem Framework für effiziente Querys nach Subsequenzen in Zeitreihen, der UCR-Suite, benannt nach der University of California Riverside.

Die auch für das Suchen nach vollständigen Sequenzen relevante Idee ist, die verschiedenen unteren Schranken kaskadiert auszuführen: Zunächst wird die LB_{kim} in konstanter Laufzeit berechnet. So können viele schlechte Kandidaten ohne starken Rechenaufwand ausgeschlossen werden, obwohl LB_{kim} eine sehr schwache untere Schranke ist. Anschließend wird die $LB_{keogh}(c, q)$ in linearer Laufzeit berechnet. Dabei wird Early Abandoning verwendet, sodass nicht mehr als nötig berechnet wird. Falls ein Kandidat c auch diese Schranke erfüllt, wird noch die $LB_{keogh}(q, c)$ mit vertauschten Rollen berechnet. Erlaubt auch diese Schranke kein Ausschließen des Kandidaten, wird die DTW-Berechnung ausgeführt, wieder mit Early Abandoning. Da $LB_{keogh}(q, c)$ berechnet wurde, kann der noch nicht ausgeführte Teil der Berechnung abgeschätzt werden, die DTW-Berechnung kann also möglicherweise noch früher abgebrochen werden. In Algorithmus 1 wurde die Query-Funktion in Pseudocode notiert.

Für die Suche nach ähnlichsten Subsequenzen in Zeitreihen konnten Keogh et al. in [10] durch die Verwendung der UCR-Suite enorme Geschwindigkeitsgewinne erzielen, von denen auch in der Anwendung beim Clustern von Zeitreihen profitiert werden kann:

Algorithmus 1 Der Query Algorithmus in Pseudocode

```
function query(C, q):
    bestSoFar ← ∞
    best ← null
    u, l ← [ ]
    for c ∈ C :
        if LB_Kim'(c, q) < bestSoFar:
            if not |u| == |c|:
                Berechne u = u^(q) und l = l^(q)
            if LB_keogh(c, q, bestSoFar) < bestSoFar:
                Berechne u^(c) und l^(c)
                if LB_keogh(q, c, bestSoFar) < bestSoFar:
                    d ← d_dtw(c, q, bestSoFar)
                    if d < bestSoFar:
                        bestSoFar ← d
                        best ← c
    return (best,bestSoFar)
```

> We demonstrate the following extremely unintuitive fact; in large datasets we can exactly search under DTW much more quickly than the current state-of-the-art Euclidean distance search algorithms. [...] We show that our ideas allow us to solve higher-level time series data mining problem such as motif discovery and clustering at scales that would otherwise be untenable.

Der Geschwindigkeitsgewinn lässt sich durch das geschickte Vermeiden unnötiger DTW-Berechnungen erklären:

> Using the ideas developed in this work, the vast majority of potential DTW calculations are pruned with $\mathcal{O}(1)$ work, while some require up to $\mathcal{O}(n)$ work, and only a vanishingly small fraction require $\mathcal{O}(nR)$ work. The weighted average of these possibilities is less than $\mathcal{O}(n)$.

In dieser Arbeit wird nicht nach Subsequenzen gesucht; einige der Optimierungen der UCR-Suite können also nicht angewandt werden: Die mithilfe eines beweglichen Fensters optimierte z-Normalisierung und Berechnung der LB_{keogh} ist nur bei der Suche nach Subsequenzen möglich, da bei der Suche nach ganzen Sequenzen nur ein Fenster, die ganze Zeitreihe, betrachtet wird. Die vorgestellten Techniken werden in Abschnitt 4.1 auf den

k-means Clusteringalgorithmus angewant; inwieweit die Optimierungen helfen unnötige DTW-Berechnungen im Anwendungsfall k-means Clustering zu vermeiden, soll in Abschnitt 6.2.3 experimentell untersucht werden.

Kapitel 4

k-means Clustering von Zeitreihen

Im Folgenden soll der k-means Algorithmus aus Kapitel 2 für das Clustering von Zeitreihen mit DTW als Abstandsmaß modifiziert werden. Dazu muss, wie schon am Beispiel des k-means Clustering von reellwertigen Vektoren mit quadrierter euklidischer Distanz gesehen, insbesondere eine mit DTW kompatible Technik zum Mitteln einer Menge von Zeitreihen gefunden werden. Außerdem sollen die in Kapitel 3 vorgestellten Ansätze Keoghs zum effizienteren Umgang mit Querys bei der Zuweisung der Zeitreihen auf die nächsten Zentroide Verwendung finden.

4.1 Zuweisung der Zeitreihen auf die Zentroide

Die Zuweisung einer Zeitreihe x zum bezüglich d_{dtw} nächsten Zentroid $\arg\min_{z_i} d_{dtw}(x, z_i)$ kann wieder durch die Query Funktion ausgeführt werden. Für jede Zeitreihe x der zu clusternden Menge wird aus der Menge der aktuellen Zentroide $z_1, ..., z_k$ dasjenige gesucht, dessen Abstand $d_{dtw}(z_i, x)$ minimal ist. Im besten Fall muss also mithilfe der Optimierungen von Keogh statt k nur eine Abstandsberechnung ausgeführt werden.

Dabei kann vermutet werden, dass der Effekt der Optimierungen besonders für große k sichtbar wird. Dies wird in Abschnitt 6.2.3 experimentell untersucht.

4.2 Ermitteln neuer Zentroide

Das Ermitteln neuer, mit DTW kompatibler Zentroide war im Vorfeld dieser Arbeit die größte Unbekannte. Im diesem Unterkapitel sollen verschiedene Ansätze dazu präsentiert

werden. Dabei stelle ich zunächst den naiven Ansatz vor, aus dem ich eine eigene Idee, die Mittelung durch Projektion, ableite. Anschließend präsentiere ich eine weitere eigene Idee, die Mittelung durch Fixpunktiteration, von der sich jedoch herausstellen wird, dass sie eng mit der Mittelung durch Projektion verwandt ist. Es existieren diverse Publikationen zur Mittelung von Zeitreihen mit DTW [7, 6, 14]; ich halte den jüngsten, in [14] von Ratanamahatana et al. vorgestellten Ansatz des Shapebased Averaging für den vielversprechendsten und präsentiere ihn daher ausführlich. Schließlich stelle ich noch einen Ansatz vor, der sich immer dann bewährt hat, wenn die Mittelung vermieden werden soll, das sogenannte k-medoids. Die verschiedenen Ansätze werden in Kapitel 6 experimentell miteinander verglichen.

4.2.1 Euklidische Mittelung

Ein naiver Ansatz, eine Menge von gleichlanger Zeitreihen zu mitteln ist, sie, wie Vektoren aus dem $\mathbb{R}^n$, arithmetisch zu mitteln:

$$average_{euklid}(C) = \frac{1}{|C|} \sum_{x \in C} x.$$

Auf diese Art und Weise kann eine Mittelung in $\mathcal{O}(n \cdot |C|)$ berechnet werden, wobei n die Länge der Zeitreihen ist. Diese geringe Komplexität muss als Vorteil hervorgehoben werden.

Man spricht von euklidischer Mittelung, da das Mittelungsverfahren verwendet wird, das bezüglich des euklidischen Abstands optimale Zentroide berechnet. Dennoch wird im k-means Algorithmus für die Zuteilung der Zeitreihen auf die nächsten Zentroide das mächtigere DTW Abstandsmaß verwendet. Das ist problematisch; selbst mit gleichlangen Zeitreihen berechnet die Euklidische Mittelung Zentroide, die mit DTW inkompatibel scheinen (Siehe Abbildung 4.1.a). Somit kann also auch keine Konvergenz garantiert werden und die lokale Optimalität einer eventuellen gefundenen Lösung kann auch nicht garantiert werden.

Im Experimentalteil dieser Arbeit soll untersucht werden, ob das Verfahren in der Praxis dennoch akzeptable Ergebnisse liefert.

4.2.2 Mittelung durch Projektion auf alte Zentroide

In Abbildung 4.1.b) ist ersichtlich, dass die naive Mittelung zu funktionieren scheint, wenn die Zeitreihen eine einheitliche Form haben. Wenn man also alle Zeitreihen in eine

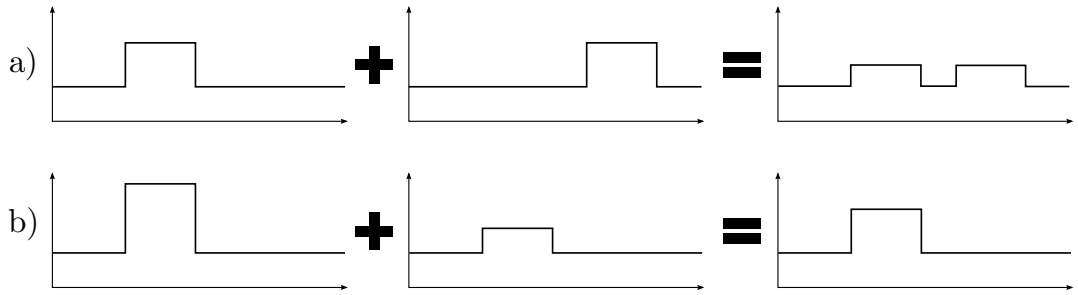


Abbildung 4.1: Während die euklidische Mittelung in Fall a) eine Zeitreihe berechnet, die nicht mehr die gleiche Form hat wie die beiden ursprünglichen Zeitreihen, das Zentroid also mit DTW inkompatibel ist, liefert es im Fall b) mit formgleichen Zeitreihen ein sinnvolles Ergebnis.

einheitliche Form transformieren könnte, ließen sie sich naiv mitteln.

In der t-ten Iteration des k-means Algorithmus haben alle Zeitreihen eines Clusters $C^{(t)}$ nach bisher bestem Kenntnisstand die Form des Zentroids $z^{(t)}$ gemein.

Anhand des Warping-Pfades $p_{x,z}$ von $x \in C$ und z kann x nun auf das Zentroid z projiziert werden:

$$x'_i = \frac{1}{|\{x_a \mid (a,i) \in p^*_{x,z}\}|} \sum_{(a,i) \in p^*_{x,z}} x_a \tag{4.1}$$

Das Element x'_i einer Zeitreihe setzt sich also aus allen Elementen der Zeitreihe x_a zusammen, die mit z_i aligniert werden. Wird z_i dabei mit mehr als einem Element aligniert, bildet sich x'_i aus dem arithmetischen Mittel dieser Elemente. Ein Beispiel einer solchen Projektion findet sich in Abbildung 4.2.

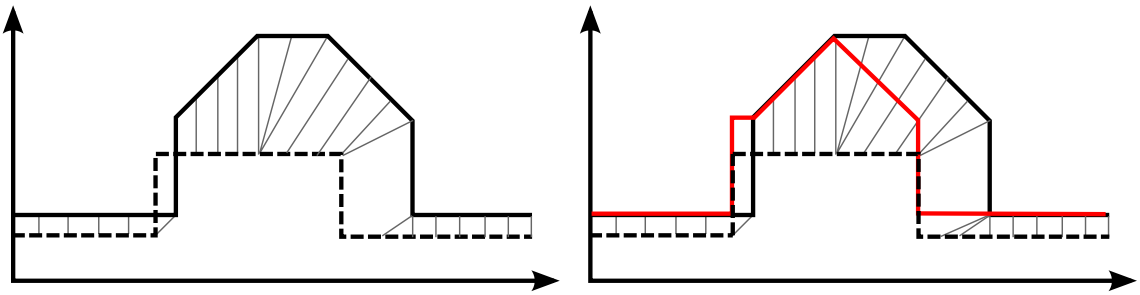


Abbildung 4.2: Die durchgezogen gezeichnete Zeitreihe wird auf die gestrichelte Zeitreihe projiziert anhand des durch dünne Linien angedeuteten Alignments. Das Ergebnis dieser Projektion ist rechts in rot gezeichnet.

Die so entstehenden Zeitreihen x' haben alle die selbe Länge, die des alten Zentroids z, und können demnach mit der euklidischen Mittelung zu einem neuen Zentroid zusammen-

gefasst werden. Da sie außerdem alle in etwa die selbe Form wie z haben, vermute ich, dass die so entstehenden Zentroide kompatibel mit DTW sind. Dies wird durch die im nächsten Abschnitt gewonnen Erkenntnisse deutlicher, außerdem wird es in Kapitel 6 durch eine empirische Untersuchung gezeigt.

Da die DTW-Matrizen im Zuge der Zuweisung der Elemente auf Cluster von Schritt II des k-means Algorithmus ohnehin berechnet werden, können die Warping-Pfade ohne zusätzliche DTW-Berechnungen und somit effizient berechnet werden. Somit ist das Laufzeitverhalten der Mittelung durch Projektion nicht signifikant schlechter als bei der euklidischen Mittelung; eine Projektion kann in $\mathcal{O}(L)$ berechnet werden, die Zeitreihen x' können dann in $\mathcal{O}(n \cdot |C|)$ gemittelt werden, wobei n die Länge des alten Zentroids z ist.

4.2.3 Mittelung durch Fixpunktiteration

Im Beweis von Satz 1 auf Seite 10 können wir sehen, dass ein optimales Mittelungsverfahren für eine Menge von Vektoren direkt durch die Minimierung der Kostenfunktion bei fester Verteilung auf die Cluster, also durch Nullsetzen der abgeleiteten Kostenfunktion, gegeben ist. Diese Idee soll nun auf das Mitteln einer Menge von Zeitreihen bezüglich des quadrierten DTW-Abstands übertragen werden.

Dabei wird das Problem zunächst etwas beschränkt; statt zu einer gegebenen Menge von Zeitreihen die Zeitreihe zu suchen, die die Kostenfunktion minimiert, soll nur die optimale Zeitreihe einer vorgegebenen Länge n gesucht werden. Eine geeignete Wahl von n, beispielsweise der Median oder Mittelwert aller vorhandenen Längen, sollte zu einer guten Approximation der optimalen Lösung führen. Somit sollte sich diese Einschränkung nicht als problematisch erweisen.

Jetzt bleibt zu klären, ob und wie die Kostenfunktion

$$cost(C, z) = \sum_{x \in C} d_{dtw}(x, z)$$

abgeleitet werden kann.

Da nur eine optimale Zeitreihe der Länge n gesucht ist, hat z die Länge n und kann wie ein Vektor aus dem $\mathbb{R}^n$ behandelt werden. Die Kostenfunktion $cost(C, z)$ könnte also komponentenweise partiell abgeleitet werden. Allerdings ist d_{dtw} an Stellen, an denen es zwei verschiedene, optimale Warping-Pfade gibt, im Allgemeinen nicht nach z_i ableitbar. Ich ignoriere diesen seltenen Fall und leite d_{dtw} mit den üblichen Mitteln der Differenzialrech-

nung partiell ab:

$$\frac{\partial}{\partial z_i} d_{dtw}(x,z) = \frac{\partial}{\partial z_i} \sum_{(a,b)\in p^*_{x,z}} (x_a - z_b)^2 = \sum_{(a,i)\in p^*_{x,z}} (2x_a - 2z_i).$$

Nun kann die Kostenfunktion bei festem Cluster C auf lokale Minima untersucht werden, denn es gilt

$$\frac{\partial}{\partial z_i} cost(C,z) = \frac{\partial}{\partial z_i} \sum_{x\in C} d_{dtw}(x,z) = \sum_{x\in C} \sum_{(a,i)\in p^*_{x,z}} (2x_a - 2z_i).$$

Durch Nullsetzen der Ableitung kann die Kostenfunktion, eine quadratische Funktion, auf lokale Minima untersucht werden. Man erhält

$$z_i = \frac{1}{|\{x_a \mid \exists x \in C \text{ mit } (a,i) \in p^*_{x,z}\}|} \sum_{x\in C} \sum_{(a,i)\in p^*_{x,z}} x_a.$$

Dies ist natürlich keine geschlossene Form des Minima, da z_i sowohl explizit auf der linken als auch auf implizit in der Berechnung optimaler Warping-Pfade auf der rechten Seite des Gleichheitszeichens steht.

Allerdings lässt sich die Gleichung als Funktionsvorschrift

$$\pi_i(z) = \frac{1}{|\{x_a \mid \exists x \in C \text{ mit } (a,i) \in p^*_{x,z}\}|} \sum_{x\in C} \sum_{(a,i)\in p^*_{x,z}} x_a$$

einer Fixpunktiteration

$$z^{(t+1)} = \pi(z^{(t)})$$

verwenden und es gilt:

Satz 4 (Konvergenz der Fixpunktiteration)**.** Die Fixpunktiteration $z^{(t+1)} = \pi(z^{(t)})$ konvergiert in einem lokales Minimum der Kostenfunktion $z \mapsto cost(C,z)$.

Beweis. Zunächst zeigen wir, dass $cost(C,\pi(z)) \leq cost(C,z)$. Es gilt:

$$\begin{aligned} cost(C,\pi(z)) &= \sum_{x\in C} d_{dtw}(x,\pi(z)) \\ &= \sum_{x\in C} \sum_{(a,b)\in p^*_{x,\pi(z)}} (x_a - \pi_b(z))^2 \end{aligned}$$

Die optimalen Warpingpfade verursachen niedrigere Kosten als beliebige Warpingpfade, also insbesondere als die optimalen Warping-Pfade der letzten Iteration:

$$\leq \sum_{x\in C} \sum_{(a,b)\in p^*_{x,z}} (x_a - \pi_b(z))^2$$

Für die Berechnung von $\pi(z)$ bestimmt z nur eine Menge von Warping-Pfaden, hat aber sonst keinen weiteren Einfluss auf das Resultat. Für diese durch z gegebene Menge von Warping-Pfaden berechnet π, nach Konstruktion, die Lösung, die die Kosten minimiert. Aus der Minimalität folgt:

$$\begin{aligned} &\leq \sum_{x \in C} \sum_{(a,b) \in p^*_{x,z}} (x_a - z_b)^2 \\ &= cost(C, z) \end{aligned}$$

Somit sinken die Gesamtkosten mit jeder Iteration von π. Daraus folgt, dass nach einer Verbesserung die alte, durch z beschriebene Menge von Warping-Pfaden nie wieder verwendet wird, da eine bessere Lösung als das Optimum für diese Menge gefunden wurde. Da es nur eine endliche Menge von Warping-Pfaden zwischen zwei Zeitreihen gibt und somit auch nur eine endliche Menge von Warpingpfad-Mengen, muss in endlich vielen Schritten ein optimaler Wert angenommen werden. □

Satz 5 (Konvergenz von k-means)**.** Für eine Menge von Zeitreihen T und ein gegebenes $k \in \mathbb{N}$ konvergiert der k-means Algorithmus mit Distanzmaß d_{dtw} und der oben definierten Mittelung durch Fixpunktiteration in einem lokalen Optimum.

Beweis. Es wurde gezeigt, dass bei der Mittelung durch Fixpunktiteration die Kosten des Zentroids in jedem Iterationsschritt abnehmen. Startet man die Fixpunktiteration also im alten Zentroid des k-means Algorithmus, wird ein neues Zentroid berechnet, welches niedrigere Kosten verursacht.

Somit kann analog zum Beweis von Satz 1 argumentiert werden, dass mit jeder Iteration des k-means Algorithmus die Gesamtkosten des Clusterings abnehmen und der Algorithmus somit terminieren muss.

Da die Mittelung durch Fixpunktiteration in einem lokalen Minimum konvergiert, konvergiert auch der k-means Algorithmus in einem lokalen Minimum. □

Führt man die Fixpunktmittelung auf Beispieldaten aus, stellt man fest, dass die Kostenfunktion zunächst sehr schnell minimiert wird, aber schließlich nur sehr langsam gegen das Minimum konvergiert. Deshalb bietet es sich für praktische Anwendungen an, die Iteration nur solange auszuführen, wie die relative Verbesserung $\frac{cost(C,z^{(t+1)})}{cost(C,z^{(t)})}$ kleiner einer gegebenen Genauigkeit $0 < \varepsilon \leq 1$ ist. Beispielsweise für $\varepsilon := 0.95$ sind die so approximierten Zentroide immer noch ausreichend optimal, die Anzahl der Iterationen wird aber häufig um Faktor 10 oder mehr verringert.

Betrachten wir nochmals den Ansatz der Mittelung durch Projektion auf alte Zentroide, stellen wir fest, dass die Funktionsvorschrift der Projektion in Gleichung 4.1 mit der Funktionsvorschrift der Fixpunkt-Iteration π übereinstimmt, wenngleich sie anders motiviert wurde. Die Projektion stellt also mathematisch eine andere Variante dar, mit der ein tatsächliches Minimum der Kostenfunktion approximiert werden kann. Konvergenz des k-means Algorithmus mit Mittelung durch Projektion kann analog zu Satz 5 gezeigt werden; allerdings ist die Lösung im Allgemeinen nicht lokal optimal.

4.2.4 Shape-based Template Matching Framework

In [16] präsentieren Ratanamahatana et al. einen Ansatz, der es zunächst erlaubt, zwei Zeitreihen mithilfe ihres gemeinsamen Warping-Pfades zu mitteln. Gegeben den Warping-Path $p_{q,c}$ der Länge L zweier Zeitreihen q und c ergibt sich eine neue Liste mit L Zeit-Wert Paaren (t_k, v_k) wie folgt:

$$t_k = \frac{i+j}{2}, v_k = \frac{q_i + c_j}{2}, \text{ wobei } (i,j) \text{ das } k\text{-te Element von } p_{q,c} \text{ ist.}$$

Diese Liste von Zeit-Wert Paaren kann dann abgetastet bzw. resampled werden, sodass durch Interpolation eine Zeitreihe im ursprünglichen Sinn berechnet wird, also mit gleichmäßig verteilten Messpunkten im Abstand von einer Zeiteinheit. Siehe dazu Abbildung 4.3.

Der Ansatz wird noch verallgemeinert, statt zwei Zeitreihen gleichberechtigt zu mitteln,

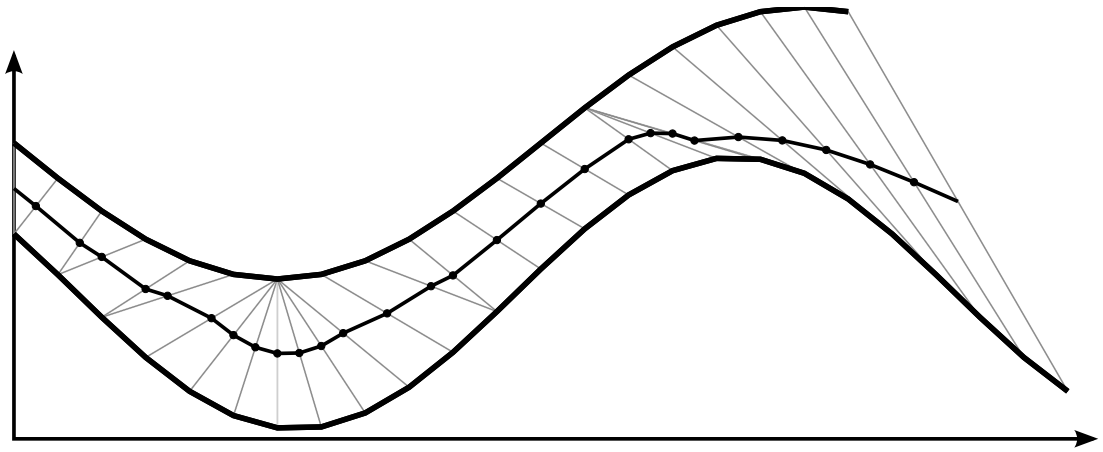


Abbildung 4.3: Das Mitteln zweier Zeitreihen nach Ratanamahatana. Man sieht, dass die gemittelte Zeitreihe (dünner eingezeichnet) durch den hellgrau eingezeichneten Warping-Pfad definiert wird und zunächst aus Messdaten besteht, die auch zwischen zwei ganzzahligen Zeitpunkten liegen können. Durch Interpolation kann die Zeitreihe aber in die gewohnte Repräsentation überführt werden.

werden den Zeitreihen q, c Gewichte $w_q, w_c > 0$ zugeteilt, um deren Einfluss an der gemittelten Zeitreihe zu steuern:

$t_k = \frac{w_q \cdot i + w_c \cdot j}{w_q + w_c}, v_k = \frac{w_q \cdot q_i + w_c \cdot c_j}{w_q + w_c}$, wobei (i, j) das k-te Element von $p_{q,c}$ ist.

Eine Menge von Zeitreihen kann nun gemittelt werden, in dem solange zwei Zeitreihen zu einer gemittelten zusammengefasst werden, bis nur noch eine Zeitreihe übrig ist. Allerdings ist die Mittelung nicht assoziativ, somit spielt die Reihenfolge eine Rolle, in der die Zeitreihen gemittelt werden.

In [14] entwickeln die Autoren die Idee, immer die zwei ähnlichsten Zeitreihen der Menge gewichtet zu mitteln und die so entstehende Zeitreihe der Menge wieder hinzuzufügen. (Siehe Algorithmus 2)

Problematisch daran ist, dass ohne Optimierungen $\mathcal{O}(|C|^3)$ DTW-Berechnungen durchgeführt werden müssen, und selbst wenn die Optimierungen von Keogh auf das Problem übertragen werden, ist die Laufzeit dennoch hoch (Siehe Abschnitt 6.1). Außerdem demonstrieren die Autoren in [16], dass k-means mit der vorgestellten Mittelung häufig degeneriert, also weniger als k Cluster findet. Daraus kann man schließen, dass die Mittelung nicht die Kosten eines Clusters minimiert, da in einem Cluster nach dem Mittelungsschritt alle Zeitreihen in ein anderes Cluster wechseln.

Im Vergleich zur Mittelung durch Fixpunktiteration fällt positiv auf, dass alle Zeitreihen des Clusters für die Form des neuen Zentroids zunächst gleich wichtig sind, wohingegen die Form des Zentroids beim Mitteln durch Fixpunktiteration stark vom alten Zentroid, dem Ausgangspunkt der Fixpunktiteration, abhängt und somit das insgesamt berechnete Clustering sehr stark von den initialen Zentroiden abhängt. In Abschnitt 4.3 wird deshalb noch eine weitere Methode zur Wahl initialer Zentroide, das k-means++, vorgestellt.

In [14] präsentieren die selben Autoren einen Ansatz, der die Laufzeit durch Approximieren der DTW-Berechnungen um bis zu Faktor 400 beschleunigen soll.

Dazu wird angenommen, dass DTW die Dreiecksungleichung erfüllt. Dies ist eigentlich nicht der Fall (siehe [1]), allerdings erfüllen unter Verwendung von Global Constraints auf echten Datensätzen nur sehr wenig Tripel von Zeitreihen die Dreiecksungleichung nicht. Somit erhält man eine schöne Approximierung der Distanz zweier Zeitreihen über die Abstände der beiden Zeitreihen zu einem dritten Punkt. Dazu betrachten wir:

$$d_{dtw}(p, z) \leq d_{dtw}(p, q) + d_{dtw}(q, z) \Leftrightarrow d_{dtw}(p, z) - d_{dtw}(z, q) \leq d_{dtw}(p, q)$$

Algorithmus 2 Der Shape Based Averaging Algorithmus in Pseudocode

function $shapeBasedAverage(C)$:
  for $c \in C$:
    $w_c \leftarrow 1$
  while $|C| > 1$:
    $(q, c) \leftarrow query(C)$ //die zwei sich ähhnlichsten Zeitreihen der Menge abfragen
    $(t, v) \leftarrow average(q, c, w_q, w_c)$ //nach obiger Gleichung mitteln
    $z \leftarrow resample(t, v)$ //und in die gewünschte Form bringen
    $w_z \leftarrow w_q + w_c$
    $C \leftarrow \{z\} \cup (C \setminus \{q, c\})$
  return c_1

$$d_{dtw}(q, z) \leq d_{dtw}(q, p) + d_{dtw}(p, z) \Leftrightarrow d_{dtw}(q, z) - d_{dtw}(z, p) \leq d_{dtw}(p, q)$$

Aus den beiden Gleichungen folgt, dass $|d_{dtw}(q, z) - d_{dtw}(z, p)| \leq d_{dtw}(p, q)$, diese untere Schranke wird nun ausgenutzt, um den Abstand zweier Zeitreihen zu approximieren.

Im k-means Algorithmus werden für die Neuzuweisung der Zeitreihen der Menge M auf die Cluster 1-k die Abstände aller Zeitreihen zu den Zentroiden berechnet. Somit kann der Abstand zweier Zeitreihen aus M ohne neue DTW-Berechnungen wie folgt approximiert werden:

$$d_{approx}(p, q) = \max_{1 \leq i \leq k} |d_{dtw}(q, z_i) - d_{dtw}(z_i, p)|.$$

Die Optimierungen von Keogh erlauben es allerdings, dass im besten Fall nur der DTW-Abstand der Zeitreihe und dem zugehörigen Zentroid bekannt ist, also nur 1 statt k Abstände. Also werde ich mich mit einer schlechteren Approximation begnügen:

$$d_{approx'}(p, q) = |d_{dtw}(q, z_q) - d_{dtw}(z_p, p)|, \text{ wobei } z_p = z_q = \arg\min_{1 \leq i \leq k} d_{dtw}(z_i, x) \ .$$

In Algorithmus 3 ist der sich somit ergebene Algorithmus nochmals aufgeführt. Dieser ergibt sich aus dem Shapebased Averaging Algorithmus, mit dem Unterschied, dass die DTW-Berechnungen in der Suche nach den zwei sich ähnlichsten Zeitreihen durch $d_{approx'}$ ersetzt werden. Zu jeder im Verlauf des Algorithmus neu erstellten Zeitreihe z muss nun noch der Abstand zum alten Zentroid der Menge C berechnet werden, damit die Approximation $d_{approx'}$ auch für die neue Zeitreihe z bestimmt werden kann. Somit ergeben sich also im Verlauf des Algorithmus nur $2 \cdot |C|$ DTW-Berechnungen.

Auf meine Anfrage hin versicherte mir einer der Autoren, dass die Änderungen auch das Problem der degenerierter Clusterings gelöst haben, die Approximation des Algorithmus

Algorithmus 3 Der Fast Shape Based Averaging Algorithmus in Pseudocode

function $fastShapeBasedAverage(C)$:
  z_C := altes Zentroid von C
  $d_x := d_{dtw}(x, z_C) \; \forall \; x \in C$ // Werte bekannt aus k-means
  for $c \in C$:
    $w_c \leftarrow 1$
  while $|C| > 1$:
    // die zwei sich unter $d_{approx'}$ ähhnlichsten Zeitreihen der Menge abfragen
    $(q, c) \leftarrow queryApprox(C)$
    // wie gehabt die beiden ähnlichsten Zeitreihen mitteln
    $(t, v) \leftarrow average(q, c, w_q, w_c)$ // berechnet optimalen Warping-Pfad von q und c
    $z \leftarrow resample(t, v)$
    $w_z \leftarrow w_q + w_c$
    // Abstand der neuen Zeitreihe zum alten Zentroid bestimmen
    $d_z \leftarrow d_{dtw}(z, z_C)$
    $C \leftarrow \{z\} \cup (C \setminus \{q, c\})$
  // das einzige Element von C ist das gesuchte Zentroid
  return c_1

also besser funktioniere als der ursprüngliche Algorithmus. Es stellt sich allerdings im Experimentalteil heraus, dass zumindest bei Anwenden der stärkeren Approximation über nur ein Zentroid das Problem nicht vollständig gelöst wurde.

4.2.5 k-medoids

Ein bewährter Ansatz, das Mitteln von Zeitreihen zu vermeiden, ist stattdessen das Medoid einer Menge C von Zeitreihen zu bestimmen [5]. Das Medoid ist die Zeitreihe, die die größte Ähnlichkeit mit allen anderen Zeitreihen der Menge hat, also die Zeitreihe

$$medoid(C) = \arg\min_{x \in C} \sum_{z \in C} d_{dtw}(x, z).$$

Somit vermeidet man also das Problem, neue Zeitreihen generieren zu müssen, die die Kostenfunktion der Cluster minimieren und verwendet nur die Zeitreihen aus der Eingabemenge als mögliche Zentroide. Das bedeutet auch, dass nur DTW-Berechnungen zwischen den Eingabe-Zeitreihen ausgeführt werden müssen, diese können also einmalig ausgeführt werden und in einer Abstandsmatrix der Größe $|C|^2$ gespeichert werden.

Da beim Bestimmen der Medoide nicht im eigentlichen Sinne gemittelt wird, spricht man nicht mehr von k-means Clustering, sondern von k-medoids Clustering.

4.3 Wahl initialer Zentroide

Wir haben gesehen, dass die Mittelung durch Projektion und die Mittelung durch Fixpunktiteration das alte Zentroid als Ausgangspunkt benötigen, um neue Zentroide zu berechnen. So hängt die Qualität der Zentroide stark von der Qualität der alten Zentroide ab. Insgesamt sind also gute initiale Zentroide sehr wichtig für die Qualität des berechneten Clusterings. Daher liegt es nahe, die ersten Zentroide nicht komplett zufällig zu wählen, sondern geschickter vorzugehen.

In [2] wird eine Variante der Startwertinitialisierung präsentiert, die initiale Zentroide zufällig so auswählt, dass sie möglichst weit auseinander liegen. Die Idee dabei ist, dass weit auseinander liegende Zentroide wahrscheinlich nicht zum selben Cluster gehören und somit die Anzahl der benötigten Iterationen reduziert werden könnten.

Die Auswahl der initialen Zentroide z_k aus einer Menge M verläuft in den in Algorithmus 4 aufgeführten Schritten; die Wahrscheinlichkeit eine Zeitreihe zufällig als Zentroid zu ziehen

ist proportional zum kleinsten Abstand der Zeitreihe zu den schon gezogenen Zentroiden.

Algorithmus 4 Die k-means++ Initialisierung

Wähle z_1 zufällig gleichverteilt aus M

for $i \in 2, \ldots, k$:

Berechne $d_x = \min_{j=1,\ldots,i-1} d_{dtw}(x, z_j) \ \forall \ x \in M$

Wähle z_i zufällig, dabei wird x aus M mit Wahrscheinlichkeit $\frac{d_x}{\sum_{z \in M} d_z}$ gezogen.

return $\{z_1, ..., z_k\}$

Beachte, dass für die Berechnung von d_t wieder die Optimierungen von Keogh et al. angewandt werden können. Das Suchen des Zentroids, das unter allen schon ausgewählten den kleinsten Abstand zu einer Zeitreihe t hat, entspricht dem Ausführen eines Querys mit der Menge $\{z_1, \ldots, z_i\}$ und einer Query-Zeitreihe t.

Für den k-means Algorithmus auf $\mathbb{R}^n$ mit quadrierter euklidischer Distanz können die Autoren zeigen, dass die berechneten Gesamtkosten des Algorithmus mit k-means++ Startwertinitialisierung im Erwartungswert nur um einen Faktor logarithmisch in k schlechter sind als die optimale Lösung. Eine solche garantierte Qualität der Lösung kann für k-means mit zufälliger Wahl der initialen Zentroide nicht gegeben werden.

Zwar gelten die in [2] gezeigten Garantien für die gefundene Lösung des k-means++ Clusterings nur für den $\mathbb{R}^n$ mit quadrierter euklidischer Distanz, die Idee, auch Zeitreihen so zu wählen, dass sie möglichst weit auseinander liegen, scheint aber nichtsdestotrotz vernünftig und wird deshalb auch in Abschnitt 6.2.1 experimentell untersucht.

Kapitel 5

Implementierung

Im Rahmen dieser Bachelorarbeit wurden die vorgestellten Ideen zum k-means Clustering von Zeitreihen auch als RapidMiner-Operator implementiert. Die RapidMiner-Extension ist unter `https://bitbucket.org/Whadup/dtw-clustering/` frei verfügbar. Die DTW-Clustering-Extension benötigt die RapidMiner-eigene ValueSeries-Extension, denn als Repräsentation der Zeitreihen wird die dortige Implementierung ValueSeries verwendet. Eine Menge von Zeitreihen wird als IOObjectCollection von ValueSeries repräsentiert. Dies hat gegenüber einer Speicherung der Zeitreihen in einem ExampleSet den Vorteil, dass die einzelnen Zeitreihen unterschiedliche Längen haben können. Außerdem können so leichter die in der ValueSeries-Extension bereitgestellten Operatoren für Zeitreihen verwendet werden.

Die wichtigste Komponente der DTW-Clustering-Extension ist dabei der k-Means Clustering Operator, der das k-Means Clustering von Zeitreihen implementiert. Dieser hat folgende Funktionen:

- Für eine Collection von ValueSeries wird ein Clustering berechnet. Die berechnete Clusterzugehörigkeit wird in einem Feature 'cluster' der ValueSeries gespeichert. Der Operator gibt die so gelabelten Zeitreihen sowie die Menge der Zentroide zurück.
- Über Parameter lassen sich k sowie die maximale Anzahl Iterationen einstellen. Außerdem kann ein Sakoe-Chiba-Band und dessen Breite eingestellt werden und die k-means++ Initialisierung ausgewählt werden.
- Über einen Parameter lassen sich alle vorgestellten Mittelungsverfahren auswählen. Die Vorauswahl steht auf der Mittelung durch Projektion.
- Wie in RapidMiner üblich kann ein Random Seed eingestellt werden, der Operator

verwendet den Zufallszahlen Generator aus RapidMiner.

- Der Operator kann über den Log-Operator einige interessante Werte protokollieren, unter anderem die berechneten Clustergesamtkosten und die benötigten DTW-Berechnungen.
- Der Operator verwendet alle verfügbaren CPUs des Rechners. Dabei findet Parallelisierung an zwei Punkten statt: Die Zuteilung der Zeitreihen auf die Zentroide findet parallel statt; die Eingabemenge wird auf die verfügbaren Prozessoren verteilt, somit werden parallel Querys ausgeführt. Außerdem wird die Mittelung parallelisiert, jedes Cluster wird in einem anderen Thread gemittelt.

Neben dem k-means Operator wurden einige Operatoren implementiert, die den Umgang mit Mengen von Zeitreihen in RapidMiner vereinfachen sollen:

- **ExampleSet To ValueSeriesSet:** Wandelt jedes Example eines ExampleSets in eine reellwertige ValueSeries um. Dabei wird der erste Missing Value eines Examples als Ende der ValueSeries aufgefasst. Liefert eine Collection von ValueSeries zurück.
- **ValueSeriesSet To ExampleSet:** Erzeugt aus einer Collection von ValueSeries ein ExampleSet, in dem jede Zeitreihe als ein Example gespeichert wird. Jedes in den ValueSeries gespeicherte Feature wird zu einem reellwertigen Special Attribute.
- **ValueSeriesSet Features To ExampleSet:** Speichert nur die in den ValueSeries gespeicherten Features in einem ExampleSet.
- **Z-Normalization:** z-normalisiert jede ValueSeries einer Collection von ValueSeries'
- **One-Nearest-Neighbor:** Lernt aus einer gegebenen Trainingsmenge ein Modell zur 1-NN Klassifizierung. Dabei werden die mit den Methoden von Keogh beschleunigten Querys verwendet.
- **Apply ValueSeries Model:** Wendet ein Modell zur Klassifikation von Zeitreihen auf eine Menge von ValueSeries an. Falls später noch andere Klassifizierer außer dem 1-NN implementiert werden, kann das dem 1-NN zugrunde liegende Interface für Modelle verwendet werden.
- **Compute Zentroid:** Berechnet ein Zentroid einer gegebenen Menge von Zeitreihen. Dabei kann das Mittelungsverfahren über einen Parameter eingestellt werden.

Kapitel 6

Experimente

Nun sollen die in Kapitel 4 vorgestellten Ideen evaluiert werden. Dabei wird der Fokus auf dem Vergleich der verschiedenen Mittelungsverfahren liegen.

6.1 Vergleich der verschiedenen Mittelungsverfahren

Zunächst sollen die verschiedenen Mittelungsverfahren unabhängig von k-means getestet werden. Dafür müssen allerdings erst Kriterien festgelegt werden, was ein gutes Mittelungsverfahren ausmacht. Das sollen erstens objektive Kriterien sein: Aus mathematischer Sicht minimiert k-means eine Kostenfunktion, folglich sollte auch das Mittelungsverfahren die Kosten eines Clusters minimieren. Somit sollen die Kosten des berechneten Zentroids verglichen werden.

Das Mittelungsverfahren sollte weiterhin möglichst effizient sein, also möglichst wenig Rechenzeit benötigen. Ein schönes Kriterium zum Vergleich von Laufzeiten ist der Vergleich der Anzahlen durchgeführter DTW-Berechnungen, da diese den größten Anteil an der Laufzeit haben und das Zählen der Aufrufe die Laufzeit von der konkreten Implementierung und Rechnerarchitektur abstrahiert.

Zweitens sollten die ermittelten Zentroide auch subjektiv beurteilt werden um zu beurteilen, inwieweit das Zentroid eine gute Zusammenfassung aller Zeitreihen darstellt.

Im Verlauf des k-means Algorithmus müssen die Mittelungsverfahren nicht nur Mengen gleichgeformter Zeitreihen mitteln, sondern, insbesondere in frühen Iterationen, auch Mengen von Zeitreihen mitteln, die sich in ihrer Form stark unterscheiden. Es ist wichtig, dass das Mittelungsverfahren so robust ist, dass es auch für fälschlicherweise zu einem Clus-

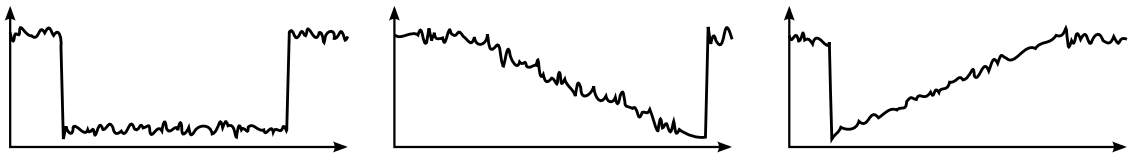


Abbildung 6.1: Je eine Zeitreihe der Klassen Cylinder, Bell und Funnel.

ter zusammengefasste und somit potentiell sehr unterschiedliche Zeitreihen gut funktioniert. Beispielsweise sollte die Laufzeit bei nicht idealen Eingabemengen nicht wesentlich erhöhen. Folglich sollten Mittelungsverfahren nicht nur unter idealen Bedingungen getestet werden.

Unter diesen Gesichtspunkten ist das folgende Experiment entwickelt worden.

Für das Experiment wird der Cylinder-Bell-Funnel Datensatz verwendet. Dabei handelt es sich um einen Zufallsgenerator für Zeitreihen der Länge 128, der erstmals in [19] vorgestellt wurde und Zeitreihen aus drei Klassen, Cylinder, Bell und Funnel, generiert.
Zunächst werden aus dem Cylinder-Bell-Funnel Datensatz aus jeder Klasse 100 Zeitreihen gezogen. Diese idealen Cluster sollen von allen echten Mittelungsverfahren gemittelt werden. Anschließend wird das Experiment wiederholt, statt mit 100 Zeitreihen aus einer Klasse jedoch mit 80 Zeitreihen aus einer Klasse und je 10 aus den beiden anderen Klassen.

Die Verfahren zur Mittelung via Projektion und Fixpunktiteration sowie das Fast Shapebased Averaging benötigen zur Mittelung einer Menge von Zeitreihen, wie in Kapitel 4 gesehen, ein altes Zentroid. Um zu einer Einschätzung zu gelangen, wie wichtig die initiale Wahl für die Qualität des berechneten Zentroids ist, wird jede Zeitreihe der Menge einmal als altes Zentroid verwendet; die Mittelung wird also 100 mal ausgeführt. Dabei ergaben sich die in Tabellen 6.1 und 6.2 auf Seite 44 aufgeführten Messwerte für die Kosten und Anzahlen der DTW-Berechnungen.

Betrachten wir zunächst die sich durch das berechnete Zentroid ergebenden Gesamtkosten. Man sieht, dass die 'Mittelung durch Fixpunktiterationen' auf allen Testklassen die besten Ergebnisse erzielt. Die schlechteste berechnete Lösung des Verfahrens ist immer vergleichbar mit der besten Lösung des 'Fast Shapebased Averaging' oder der Lösung des 'Shapebased Averaging', die beste berechnete Lösung ist auf vier von sechs Testmengen in etwa doppelt so gut.

Weiterhin sieht man, dass die 'Mittelung durch Projektion', die ja nur eine Approximation der 'Mittelung durch Fixpunktiteration' darstellt, im Durchschnitt auch Kosten verursacht, die mit denen der beiden 'Shapebased Averaging'-Verfahren vergleichbar sind.

Wie erwartet schneidet die 'Euklidsche Mittelung' nicht gut ab, die verursachten Kosten sind höher als alle anderen Durchschnittskosten. Aus mathematischer Sicht sollte also zur Lösung des Optimierungsproblems k-means die 'Mittelung durch Fixpunktiteration' verwendet werden.

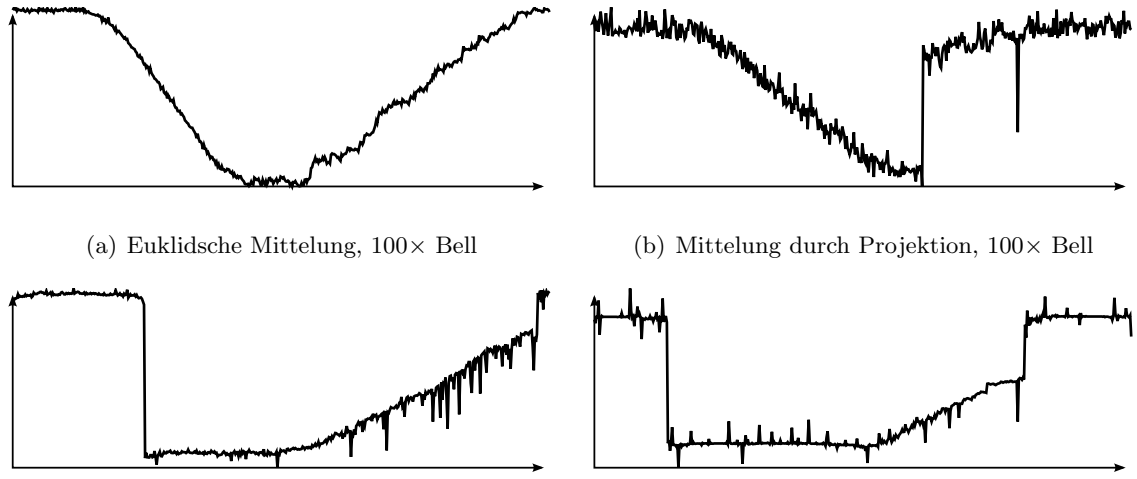
Nun betrachten wir noch die zur Berechnung der Zentroide benötigten DTW-Berechnungen. Dabei gilt, dass bei den Verfahren 'Mittelung durch Projektion', 'Mittelung durch Fixpunktiteration' und 'Fast Shapebased Averaging' jeweils 100 der DTW-Berechnungen schon im Rahmen des k-means Algorithmus ausgeführt werden und somit nicht erneut durchgeführt werden müssen.
Die 'Euklidsche Mittelung' lässt sich natürlich ohne DTW-Berechnungen ausführen und auch die 'Mittelung durch Projektion' braucht neben den bei k-means anfallenden Berechnungen keine zusätzlichen auszuführen. Die beiden Verfahren sind demnach die performantesten.

Das 'Fast Shapebased Averaging' punktet mit einer linearen Anzahl von Abstandsberechnungen. Zum Mitteln von n Zeitreihen müssen genau $2(n-1)$ zusätzliche DTW-Berechnungen ausgeführt werden. Im Gegensatz dazu hängt die Anzahl der DTW Berechnungen bei der 'Mittelung durch Fixpunktiteration' von der Anzahl benötigter Iterationen c ab, in jeder Iteration müssen n Berechnungen durchgeführt werden, somit werden $c \cdot n$ Berechnungen durchgeführt. Auf den gegebenen Daten gilt immer $c > 2$, durchschnittlich werden 4-5 Iterationen benötigt. Somit ist die 'Mittelung durch Fixpunktiteration' nicht so performant wie das 'Fast Shapebased Averaging'.

Von der Verwendung des 'Shapebased Averaging' muss klar abgeraten werden, die Anzahl der DTW-Berechnungen ist trotz der Optimierungen von Keogh gigantisch groß und bei den vorliegenden Alternativen nicht hinnehmbar.

Schließlich soll noch die Form der berechneten Zentroide visualisiert und bewertet werden. Dazu wurden in Abbildung 6.2 und 6.3 einige der berechneten Zentroide graphisch dargestellt.

In der ersten Zeile von Abbildung 6.2 sehen wir das euklidisch ermittelte Zentroid (Abb. 6.2(a)) einem durch Projektion ermittelten Zentroid (Abb. 6.2(b)) gegenübergestellt. Die Form des euklidschen Zentroids hat mit der erwünschten Form der Bell Klasse jedoch nicht viel gemein und könnte auch als Zentroid für Zeitreihen der Klasse Cylinder oder sogar Funnel durchgehen. Dem gegenüber kann man beim durch Projektion berechneten Zentroid eindeutig die Charakteristika einer Bell-Kurve erkennen, also zunächst ein konstanter Ab-



(a) Euklidsche Mittelung, 100× Bell

(b) Mittelung durch Projektion, 100× Bell

(c) Mittelung durch Projektion, 100× Cylinder

(d) Mittelung durch Fixpunktiteration, 100× Cylinder

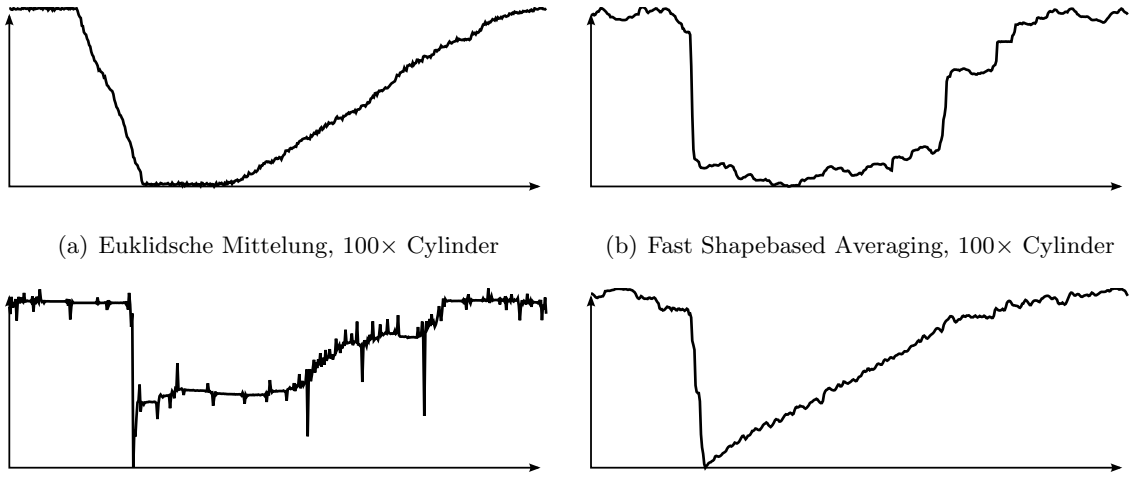
Abbildung 6.2: Einige der ermittelten Zentroide

schnitt, dann ein Abschnitt mit linear abnehmenden Messwerten, schließlich ein harter Sprung gefolgt von einem konstanten Abschnitt.

In der zweiten Zeile wird ein durch Projektion ermitteltes Zentroid (Abbildung 6.2(c)) mit einem durch Fixpunktiteration gemittelten Zentroid (Abbildung 6.2(d)) verglichen auf Zeitreihen der Klasse Cylinder. Man sieht, dass eine ungünstige Wahl des alten Zentroids, in diesem Fall vermutlich eine Zeitreihe, deren zweiter Niveauwechsel sehr nah am rechten Rand liegt, bei der Mittelung zu Projektion zu unschönen Ergebnissen führt. Durch wiederholtes Ausführen der Projektion im Rahmen der Fixpunktiteration löst sich dieses Problem jedoch zumindest teilweise; am rechten Rand der Zeitreihe ist jetzt deutlich ein Sprung zu erkennen.

In der ersten Zeile von Abbildung 6.3 werden nochmals Zeitreihen der Cylinder-Klasse gemittelt; im euklidisch ermittelten Zentroid (Abbildung 6.3(a)) erkennt man zwar einen konstanten Abschnitt in der Mitte, allerdings keinen abrupten Sprung der Werte. Das von Fast Shapebased Averaging berechnete Zentroid (Abbildung 6.3(b)) identifiziert im linken Teil der Zeitreihe den abrupten Niveauwechsel, am rechten Rand kommt es jedoch zu Treppenbildung. Diese resultiert möglicherweise daraus, dass eine Teilmenge der Cylinder-Zeitreihen längere Mittelteile hatten, als das Global Constraint erlaubt.

In der zweiten Zeile betrachten wir Zentroide, die sich aus einer unsauberen Menge von Bell-Zeitreihen ergeben. Das durch Fixpunktiteration berechntete Zentroid (Abbildung 6.3(c)) hat zwar fast die Form einer Bell-Zeitreihe, ist allerdings sehr zackig und hat an einigen



(a) Euklidsche Mittelung, 100× Cylinder

(b) Fast Shapebased Averaging, 100× Cylinder

(c) Mittelung durch Fixpunktiteration, 80× Funnel, 10× Cylinder, 10× Bell

(d) Shapebased Averaging, 80× Funnel, 10× Cylinder, 10× Bell

Abbildung 6.3: Einige der ermittelten Zentroide

Positionen Ausreißer. Es scheint, als würde die Fixpunktiteration ein überangepasstes Zentroid berechnen. Dem gegenüber ist das durch Shapebased Averaging berechnete Zentroid (Abbildung 6.3(d)) sehr glatt und hat eine fast perfekte Form. Im Umkehrschluss heißt das allerdings, das 20% der gegebenen Zeitreihen keinen Einfluss auf das Resultat haben. Außerdem kann davon ausgegangen werden, dass für größere Eingabemengen die durch Fixpunktiteration berechneten Zentroide auch glatter werden, da sich das Rauschen ausmittelt.

Durch (Fast) Shapebased Averaging ermittelte Zentroide eignen sich aber scheinbar gut dazu, eine unverrauschte Anschauung über die Form der vorliegenden Zeitreihen zu gewinnen.

C	**Euklidsche Mittelung**	**Mittelung durch Projektion**	**Mittelung durch Fixpunktiteration**	**Shapebased Averaging**	**Fast Shapebased Averaging**
100 Cylinder	10998	min: 6092 avg: 8403 max: 12170	min: 5752 avg: 6310 max: 7317	8216	min: 7473 avg: 8046 max: 8631
100 Bell	19744	min: 9534 avg: 13870 max: 20215	min: 6795 avg: 8754 max: 11825	13135	min: 11881 avg: 12942 max: 14034
100 Funnel	5119	min: 715 avg: 930 max: 1456	min: 665 avg: 711 max: 762	1483	min: 1197 avg: 1339 max: 1438
80 Cylinder + 20 Rest	12181	min: 6747 avg: 9499 max: 14195	min: 6170 avg: 6856 max: 8226	9334	min: 8179 avg: 8737 max: 9403
80 Bell + 20 Rest	20237	min: 10934 avg: 15657 max: 22770	min: 7749 avg: 9839 max: 13233	14122	min: 13622 avg: 14448 max: 15325
80 Funnel + 20 Rest	9346	min: 4565 avg: 6404 max: 12738	min: 3456 avg: 4496 max: 6676	6335	min: 5931 avg: 6226 max: 6606

Tabelle 6.1: Kosten $cost(C, z)$ des ermittelten Zentroids z

C	**Euklidsche Mittelung**	**Mittelung durch Projektion**	**Mittelung durch Fixpunktiteration**	**Shapebased Averaging**	**Fast Shapebased Averaging**
100 Cylinder	0	100	min: 400 avg: 602 max: 1000	20854	298
100 Bell	0	100	min: 400 avg: 723 max: 1300	39157	298
100 Funnel	0	100	min: 400 avg: 678 max: 1000	47248	298
80 Cylinder + 20 Rest	0	100	min: 400 avg: 733 max: 1200	12909	298
80 Bell + 20 Rest	0	100	min: 500 avg: 743 max: 1200	24145	298
80 Funnel + 20 Rest	0	100	min: 500 avg: 807 max: 1200	31689	298

Tabelle 6.2: Benötigte DTW-Berechnungen zum Ermitteln der Zentroide.

6.2 Evaluierung des Clusteringalgorithmus

Die folgenden drei Experimente sollen den entwickelten k-means Clusteringalgorithmus evaluieren. Zunächst wird die k-means++ Initialisierung analysiert. Dann werden in einem umfangreichen Experiment mit 20 verschiedenen Datensätzen die verschiedenen Mittelungstechniken ausführlich verglichen. Anschließend wird in einem weiteren Experiment untersucht, wie effektiv die optimierten Querys im Anwendungsfall k-means tatsächlich sind.

6.2.1 Analyse der k-means++ Startwertinitialisierung

Das folgende Experiment soll untersuchen, inwieweit die k-means++ Startwertinitialisierung auch beim Clustering von Zeitreihen zu besseren Ergebnissen führt als die vollständig zufällige Wahl initialer Zentroide.

Dazu wird wieder der Cylinder-Bell-Funnel Datensatz verwendet. Mit insgesamt 492 z-normalisierten Zeitreihen der Länge 256, also je 164 Zeitreihen jeder Klasse, wird das k-means Clustering 1000 mal mit k-means++ Initialisierung und 1000 mal ohne ausgeführt. Dabei wird die Mittelung durch Fixpunktiteration sowie ein Sakoe-Chiba-Band der Breite 64 eingesetzt. Als Leistungsmaß werden die berechneten Gesamtkosten und die benötigte Anzahl der Iterationen[1] gemessen.

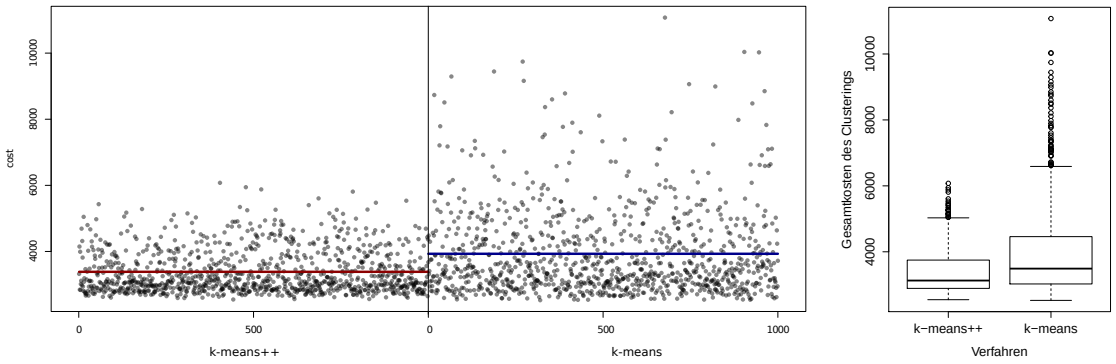
In Abbildung 6.4 sieht man die berechneten Gesamtkosten der 2000 Durchläufe des Algorithmus. Der k-means++ Algorithmus hat deutlich weniger Ausreißer nach oben, also schlechte Durchläufe, als der k-means Algorithmus. Dies spiegelt sich auch im Mittelwert aller berechneten Kosten wieder; der Mittelwert aller Kosten der Durchläufe mit k-means++ beträgt 3384.453, der Mittelwert von k-means beträgt nur 3929.466[2].

Allerdings berechnet der k-means Algorithmus die beste Lösung von $min = 2526.11$. Nur 10 Durchläufe von k-means++ erreichen Gesamtkosten kleiner als $min + 100$, mit k-means erreichen 28 Lösungen sehr gute Lösungen.
Somit berechnet k-means++ weniger schlechte und weniger sehr gute Lösungen, erzielt aber im Durchschnitt bessere und stabilere Resultate.

[1] maximal 20

[2] Um diese Kosten in Relation setzen zu können, mitteln wir die Klassen Cylinder,Bell und Funnel separat und berechnen die Gesamtkosten der so entstehenden Zentroide - es fallen Kosten um 8500 an. Somit berechnet k-means Lösungen, die im mathematischen Sinn besser sind als die Enteilung in die ursprünglichen Klassen.

Betrachtet man in Abbildung 6.5 die Anzahl der benötigten Iterationen der beiden Algorithmen, sieht man, dass k-means++ häufiger weniger Iterationen benötigt als k-means. Auch im Durchschnitt benötigt k-means++ 2 Iterationen weniger als k-means.
Insgesamt scheint das Suchen von initialen Zentroiden, die möglichst weit auseinander liegen, mit der k-means++ Startwertinitialisierung also auch beim Clustering von Zeitreihen eine gute Idee zu sein.



(a) Scatterplot (b) Boxplot

Abbildung 6.4: Die berechneten Endkosten von k-means im Vergleich. Zum Scatterplot: Die ersten 1000 Iterationen benutzen die k-means++ Startwertinitialisierung, die zweiten 1000 Iterationen wählen die Initialen Zentroide vollständig zufällig. Der in rot eingezeichnete Mittelwert der ersten 1000 Kosten ist 3384.453, der in blau eingezeichnete Mittelwert der zweiten 1000 Iterationen ist 3929.466.

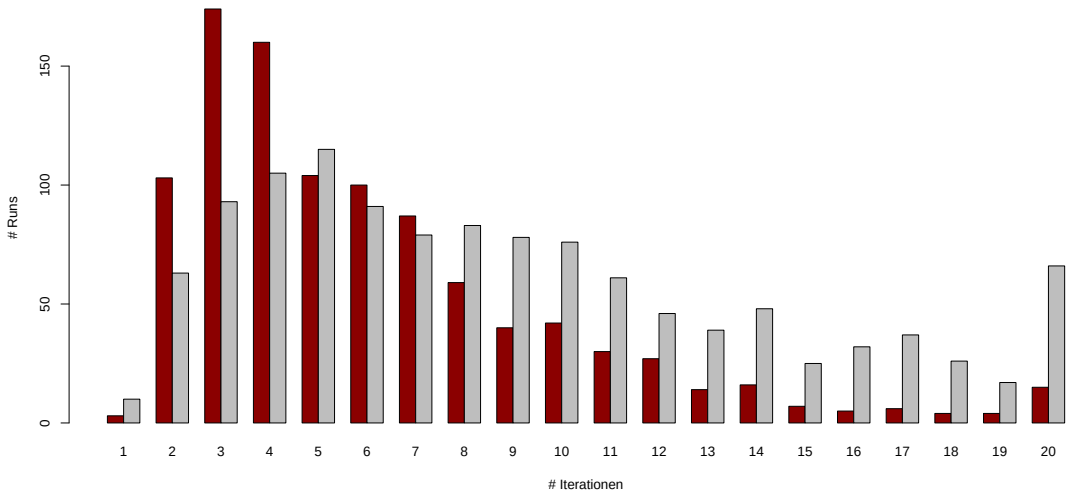


Abbildung 6.5: Vergleich der Anzahl der benötigten Iterationen von k-means++ und k-means. Der Algorithmus wurde je 1000 mal ausgeführt, im Barplot wird die Anzahl der Durchläufe von k-means++(Dunkelrot) und k-means(Grau), die i Iterationen benötigen, verglichen. Im Durchschnitt benötigt k-means++ 6.147 und k-means 8.650 Iterationen.

6.2.2 Vergleich der Mittelungsverfahren auf verschiedenen Testdatensätzen

Keogh stellt eine Reihe von Datensätzen mit gelabelten Zeitreihen zur Verfügung (siehe [8]). Aus diesen Datensätzen habe ich 20 ausgewählt, die sich in Anzahl Klassen, Anzahl Zeitreihen[3] und Länge der Zeitreihen unterscheiden. In Tabelle 6.3 sind die Eigenschaften zusammengefasst.

Name	Anzahl Klassen	Anzahl Zeitreihen	Länge Zeitreihen
50words	50	450	270
FaceFour	4	24	350
Trace	4	100	275
Adiac	37	390	176
Gun-Point	2	50	150
Two-Patterns	4	1000	128
Beef	5	30	470
Lighting2	2	60	637
fish	7	175	463
CBF	3	30	128
Lighting7	7	70	319
synthetic-control	6	300	60
Coffee	2	28	286
OSULeaf	6	200	427
wafer	2	1000	128
ECG200	2	100	96
OliveOil	4	30	570
yoga	2	300	426
FaceAll	14	560	131
SwedishLeaf	15	500	128

Tabelle 6.3: Die ausgewählten Testdatensätze

Die Zeitreihen dieser Datensätze werden z-normalisiert, dann wird für jeden Datensatz das k-means++ Clustering mit 5 verschiedenen Mittelungsverfahren - Mittelung durch Fixpunktiteration, Mittelung durch Projektion, Fast Shapebased Averaging, Medoid und Euklidische Mittelung - je 50 mal ausgeführt. Die Anzahl der zu berechnenden Cluster k wird auf die Anzahl der Klassen des Testdatensatzes gesetzt. Dabei werden die Anzahl benötigter Iterationen, die Anzahl der DTW-Berechnungen, die berechneten Gesamtkosten des Clusterings und die Anzahl fehlgeschlagener[4] Durchläufe protokolliert.

[3] Die Daten sind unterteilt in Trainings- und Testdatensätze, ich verwende nur den Trainingsdatensatz
[4] Es wurden weniger als k Cluster berechnet

Da die Daten alle gelabelt sind, kann auch ermittelt werden, inwieweit das berechnete Clustering mit den tatsächlichen Klassen übereinstimmt, so erhält man mit der Accuracy ein weiteres Leistungsmaß. Zum Berechnen der Accuracy muss die passendste Bijektion zwischen den k Clustern und den k Labels gefunden werden. Für kleine k kann das noch durch Ausprobieren aller möglichen Permutationen erreicht werden; mit der Ungarischen Methode [12] kann die Accuracy jedoch auch in polynomieller Laufzeit berechnet werden.

Mithilfe der so gewonnen Daten in Tabelle 6.4 bis 6.7 lassen sich einige Hypothesen untersuchen, die im Verlauf der Arbeit schon aufgestellt wurden. Diese Hypothesen sollen dann, wie in [4] empfohlen, mit dem Wilcoxon-Vorzeichen-Rang-Test untersucht werden. Der Wilcoxon-Vorzeichen-Rang-Test testet für zwei gepaarte Stichproben, ob die Differenz der Mediane signifikant größer 0 ist, ohne dafür eine Normalverteilungsannahme zu treffen.

> The Wilcoxon signed-ranks test (Wilcoxon, 1945) is a non-parametric [...] [paired test], which ranks the differences in performances of two classifiers for each data set, ignoring the signs, and compares the ranks for the positive and the negative differences.[4]

Beim Vergleich zweier Algorithmen bezüglich eines Leistungsmaßes auf einer Menge von Datensätzen lautet die Nullhypothese immer, dass beide Algorithmen gleich gut sind bzw. die Differenz der Mediane des jeweiligen Leitungsmaßes 0 ist.

In Kapitel 4.2.3 wurde vermutet, dass Mittelung durch Projektion und Mittelung durch Fixpunktiteration gleich gute Ergebnisse liefern, da die Iterationen des Mittelns auf die Iterationen des k-means Clusterings umverteilt werden. Dies liefert die folgende Hypothese:

> H_1 : Die Mittelung durch Projektion berechnet ein Clustering mit Gesamtkosten, die nicht wesentlich schlechter sind als die der Mittelung durch Fixpunktiteration.

In Kapitel 6.1 wurden die verschiedenen Mittelungsverfahren unabhängig von k-means verglichen. Die Mittelung durch Fixpunktiteration berechnete dort Zentroide, die niedrigere Gesamtkosten verursachten als das des Fast Shapebased Averaging. Dies sollte sich auch beim k-means Clustering beobachten lassen:

> H_2 : Die Mittelung durch Fixpunktiteration berechnet ein Clustering mit den geringeren Gesamtkosten als das Fast Shapebased Averaging.

Die Verwendung von k-medoids ist mit einer Beschränkung des Suchraums der Zentroide verbunden, da die Medoide nicht weiter an die Daten angepasst werden können. Somit

sollten die Medoide auch höhere Gesamtclusterkosten verursachen als die anderen Mittelungsverfahren, ausgenommen die euklidische Mittelung, da diese nicht an das DTW-Abstandsmaß angepasst ist.

> H_3 : k-medoids berechnet höhere Gesamtkosten als k-means mit einem der Mittelungsverfahren für DTW.

Nach Zerlegung der Hypothese H_3 in drei Teilhypothesen - für jedes für DTW entwickelte Mittelungsverfahren eine - kann die These mit dem Wilcoxon-Test untersucht werden.

Nach einem Blick in die Daten können weitere Hypothesen aufgestellt werden. Auffällig ist, dass die Euklidische Mittelung und das Fast Shapebased Averaging auf manchen Datensätzen viele Fehlschläge verursachen:

> H_4 : Die euklidische Mittelung und das Fast Shapebased Averaging schlagen häufiger fehl als die restlichen Verfahren

Analog zu H_3 kann die Hypothese H_4 in sechs Teilhypothesen zerlegt werden, indem die beiden Verfahren euklidische Mittelung und Fast Shapebased Averaging einzeln gegen die verbleibenden drei Verfahren getestet werden.

Beim Betrachtung der benötigten Anzahlen von Iterationen fällt folgendes auf:

> H_5 : Das Shapebased-Averaging benötigt deutlich mehr Iterationen für die Berechnung als die Mittelung durch Projektion/Fixpunktiteration.

Ein direktes Resultat daraus ist:

> H_6 : Das Shapebased-Averaging benötigt deutlich mehr DTW-Berechnungen für die Berechnung als die Mittelung durch Projektion/Fixpunktiteration.

Auch die Hypothesen H_5 und H_6 müssen in je zwei Thesen zerlegt werden, die beide mit einen Wilcoxon Test untersucht werden.

Als Letzes sollte noch die Klassifikationsgüte untersucht werden, hier sieht es so aus, als würden alle Verfahren gleich gut klassifizieren, mit der Einschränkung, dass die Euklidische Mittelung nur auf den Datensätzen mithalten kann, auf denen sie nicht immer fehlschlägt.

> H_7 : Die durchschnittlichen Accuracys aller Verfahren unterscheiden sich nicht.

Die These H_7 kann gezeigt werden, indem alle Hypothesen, dass ein Verfahren nicht besser oder schlechter ist als ein anderes, mit einem Wilcoxon-Test untersucht werden.

Alle aus den sieben aufgestellten Hypothesen resultierenden Teilhypothesen können durch Wilcoxon-Vorzeichen-Rang-Test mit Signifikanzniveau von 5% belegt werden.

Die erste These H_1 besteht eigentlich aus zwei Thesen: Zum einen die, dass die Fixpunktiteration besser ist als die Projektion und zum andern, dass der Unterschied nicht wesentlich ist. Die Gültigkeit der ersten Teilthese zeigt der Wilcoxon-Vorzeichen-Rang-Test mit Signifikanzniveau von 5%. Die zweite Teilthese kann nicht durch einen Test gegründet werden, ein Blick in die Daten zeigt aber, dass die Mittelung durch Projektion im Mittel nur 5% schlechter ist als die Mittelung durch Fixpunktiteration. Die relativen Unterschiede zwischen der Mittelung durch Fixpunktiteration und den anderen Verfahren sind wesentlich höher. Folglich kann H_1 begründet angenommen werden.

Insgesamt zeigt sich also, dass die beiden von mir entwickelten Verfahren keine schlechtere Accuracy haben als die bestehenden Verfahren, jedoch weniger Iterationen und folglich weniger DTW-Berechnungen benötigen, um eine Lösung zu berechnen, die geringere Gesamtclusterkosten hat als alle bestehenden Verfahren.

dataset	projection	fixpoint	shapebased	euclidian	medoid
Beef	0.273 (0.367)	0.280 (0.400)	0.518 (0.567)	0.535 (0.567)	0.545 (0.567)
CBF	0.620 (0.833)	0.620 (0.767)	0.628 (0.733)	0.654 (0.667)	0.633 (0.633)
Coffee	0.852 (1.000)	0.865 (1.000)	0.919 (1.000)	0.929 (1.000)	0.840 (0.857)
ECG200	0.719 (0.810)	0.722 (0.800)	0.741 (0.810)	0.751 (0.800)	0.720 (0.750)
FISH	0.458 (0.543)	0.457 (0.583)	0.452 (0.514)		0.447 (0.526)
FaceFour	0.539 (0.708)	0.544 (0.792)	0.560 (0.708)	0.516 (0.625)	0.486 (0.625)
Gun	0.557 (0.720)	0.576 (0.720)	0.549 (0.600)	0.525 (0.560)	0.560 (0.560)
Lighting2	0.647 (0.700)	0.650 (0.750)	0.649 (0.683)	0.646 (0.683)	0.665 (0.683)
Lighting7	0.562 (0.671)	0.542 (0.657)	0.520 (0.586)	0.451 (0.529)	0.558 (0.643)
OSULeaf	0.388 (0.475)	0.383 (0.435)	0.384 (0.435)	0.360 (0.380)	0.373 (0.440)
OliveOil	0.755 (0.867)	0.759 (0.900)	0.776 (0.867)	0.772 (0.867)	0.806 (0.867)
Trace	0.616 (0.740)	0.613 (0.720)	0.646 (0.750)	0.567 (0.590)	0.584 (0.710)
Two	0.370 (0.459)	0.374 (0.493)	0.363 (0.445)	0.345 (0.413)	0.370 (0.428)
synthetic	0.691 (0.917)	0.675 (0.910)	0.653 (0.843)	0.753 (0.857)	0.620 (0.710)
wafer	0.654 (0.654)	0.654 (0.654)	0.668 (0.674)	0.670 (0.677)	0.667 (0.677)
yoga	0.512 (0.547)	0.514 (0.547)	0.507 (0.513)	0.516 (0.527)	0.529 (0.540)

Tabelle 6.4: Durchschnittliche Accuracys, in Klammern die beste Accuracy aus 50 Runs.

dataset	projection	fixpoint	shapebased	euclidian	medoid
50words	$2237.3 \pm 3\%$	$2187.0 \pm 3\%$	$1989.9 \pm 17\%$		$3421.5 \pm 12\%$
Adiac	$31.1 \pm 2\%$	$31.5 \pm 3\%$		$40.3 \pm 2\%$	$46.3 \pm 6\%$
Beef	$81.7 \pm 31\%$	$77.7 \pm 32\%$	$21.9 \pm 74\%$	$19.5 \pm 18\%$	$28.6 \pm 30\%$
CBF	$529.9 \pm 9\%$	$497.8 \pm 8\%$	$649.7 \pm 6\%$	$830.7 \pm 2\%$	$868.2 \pm 0\%$
Coffee	$14.4 \pm 10\%$	$13.5 \pm 8\%$	$18.5 \pm 3\%$	$17.6 \pm 5\%$	$25.2 \pm 0\%$
ECG200	$670.8 \pm 9\%$	$640.3 \pm 7\%$	$970.4 \pm 6\%$	$923.2 \pm 6\%$	$2396.6 \pm 24\%$
FISH	$75.9 \pm 6\%$	$75.2 \pm 6\%$	$117.4 \pm 12\%$		$118.8 \pm 18\%$
FaceAll	$7493.2 \pm 3\%$	$7507.7 \pm 2\%$	$12605.0 \pm 2\%$		$13178.9 \pm 6\%$
FaceFour	$543.9 \pm 6\%$	$524.2 \pm 6\%$	$481.0 \pm 19\%$	$831.1 \pm 22\%$	$453.6 \pm 11\%$
Gun	$202.0 \pm 19\%$	$152.7 \pm 11\%$	$202.6 \pm 15\%$	$203.5 \pm 1\%$	$412.9 \pm 0\%$
Lighting2	$7475.2 \pm 7\%$	$6657.3 \pm 7\%$	$9248.0 \pm 11\%$	$12370.2 \pm 6\%$	$13037.4 \pm 7\%$
Lighting7	$2303.2 \pm 4\%$	$2192.2 \pm 4\%$	$3512.1 \pm 9\%$	$4787.9 \pm 5\%$	$4076.1 \pm 14\%$
OSULeaf	$4508.1 \pm 6\%$	$4346.7 \pm 5\%$	$7282.7 \pm 3\%$	$27350.4 \pm 7\%$	$13390.4 \pm 14\%$
OliveOil	$0.3 \pm 8\%$	$0.3 \pm 6\%$	$0.3 \pm 8\%$	$0.3 \pm 8\%$	$0.5 \pm 12\%$
SwedishLeaf	$850.7 \pm 5\%$	$853.7 \pm 5\%$			$1758.5 \pm 14\%$
Trace	$282.8 \pm 18\%$	$264.2 \pm 20\%$	$381.2 \pm 28\%$	$1336.3 \pm 20\%$	$1313.6 \pm 52\%$
Two	$29294.0 \pm 2\%$	$29105.5 \pm 3\%$	$55101.5 \pm 5\%$	$105029.4 \pm 2\%$	$58952.5 \pm 9\%$
synthetic	$2944.8 \pm 4\%$	$2902.2 \pm 2\%$	$5527.5 \pm 4\%$	$6352.9 \pm 3\%$	$4437.0 \pm 8\%$
wafer	$19190.0 \pm 6\%$	$18729.4 \pm 7\%$	$28775.8 \pm 3\%$	$34445.3 \pm 3\%$	$48453.5 \pm 18\%$
yoga	$6038.9 \pm 15\%$	$5176.9 \pm 14\%$	$9605.6 \pm 4\%$	$20226.2 \pm 2\%$	$25856.7 \pm 36\%$

Tabelle 6.5: Durchschnittliche Gesamtclusterkosten $\pm$ relative Standardabweichung

dataset	projection	fixpoint	shapebased	euclidian	medoid
50words	112175.0	110340.7	137201.8		134997.9
Adiac	71043.1	71902.5		89831.5	125328.6
Beef	387.4	617.0	575.5	368.1	502.1
CBF	333.6	506.8	653.6	383.4	570.0
Coffee	303.8	521.7	533.5	256.4	572.5
ECG200	1251.5	1899.2	4191.8	893.8	7533.1
FISH	8866.3	9544.0	24403.9		24235.2
FaceAll	84792.0	94073.6	133446.6		214521.0
FaceFour	340.1	476.2	580.0	800.2	410.3
Gun	455.2	1009.5	1061.6	517.7	1409.5
Lighting2	583.2	1109.3	1144.4	442.4	1853.4
Lighting7	1847.6	2240.2	7302.4	3919.7	2466.7
OSULeaf	8627.0	9119.2	26421.8	13836.3	23137.4
OliveOil	502.1	642.2	640.0	425.8	631.0
SwedishLeaf	40837.9	41890.3			205388.9
Trace	1480.4	2190.5	3452.2	4092.8	3422.0
Two	86793.6	107667.3	132858.8	61825.8	441663.6
synthetic	9358.1	11633.6	38407.3	14518.4	23479.7
wafer	9033.2	13937.8	23710.1	6955.7	307626.9
yoga	4825.1	7399.7	10924.8	3730.2	63318.0

Tabelle 6.6: Durchschnittliche Gesamtanzahl der DTW-Berechnungen

dataset	projection	fixpoint	shapebased	euclidian	medoid
50words	6.7 (0)	4.4 (0)	15.4 (39)	(50)	19.8 (0)
Adiac	8.5 (0)	7.5 (0)	(50)	20.0 (11)	18.9 (0)
Beef	1.4 (0)	1.6 (0)	1.6 (5)	1.6 (0)	1.7 (0)
CBF	1.5 (0)	1.5 (0)	2.9 (0)	3.8 (0)	2.4 (0)
Coffee	2.3 (0)	2.3 (0)	2.7 (0)	3.3 (0)	2.4 (0)
ECG200	3.2 (0)	2.7 (0)	8.2 (0)	3.9 (0)	4.3 (0)
FISH	8.2 (0)	6.5 (0)	20.0 (5)	(50)	11.3 (0)
FaceAll	13.0 (0)	13.0 (0)	20.0 (0)	(50)	15.4 (0)
FaceFour	1.5 (0)	1.5 (0)	2.9 (2)	13.1 (42)	1.8 (0)
Gun	2.0 (0)	2.1 (0)	3.5 (0)	4.9 (0)	2.4 (0)
Lighting2	2.2 (0)	2.3 (0)	2.8 (0)	2.7 (0)	1.9 (0)
Lighting7	2.8 (0)	2.6 (0)	15.4 (1)	17.4 (1)	3.0 (0)
OSULeaf	7.1 (0)	4.7 (0)	20.0 (0)	20.0 (32)	7.4 (0)
OliveOil	2.0 (0)	2.0 (0)	1.9 (0)	2.3 (0)	2.1 (0)
SwedishLeaf	12.9 (0)	9.2 (0)	(50)	(50)	16.5 (0)
Trace	2.4 (0)	2.1 (0)	5.1 (0)	18.6 (0)	3.5 (0)
Two	18.6 (0)	18.1 (0)	20.0 (0)	20.0 (22)	4.9 (0)
synthetic	4.7 (0)	4.5 (0)	20.0 (0)	14.6 (40)	5.9 (0)
wafer	1.9 (0)	1.6 (0)	4.0 (11)	2.6 (0)	2.8 (0)
yoga	4.8 (0)	3.4 (0)	6.8 (0)	6.0 (0)	5.9 (0)

Tabelle 6.7: Durchschnittliche Anzahl Iterationen, in Klammern steht die Anzahl der fehlgeschlagenen Runs.

6.2.3 Analyse des Anteils der vollständig ausgeführten DTW-Berechnungen

Ein wesentlicher Aspekt dieser Arbeit war es, die von Keogh et al. entwickelten Ideen für effizientere Querys mit Dynamic Time Warping auf das k-means Clustering anzuwenden. Dazu wurden die Ideen im Zuweisungsschritt des k-means Algorithmus angewandt, in dem für jede Zeitreihe das nächste Zentroid ermittelt werden muss. Im besten Fall muss so pro Zeitreihe statt k DTW-Berechnungen nur noch eine Berechnung ausgeführt werden. Jetzt soll analysiert werden, wie viele DTW-Berechnungen sich so tatsächlich vermeiden lassen.

Es wurde schon die Vermutung aufgestellt, dass mit größeren ks der Anteil der ausgeführten DTW-Berechnungen abnimmt. Dies soll jetzt experimentell untersucht werden. Dazu zähle ich die im Zuge der Aufteilung der Zeitreihen auf die Zentroide ausgeführten DTW-Berechnungen und berechne so den Anteil an den maximal möglichen $k \cdot |M|$ Berechnungen. Um auch das Early Abandoning zu berücksichtigen, werden nur anteilig ausgeführte DTW-Berechnungen auch nur anteilig gezählt.

Das Experiment wird auf den Daten des von Keogh bereitgestellten SwedishLeaf Datensatzes (siehe oben) ausgeführt. Dieser eignet sich gut, da die Daten in 15 verschiedenen Klassen eingeteilt sind und somit das Clustering für große k auch sinnvoll scheint. In diesem Experiment verwende ich die Vereinigung der Trainings- und der Testmenge um eine möglichst große Eingabemenge zu erhalten. Dann werden je 10 Runs des k-means Clusterings mit $k = 3, ..., 25$ und verschieden breiten Sakoe-Chiba Bändern ausgeführt.

Die Resultate in Tabelle 6.8 und Abbildung 6.6 zeigen, dass der Anteil der ausgeführten DTW-Berechnungen tatsächlich mit steigendem k abnimmt. Dabei hat die Kurve der Messwerte unabhängig von der Breite des Sakoe-Chiba-Bands immer die selbe Form, unterscheidet sich aber in den Werten. Je schmaler das Sakoe-Chiba-Band ist, desto weniger DTW-Berechnungen müssen ausgeführt werden. Dies ist wenig verwunderlich, da für schmale Bänder die LB_{keogh} die DTW-Berechnung immer genauer abschätzt.

Insgesamt sieht man, dass die Optimierungen, insbesondere für große k, auch im Anwendungsfall k-means helfen, die Anzahl der benötigten DTW-Berechnungen zu verringern. So lassen sich tatsächlich mehr als die Hälfte der DTW-Berechnungen einsparen, wenn k größer wird sogar bis zu 80% aller Berechnungen.

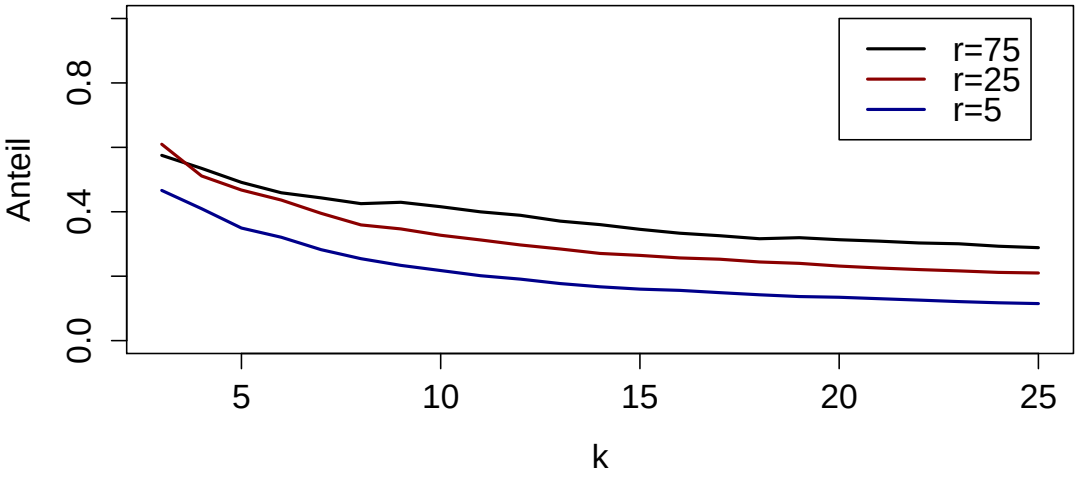


Abbildung 6.6: Der durchschnittlche Anteil der ausgeführten DTW-Berechnungen im Zuge der Zuweisung der Zeitreihen auf die nächsten Zentroide in Abhängigkeit von der Breite r des Sakoa-Chiba-Bandes

k	r=75	r=25	r=5
3	0.575209763445160	0.609724653860066	0.466300957036307
4	0.534738241467551	0.511222828803604	0.409526094420643
5	0.491283753233661	0.467439858359486	0.349448322089200
6	0.458967170604588	0.436362588269118	0.320875717570266
7	0.443099828480058	0.395139089809493	0.282328902145823
8	0.425192043362839	0.359177853737410	0.254325429473467
9	0.429430234261810	0.346726653664213	0.233567261330044
10	0.415684388244378	0.327101395965592	0.217512289193780
11	0.399764805157926	0.312517909497046	0.201471311745273
12	0.389068862612501	0.296978530300253	0.190871413206656
13	0.370914054633606	0.284486167282954	0.177121581851872
14	0.360110514451209	0.270639136091930	0.167044514401756
15	0.345469641038173	0.264590315654106	0.159808083776889
16	0.333444250796381	0.256649098566281	0.155970636584872
17	0.325645183751466	0.252765180330602	0.148906839604345
18	0.316298191043287	0.244140453446248	0.142297822137012
19	0.319568202770379	0.239852370316763	0.136784272226145
20	0.313215467055828	0.231466169346326	0.134615666038130
21	0.308855712433675	0.225426765993025	0.130183879693672
22	0.303189660253716	0.220567717563290	0.125780413995477
23	0.300629988334312	0.216487295036289	0.121121920174105
24	0.292978107573220	0.211772306118051	0.117444526461960
25	0.288563864305764	0.209910534449630	0.114983912979166

Tabelle 6.8: Die gemittelten Anteile der im Zuge der Zuweisung der Zeitreihen auf die nächsten Zentroide ausgeführten DTW-Berechnungen

6.3 Clusteranalyse der Stahlwerk-Messreihen

Abschließend sollen die schon in der Einleitung angesprochenen Messreihen aus einem Stahlwerk eines führenden, deutschen Stahlherstellers untersucht werden. Hier werden in einem mehrstufigen Prozess an verschiedenen Verarbeitungsstationen zu jedem Stahlblock Messdaten wie Temperaturen im Drehherdofen sowie Drehzahlen und Kräfte von Walzen im Verlauf der Zeit, also Zeitreihen, erhoben. (Siehe Abbildung 6.7)

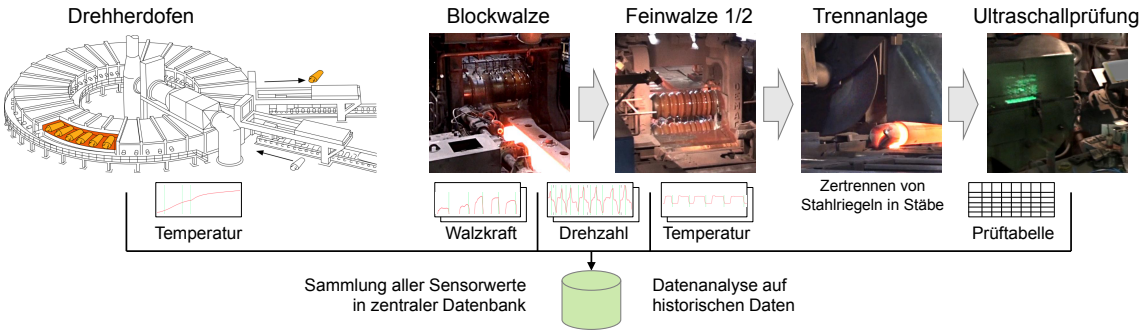


Abbildung 6.7: Der Fertigungsprozess graphisch dargestellt. An jeder Station werden Messkurven protokolliert. (Grafik: Marco Stolpe)

Die Sensordaten der verschiedenen Stationen können nun separat geclustert werden, eigentlich möchte man allerdings den gesamten Verarbeitungsprozess eines Stahlteils betrachten und nicht nur eine Station. Dazu kann man das Clustern der Zeitreihen als Vorverarbeitungsschritt einsetzen, um so eine Zeitreihe auf eine nominale Größe, die Clusterzugehörigkeit, zu reduzieren.

Es ist aus [13] bekannt, dass durch Extraktion von Features aus den Zeitreihen und anschließendem k-means Clustering der Featurevektoren die bekannten Endabmaße der Stahlblöcke identifiziert werden können. Für verschiedene Endabmaße werden verschiedene Einstellungen der Maschinen verwendet; diese verschiedenen Einstellungen resultieren dann auch in verschiedenen Messkurven. Die Lernaufgabe ist zwar eigentlich die Vorhersage der Qualität der Stahlblöcke, dies ist allerdings bis jetzt noch nicht zuverlässig gelungen. Momentan wird vermutet, dass die Unterscheidung zwischen fehlerhaften und fehlerfreien Stahlblöcken anhand feiner Unterschiede in den Messkurven einzelner Stiche der Walze geschehen kann. Da DTW jedoch die ganzen Zeitreihen betrachtet und so feine Unterschiede in Abschnitten der Zeitreihen keinen großen Einfluss auf den berechneten Abstand haben, wird auch der hier vorgestellte Ansatz das Problem der Qualitätsvorhersage nicht lösen.

Stattdessen soll überprüft werden, ob sich die Endabmaße über das Clustering der Zeitreihen mittels DTW ähnlich gut vorhersagen lassen können wie über extrahierte Merkmale.

Dabei verwende ich Zeitreihen aus sieben verschiedenen Sensoren:

Sensor:	102	257	268	501	503	504	505
Messwerte:	Temperatur im Drehherdofen	Blockwalze Drehzahl	Blockwalze Anstellung	Feinwalze 1 Anstellung	Feinwalze 1 Walzkraft	Feinwalze 1 Stichtemperatur	Feinwalze 1 Drehzahl

Für die Vorhersage wird das Clustering in Kombination mit einem weiteren Lernverfahren, dem ID3 Entscheidungsbaum-Lernen verwendet: Zunächst werden die Trainingsdaten sensorweise geclustert. Dabei setzt ich $k = 6$ und verwende die Mittelung durch Projektion und ausreichend große Sakoe-Chiba Bänder. So werden jedem Stahlblock aus der Trainingsmenge seine Endabmaße sowie die Clusterzugehörigkeiten seiner 7 Messreihen zugeordnet. Die Clusterzugehörigkeiten ergeben einen neuen Merkmalsraum, auf dem jetzt ein Entscheidungsbaum zur Vorhersage der Endabmaße gelernt werden kann. Aus diesem Baum und den 7×6 Zentroiden setzt sich das gelernte Vorhersage-Modell zusammen. (Siehe Abbildung 6.8)

Eine Verbesserung lässt sich erreichen, indem man das k-means Clustering mehrfach ausführt und aus den zufällig berechneten Clusterings diejenigen auswählt, die den besten Entscheidungsbaum liefern. Um die Qualität des Entscheidungsbaum zu beurteilen, kreuzvalidiere ich nur das Entscheidungsbaumlernen in 10 Schritten und verwende die

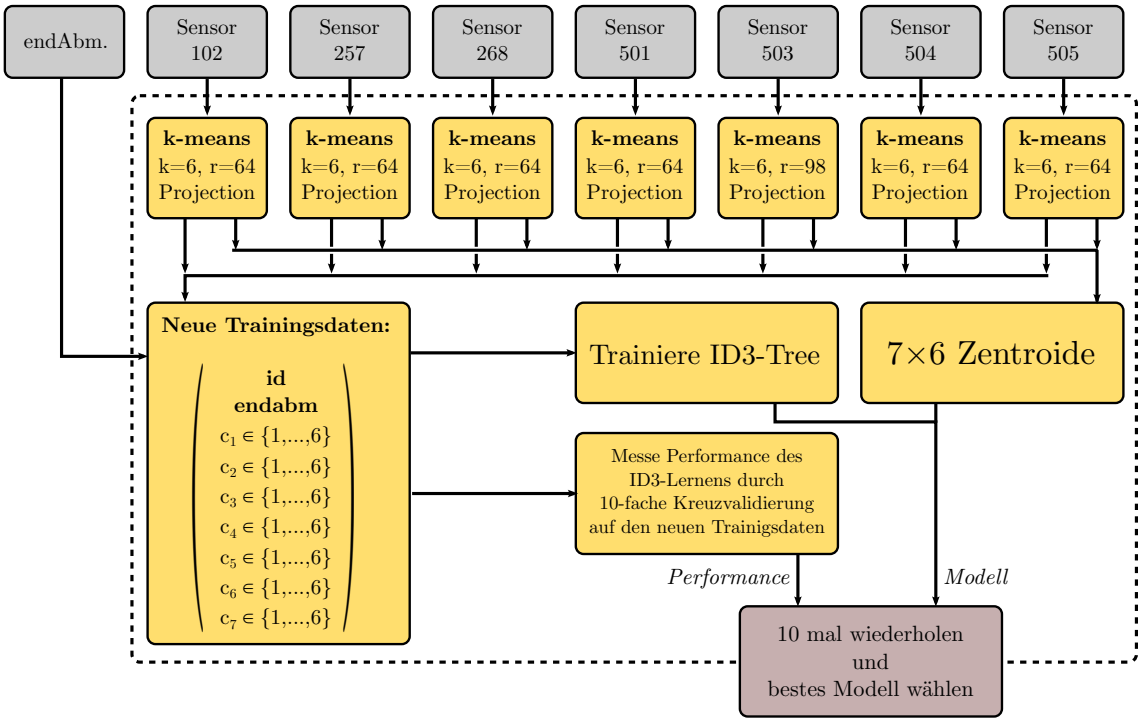


Abbildung 6.8: Der Lernvorgang zur Vorhersage der Endabmaße von Stahlblöcken

accuracy: 88.21% +/- 6.08% (mikro: 88.21%)

	true 140,0V	true 120,0V	true 110,0V	true 130,0V	true 160,0V	true 100,0V	true 182,5V	true 60,0V	class precision
pred. 140,0V	145	10	3	5	10	0	0	0	83.82%
pred. 120,0V	0	92	0	10	0	0	0	0	90.20%
pred. 110,0V	0	0	52	0	1	0	0	0	98.11%
pred. 130,0V	0	10	0	26	0	0	0	0	72.22%
pred. 160,0V	2	0	0	1	74	0	0	0	96.10%
pred. 100,0V	0	0	0	0	0	0	0	0	0.00%
pred. 182,5V	0	0	0	0	0	0	0	0	0.00%
pred. 60,0V	0	0	0	0	0	0	0	0	0.00%
class recall	98.64%	82.14%	94.55%	61.90%	87.06%	0.00%	0.00%	0.00%	

Abbildung 6.9: Die durch 10-fache Kreuzvalidierung berechnete Genauigkeit

durchschnittliche Accuracy. Schon eine kleine Anzahl von Wiederholungen des k-means Algorithmus führt zu stabileren Ergebnissen.

Die Vorhersage eines unbekannten Stahlblocks erfolgt, indem die Messkurve jedes Sensors dem jeweils nächsten Zentroid des Modells zugeordnet werden. So erhält man auch zu dem unbekannten Stahlblock sieben Clusterzugehörigkeiten und somit einen Vektor mit den sieben Merkmalen, auf denen ein ID3 Baum trainiert wurde. Dann können mit dem ID3-Baum des Modells die Endabmaße vorhergesagt werden.

Das gesamte Verfahren kann durch 10-fache Kreuzvalidierung getestet werden. Man erreicht eine Accuracy von ca. 88%, also eine Fehlerrate von 12%, siehe dazu auch Abbildung 6.9. Im Vergleich dazu konnte mit einem Entscheidungsbaum, der auf aus den Zeitreihen extrahierten Merkmalen lernt, in [13] eine etwa gleich gute Accuracy von 90% erreicht werden.

Die von mir erreichte Accuracy lässt sich sicherlich weiter verbessern, wenn die Messreihen noch weiter vorverarbeitet werden anstatt sie nur zu z-normalisieren. So findet man beispielsweise durch Clustern der Temperaturdaten aus Sensor 102 fehlerhafte Zeitreihen, die statt einem Temperaturverlauf zwei aufeinander folgende Temperaturverläufe zeigen. Diese fehlerhaften Zeitreihen 'verbrauchen' dann schon einige der 6 Cluster. Da die Messdaten in den anderen Stationen des Prozesses sinnvoll sind, möchte ich die Daten nicht einfach aus der Trainingsmenge nehmen, da der Trainingsdatensatz ohnehin nicht besonders groß ist. Dies demonstriert eine weitere Anwendung des Clustern von Zeitreihen: Ohne sich alle Zeitreihen anzusehen bekommt man durch Clustern mit geeignetem k einen Überblick über die verschiedenen Formen von Zeitreihen im Datensatz.

Weiterhin liegen aus technischen Gründen nicht für alle Stationen des Produktionsprozesses Messdaten vor, insbesondere die Messdaten der zweiten Feinwalze sollten aber laut Domänenexperten für die Vorhersage der Endabmaße wichtige Informationen beinhalten.

Außerdem könnten in einem aufwändigeren Prozess die Parameter der k-means Clusterer

weiter optimiert werden. Beispielsweise könnte der Parameter k der k-means Clusterer systematisch mit dem 'Optimize Parameter' Operator von RapidMiner optimiert werden.

Insgesamt demonstriert das Experiment erfolgreich, dass sich die Information der Form von Zeitreihen eignet, um Lernaufgaben mit Zeitreihen zu lösen, ohne dass Features extrahiert werden. Somit wird kein Hintergrundwissen benötigt um geeignete Merkmale auszuwählen und auch der Aufwand der Vorverarbeitung nimmt ab. Als Nachteil gegenüber dem Clustering von Merkmalsvektoren muss allerdings die höhere Laufzeit angeführt werden. Der Unterschied wird hauptsächlich durch die Länge der zu clusternden Zeitreihen, die mit 100-1000 Messpunkten pro Zeitreihe wesentlich größer ist als die Dimension der Merkmalsvektoren, verursacht.

Kapitel 7

Zusammenfassung und Ausblick

Ziel dieser Arbeit war die Anpassung des k-means Algorithmus auf das Clustering von Zeitreihen mit Dynamic Time Warping Abstandsmaß. Der angepasste Algorithmus sollte die von Keogh in [10] vorgeschlagenen Ideen zum effizienten Ausführen von Querys mit DTW ausnutzen.

Dazu habe ich zuerst das k-means Clusteringverfahren und das Dynamic Time Warping eingeführt sowie einen Überblick über die von Keogh et al. in [10] entwickelte UCR-Suite gegeben.

Anschließend konnte die Anpassung des k-means Algorithmus entwickelt werden. Die größte Schwierigkeit war es, im k-means Algorithmus den Schritt des Bestimmens neuer Zentroide so anzupassen, dass eine Zeitreihe gefunden wird, die ein Cluster optimal beschreibt. Ich habe zwei eigene Möglichkeiten vorgeschlagen und drei bestehenden Möglichkeiten vorgestellt.

Diese verschiedenen Mittelungsverfahren konnten dann in einem Experimentalteil ausgiebig miteinander verglichen werden, in dessen Anschluss ich das von mir entworfene Verfahren 'Mittelung durch Projektion' empfehlen würde, da es effizient zu berechnen ist, die k-means Kostenfunktion gut minimiert und in keinem Experiment ein degeneriertes Clustering berechnet hat.

Schließlich konnte mit dem entwickelten Clusteringverfahren noch ein Problem mit echten Daten gelöst werden. Auf Messwerten aus der Produktion eines führenden deutschen Stahlherstellers konnte ein Modell gelernt werden, dass mit einer niedrigen Fehlerrate Endabmaße der produzierten Stahlblöcke aus den Messkurven des Produktionsprozesses vorhersagt.

Bei der Betrachtung der Stahlwerk-Messdaten ist aufgefallen, dass man eigentlich die Messreihen nicht sensorweise clustern möchte, sondern alle Messreihen eines Stahlblocks als Einheit auffassen möchte. Man möchte also Tupel von Zeitreihen clustern. Wenn man den Abstand zweier solcher Tupel als Summe der Komponentenweise ermittelten DTW-Abstände definiert, könnte man die in dieser Arbeit entwickelten Ideen , insbesondere die Mittelung einer Menge von Zeitreihen, auch auf das Problem des Clusterings von Tupeln von Zeitreihen übertragen. Es bleibt allerdings zu prüfen, ob sich ähnliche Optimierungen auch für dieses Problem finden lassen. Die hier vorgestellte LB_{keogh} lässt sich bei Tupeln nicht mehr effizient verwenden, da die Upper- und Lowerbands bei verschiedenen Längen von Zeitreihen häufig neu berechnet werden müssen. In dieser Arbeit entschärfe ich dieses Problem, indem die Zeitreihen der Länge nach sortiert werden. Dies ist bei Tupeln von verschieden langen Zeitreihen nicht so einfach möglich, man müsste eine Reihenfolge der Tupel finden, die die Anzahl der benötigten Neuberechnungen minimiert.

Weiterhin könnte die entwickelte RapidMiner-Extension noch um weitere Lernverfahren für Zeitreihen erweitert werden, insbesondere zur Klassifikation von Zeitreihen. Man könnte beispielsweise versuchen, die Modellkomplexität des 1-NN Klassifizierers durch Mittelung ähnlicher Zeitreihen oder durch eine klassenweise Clusteranalyse zu verringern.

Anhang A

Glossar

Begriff/Notation	Erläuterung
$p_{s,t}$	Ein Warping-Pfad der Zeitreihen s und t
$p^*_{s,t}$	Der optimale Warping-Pfad der Zeitreihen s und t
z_i	Je nach Kontext das Zentroid des i-ten Clusters C_i oder der i-te Wert der Zeitreihe z.
Kostenfunktion	Entweder Gesamtkosten des Clusterings (Siehe Gleichung 2.1 auf Seite 10) oder Kosten eines Warping-Pfades (Siehe Definition 5 auf Seite 15)
Gesamtkosten	Siehe Gleichung 2.1 auf Seite 10
$cost(C_1, ..., C_k, z_1, ..., z_k)$	Die Kostenfunktion des Clusterings. Siehe Gleichung 2.1 auf Seite 10
$d(x, y)$	Abstandsmaß $d : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$ berechnet einen Abstand zweier Objekte x, y.
M	Menge aller Objekte, die geclustert werden sollen. Meistens eine Menge von Zeitreihen.

Literaturverzeichnis

[1] Dynamic time warping. In *Information Retrieval for Music and Motion*, pages 69–84. Springer Berlin Heidelberg, 2007.

[2] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[3] Sanjoy Dasgupta. CSE 291: Unsupervised learning - lecture 2 — the k-means clustering problem. 2008.

[4] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, December 2006.

[5] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning.* Springer, corrected edition, July 2003.

[6] V. Hautamaki, P. Nykanen, and P. Franti. Time-series clustering by approximate prototypes. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4, 2008.

[7] Minsu Jang, Mun-Sung Han, Jae-hong Kim, and Hyun-Seung Yang. Dynamic time warping-based k-means clustering for accelerometer-based handwriting recognition. In KishanG. Mehrotra, Chilukuri Mohan, JaeC. Oh, PramodK. Varshney, and Moonis Ali, editors, *Developing Concepts in Applied Intelligence*, volume 363 of *Studies in Computational Intelligence*, pages 21–26. Springer Berlin Heidelberg, 2011.

[8] Keogh, Zhu, Hu, Xi, Wei, and Ratanamahatana. The UCR time series classification/clustering homepage. 2011.

[9] Eamonn Keogh and Chotirat Ann Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7:358–386, 2005.

[10] Eamonn J. Keogh, Thanawin Rakthanmanon, Bilson J. L. Campana, Abdullah Mueen, Gustavo E. A. P. A. Batista, M. Brandon Westover, Qiang Zhu, and Jesin Zakaria. Searching and mining trillions of time series subsequences under dynamic time warping. 2012.

[11] Sang-Wook Kim, Sanghyun Park, and Wesley W Chu. An index-based approach for similarity search supporting time warping in large sequence databases. In *Data Engineering, 2001. Proceedings. 17th International Conference on*, pages 607–614. IEEE, 2001.

[12] H. W. Kuhn and Bryn Yaw. The hungarian method for the assignment problem. *Naval Res. Logist. Quart*, pages 83–97, 1955.

[13] Daniel Lieber, Marco Stolpe, Benedikt Konrad, Jochen Deuse, and Katharina Morik. Quality prediction in interlinked manufacturing processes based on supervised and unsupervised machine learning. In *Proc. of the 46th CIRP Conf. on Manufacturing Systems*. Elsevier, 2013.

[14] Warissara Meesrikamolkul, Vit Niennattrakul, and ChotiratAnn Ratanamahatana. Shape-based clustering for time series data. In Pang-Ning Tan, Sanjay Chawla, Chin-Kuan Ho, and James Bailey, editors, *Advances in Knowledge Discovery and Data Mining*, volume 7301 of *Lecture Notes in Computer Science*, pages 530–541. Springer Berlin Heidelberg, 2012.

[15] Ingo Mierswa and Katharina Morik. Merkmalsextraktion aus audiodaten evolutionäre aufzucht von methodenbäumen. *Informatik Spektrum*, (5):381–388.

[16] Vit Niennattrakul and Chotirat Ann Ratanamahatana. On clustering multimedia time series data using k-means and dynamic time warping. In *MUE*, pages 733–738, 2007.

[17] Chotirat Ann Ratanamahatana and Eamonn Keogh. Making time-series classification more accurate using learned constraints. In *Proceedings of SIAM international conference on data mining*, pages 11–22. Lake Buena Vista, Florida, 2004.

[18] Chotirat Ann Ratanamahatana and Eamonn Keogh. Three myths about dynamic time warping data mining. In *Proceedings of SIAM International Conference on Data Mining (SDM'05)*, pages 506–510, 2005.

[19] Naoki Saito. *Local feature extraction and its applications using a library of bases*. PhD thesis, New Haven, CT, USA, 1994. AAI9523225.

[20] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 26(1):43–49, 1978.

Eidesstattliche Versicherung

Pfahler, Lukas 143926

Name, Vorname Matr.-Nr.

Ich versichere hiermit an Eides statt, dass ich die vorliegende Bachelorarbeit/~~Masterarbeit~~* mit dem Titel

Effizienteres k-means Clustering von Zeitreihen mit Dynamic Time Warping durch kaskadiertes Berechnen von unteren Schranken

selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Dortmund, _____

Ort, Datum Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -)

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird gfls. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin“) zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

Dortmund, _____

Ort, Datum Unterschrift