Quantitatives Frequent-Pattern Mining über Datenströmen

Daniel Klan, Thomas Rohe
Department of Computer Science & Automation
TU Ilmenau, Germany
{*first.last*}@tu-ilmenau.de

Abstract

Das Aufdecken unbekannter Zusammenhänge zählt zu einer der wichtigsten Aufgaben im Data Mining. Für das Problem des Frequent Pattern Mining über statischen Daten finden sich daher in der Literatur eine Vielzahl an Lösungen. Die Integration von Sensorik in nahezu jeden Lebensbereich führt allerdings zu Datenmengen, welche mittels der klassischen Verfahren zumeist nicht mehr bewältigt werden können. Ein Paradigmenwechsel hin zur Datenstrom-Verarbeitung ist oftmals unumgänglich. Ein interessantes Problem, welches im Zusammenhang mit der Verarbeitung von Sensordaten auftritt ist der prinzipiell stetige Wertebereich von Messungen. Die bekannten Lösungen sind für die Analyse von kontinuierlichen Daten über stetigen Wertebereichen nur bedingt geeignet. Im folgenden soll mit dem FP^2-Stream ein entsprechendes Verfahren für die Analyse quantitativer häufiger Muster über Datenströmen präsentiert werden.

1 Einleitung

Das Finden häufiger Muster (*frequent pattern mining*) ist Grundlage für eine Vielzahl von Data Mining Problemen (zum Beispiel der Korrelationsanalyse, oder dem Finden von Sequenzen oder Perioden innerhalb des Datenstromes). Die populärste Anwendung für das Frequent Pattern Mining ist das Assoziation Rule Mining, bei welcher aus Transaktionen Werte extrahiert und in Relation zueinander gesetzt werden. Eine typische Anwendung ist die Analyse von Supermarkt-Transaktionen, bei welcher das Kaufverhalten der Kunden analysiert wird. Ziel ist dabei die Detektion von Regeln, welche das Kaufverhalten einer repräsentativen Menge von Kunden wiederspiegeln und anhand derer anschließend die Verkaufsprozesse optimiert werden können. Eine mögliche Regel ist "Bier $\rightarrow$ Chips (10%)", welche besagt, dass 10 Prozent aller Kunden, die Bier gekauft haben auch Chips kaufen.

Wie die meisten Data Mining Verfahren, so wurde auch das Frequent Pattern Mining ursprünglich für statische Datenbestände entwickelt. In den letzten Jahren wurden diese Verfahren an das Datenstrom-Paradigma angepasst [5; 8]. In Interessantes Beispiel für die Anwendung der Assoziation Rule Mining über Datenströmen ist in [4] beschrieben. Die Autoren ersetzen dabei fehlende Werte in einem WSN anhand zuvor abgeleiteter Assoziationsregeln.

In einer Vielzahl von Anwendungen liegen die Daten zur Analyse nicht in Form kategorischer, sondern als quantitative Attribute vor. Die Anwendung der bekannten Frequent Pattern Mining Verfahren führt hierbei oftmals nicht zu den gewünschten Zielen bzw. der Berechnungsaufwand ist nicht vertretbar. Basierend auf dieser Feststellung entwickelten die Autoren in [11] erstmals ein Verfahren für das Finden von Regeln, welche sowohl mit kategorischen Attributen als auch mit quantitativen Attributen umgehen kann. Das vorgestellte Verfahren bildet dabei quantitative auf kategorische Attribute ab. Die quantitativen Attribute werden im wesentlichen durch Intervalle dargestellt, welche eine Menge von numerischen Werten aufnehmen können. Auf Basis dieser Definition präsentierten die Autoren einen angepassten Apriori-Algorithmus, welcher den Suchraum zerlegt (diskretisiert) und anschließend durch Kombination der Teil-Intervalle Regeln findet.

Es existieren eine Vielzahl von Anwendungsszenarien in denen die Extraktion quantitativer Regeln bzw. das Finden quantitativer Frequent Itemsets über Datenströmen von Interesse sind (z.B. Gebäudeüberwachung). Die in der Literatur präsentierten Verfahren beziehen allerdings sich ausschließlich auf das Finden von Regeln über statischen Datenbeständen. Für ein Verarbeitung von Datenströmen sind diese ungeeignet. Zudem weisen die meisten Verfahren Beschränkungen auf, derart dass quantitative Attribute lediglich im Regel-Kopf auftreten dürfen.

Die Arbeit ist im weiteren wie folgt aufgeteilt. Zunächst folgt in Abschnitt 2 ein kurzer Überblick über existierende Arbeiten zum Thema. Anschließend wird in Abschnitt 3 auf für das Verständnis notwendige Grundlagen eingegangen. In Abschnitt 4 folgt eine ausführliche Beschreibung des entwickelten Verfahrens, welche in Abschnitt 5 evaluiert werden soll.

2 Verwandte Arbeiten

Eines der ersten Verfahren zum Finden von häufigen Mustern war das von Aggrawal et. al präsentierte *Apriori*-Verfahren [1]. Die Grundlage des Ansatzes bildet die Annahme, dass sich häufige Muster ausschließlich aus häufigen Teilmustern zusammensetzen können. Basierend auf dieser Heuristik erzeugt das Verfahren alle möglichen $k-itemset$ Kandidaten aus der Menge aller $k-1$-itemset Kandidaten. Aufgrund der wiederholten Analyse der Daten handelt es sich hierbei um ein *multi-pass* Verfahren, welches seine Anwendung insbesondere in der Verarbeitung von endlichen Datenmengen findet.

Auf Basis des Apriori-Algorithmus wurden eine Vielzahl an weiteren Algorithmen entwickelt, welche unter anderem die Performance-Steigerung oder die Adaption an Datenströme zum Ziel hatten. So stellen die Autoren in [3] ein effizientes Verfahren zum Finden von optimalen links-

seitigen quantitativen Regeln vor. Der Schwerpunkt ist dabei die Entwicklung eines Verfahrens zur Berechnung der optimalen Bereiche in linearer Zeit unter der Annahme das die Daten sortiert sind. Für denn Fall, dass eine Sortierung der Daten nicht möglich ist, wurde zusätzlich ein randomisiertes Verfahren präsentiert, welches die Daten in Buckets teilt und anschließend auf diesen die Suche durchführt. Die Autoren in [9] präsentierten mit dem DHP (*Direct Hashing and Pruning*) ein Verfahren, dessen Ziel die Optimierung der Anzahl an Kandidaten ist. Das Verfahren setzt dabei auf eine Hash-Tabelle für die Identifikation von Kandidaten mit einem Support größer dem geforderten.

Han et. al präsentieren in [5] mit dem FP-Tree (*frequent pattern tree*) einen Prefix-Baum zur effizienten Speicherung häufiger Muster. Zusätzlich zu dieser Datenstruktur stellen sie mit dem FP-Growth Algorithmus ein Verfahren zum Finden von häufigen Mustern auf Basis des FP-Tree vor. Im Gegensatz zum Apriori Verfahren verzichtet das FP-Growth Verfahren vollständig auf die Generierung von Kandidaten. Auch sind lediglich zwei Durchläufe über die zu prüfende Datenmenge ausreichend.

FP-Mining über Datenströmen stellt eine besondere Herausforderung dar. Neben dem in [5] beschriebenen FP^2-Stream existieren auch eine Vielzahl weiterer Lösungen. Der in [12] vorgestellte Compact Pattern Tree (CPT) zum Beispiel verwendet gleitende Fenster (analog dem FP^2-Stream und dem DSTree [7]). Die Effizienz des Verfahrens wird dabei durch eine zusätzliche Restrukturierungs Phase erreicht, während der der CPT in Abhängigkeit der Itemset-Häufigkeiten mit dem Ziel einer hohen Kompaktheit umsortiert wird. In [8] beschreiben die Autoren neben dem Sticky-Sampling und Lossy-Counting mit BTS(Buffer-Trie-SetGen) ein Verfahren, welches die Häufigkeiten von Itemsets mittels einer Gitter-Strukur speichert.

Die erste Arbeit die sich mit dem Problem der quantitativen Muster Erkennung beschäftigt stammt von Aggrawal [11]. Das vorgestellte Verfahren basierte dabei im wesentlichen auf einem Apriori-Verfahren, welches die partitionierten Wertebereiche der einzelnen Attribute in geeigneter Weise kombiniert.

Die Autoren in [2] betrachten quantitativ/kategorische Assoziationsregeln. Eine Regel wird signifikant angesehen, wenn sich deren Mittelwert in Relation zur Menge aller Transaktionen ohne dieses Regel als signifikant herausstellt. Zur Berechnung des Signifikanzlevels ziehen sie dabei den Steiger Z-Test heran. Als Null-Hypothese nehmen sie an, das die Mittelwerte beider Teilmengen gleich sind. Wird diese Null-Hypothes abgelehnt (mit einer Konfidenz von 95%), so wird die gefundene Regel als signifikant unterschiedlich von der Restmenge der Transaktionen angenommen.

3 Vorbetrachtung

Zunächst müssen verschiedene Begrifflichkeiten eingeführt werden, welche für das Verständnis des im weiteren präsentierte Verfahren notwendige sind.

Im folgenden bezeichnet a_i ein Attribut. $a_i\langle v\rangle$ entspricht dem zum Attribut a_i gehörender Wert v. Das Tripel $(a_i, l, r) = a_i(l, r)$ bezeichnet das *quantitative Attribut* a_i, welches Werte $v_1, v_2, ...$ im Intervall (l, r) aufnimmt (l bezeichnete die linke und r die rechte Intervallgrenze). Ein *kategorisches Attribut*, d.h. ein einelementiges Attribut, kann durch ein quantitatives Attribut repräsentiert werden, dessen linke und rechte Intervallgrenze gleich sind, d.h. $a_i(l, l)$ bezeichnet das kategorische Attribut, welches nur den Wert

	a_1	a_2		a_1	a_2
t_1	22	100	t_6	21.5	0
t_2	22.5	-	t_7	21	-
t_3	-	100	t_8	-	100
t_4	22.5	-	t_9	22	-
t_5	21.5	0	t_{10}	22.5	100

Abbildung 1: Beispiel Transaktionen

itemset	$freq$	$supp$
$\{\langle a_1(20, 21.5]\rangle\}$	3	0.3
$\{\langle a_1(20, 23]\rangle\}$	8	0.8
$\{\langle a_2(0, 50]\rangle\}$	4	0.4
$\{\langle a_2(50, 100]\rangle\}$	4	0.4
$\{\langle a_1(20, 21.5]\rangle, \langle a_2(0, 50]\rangle\}$	2	0.2
$\{\langle a_1(20, 23]\rangle, \langle a_2(0, 50]\rangle\}$	2	0.2
$\{\langle a_1(20, 23]\rangle, \langle a_2(50, 100]\rangle\}$	2	0.2

Abbildung 2: Beispiel Itemsets

l repräsentiert. Das Attribut $a_i(l, r)$ wird im folgenden als Item bezeichnet. $\mathcal{I}$ bezeichnet die Menge aller Items. Die Menge $X = \{a_1(l, r), ..., a_k(l, r)\} \subseteq \mathcal{I}$, with $a_i \neq a_j$ bezeichnet ein Itemset (oder auch k-Itemset).

D bezeichnet eine Menge von Transaktionen. Eine Transaktion $d_{\Delta t} \in D$ ist eine Menge von Attributwerten $a_i\langle v\rangle$ im Zeitintervall Δt. Jede Transaktion $d_{\Delta t} = \{a_i\langle v\rangle, ..., a_j\langle v\rangle\}$ kann auf ein Itemset $\mathcal{I}$ abgebildet werden, d.h. für jeden Wert $a_i\langle v\rangle$ existiert ein Item $a_i(l, r)$ in $\mathcal{I}$ mit $a_i\langle v\rangle \in a_i(l, r)$.

Die Häufigkeit $freq(X, D)$ eines Itemsets X bezeichnet die Anzahl der Transaktionen D im Zeitintervall Δt, die auf das Itemset abgebildet werden können. Der $supp(X, D)$ eines Itemsets ist definiert als der prozentuale Anteil an Transaktionen, welcher auf das Itemset entfällt. Üblicherweise sind nur häufige Itemsets von Interesse. Ein Itemset wird als häufig bezeichnet, wenn dessen Häufigkeit einen vordefinierten Schwellwert $minsupp$ überschreitet.

Signifikanz und Informationsdichte Ziel des quantitativen Frequent Itemset Mining ist das Finden von zusammenhängenden Items, deren Informationsgehalt sich *signifikant* von dem aller anderen Items unterscheidet. Ein quantitatives Item ist genau dann signifikant, wenn dessen Informationsdichte größer ist, als die Informationsdichte der alternativen Items. Die Informationsdichte eines Items $a_i(l, r)$ ist dabei wie folgt definiert

$$density(a_i(l, r)) = \frac{freq(a_i(l, r))}{dist(l, r)} \tag{1}$$

wobei $dist(l, r) = abs(l - r)$ die Distanz zwischen den beiden Intervallgrenzen des Items bezeichnet. Entsprechend dieser Definition wird ein Itemset $\mathcal{I}$ genau dann als signifikant bezeichnet, wenn alle Items dieses Itemsets signifikant sind.

Generalisierung Ein Itemset $\hat{X}$ wird genau dann als *Generalisierung* von X bezeichnet, wenn $\hat{X}$ aus den selben Items wie X besteht und es gilt

$$\forall a_i(l, r) \in X : a_i(l, r) \in X \wedge a_i(l', r') \in \hat{X} \Rightarrow l' \leq l \leq r \leq r'$$

D.h. alle Transaktion, welche sich auf X abbilden lassen, können ebenso auf $\hat{X}$ abgebildet werden. Im folgenden soll ein kurzes Beispiel die eben beschriebenen Zusammenhänge verdeutlichen.

Beispiel 1 *Es wird ein Datenstrom angenommen bestehend aus zwei Attributen a_1 (Temperatur) und a_2 (Bewegung) angenommen. Im Zeitintervall Δt wurden die in Tabelle 3 dargestellten 10 Transaktionen festgestellt. Weiterhin werden die folgenden Items angenommen:*

$$a_1(20, 21.5], a_1(20, 23], a_2(0, 50] \textit{ und } a_2(50, 100]$$

Basierend auf diesen (quantitativen) Items lassen sich über den in Tabelle 3 abgebildeten Transaktionen die in Tabelle 3 dargestellten (quantitativen) Itemsets ermitteln. Die Tabelle zeigt die entsprechenden Häufigkeiten und den Support, den die einzelnen Itemsets aufweisen. Es lassen sich die folgenden Dichten für die Items ermitteln: $density(a_1(20, 21.5])) = 2$, $density(a_1(21.5, 23]) = 3.3$ und $density(a_1(20, 23]) = 2.7$. Bei dem einelementigen Itemset $\{\langle a_1(20, 23]\rangle\}$ handelt es sich um eine Generalisierung der Itemsets $\{\langle a_1(20, 21.5]\rangle\}$ und $\{\langle a_1(21.5, 23]\rangle\}$.

4 FP^2-Stream

Die meisten in der Literatur zu findenden Verfahren [11; 2], welche quantitative Attribute betrachten, zerlegen den Wertebereich für ein Attribut in äquidistante Intervalle, welche anschließend derart miteinander kombiniert werden, dass sie den geforderten Kriterien (minimaler Support und Signifikanz) genügen. Zu den wesentlichen Problemen dieser *bottom-up* Strategie zählen dabei das Finden einer geeignete Zerlegung [11] bzw. das möglichst effiziente generieren zusammenhängender Items.

Die Zerlegung gefolgt von einer kostenintensiven Item-Rekonstruktion ist auf Datenströme nicht bzw. nur beschränkt anwendbar. Alle Verfahren, welche das Prinzip der Kombination von Teilintervallen einsetzen, basieren dabei auf dem Apriori-Algorithmus, welcher mehrere Durchläufe benötigt und daher für die Analyse über Datenströmen nur bedingt geeignet ist.

Im folgenden soll der FP^2-Stream vorgestellt werden, ein Speicher-effizientes Verfahren, welches für die Analyse über Datenströmen geeignet ist. Die Itemsets werden beim FP^2-Stream in einem Prefix-Baum (ähnlich dem FP-Tree von Han et al. [5]) verwaltet. Das Ziel des vorgestellten Verfahrens ist anschließend die kontinuierliche Verfeinerung der Items (*top-down*-Strategie) und den daraus aufgebauten Itemsets mit dem Eintreffen neuer Transaktionen, so dass diese den geforderten Kriterien genügen.

Im weiteren soll zunächst die Grundlegende Datenstruktur beschrieben werden, anschließend folgt eine Beschreibung der Algorithmen zum Einfügen von Transaktionen und zum Optimieren der Datenstruktur.

4.1 Datenstruktur

Der FP^2-Stream verwaltet die häufigsten Muster in einem Prefix-Baum. Eine Header-Tabelle enthält alle Itemsets, welche sich gegenwärtig im Prefix-Baum befinden. Der Pfad von der Wurzel bis zu einem Knoten im Prefix-Baum repräsentiert ein Itemset. Zusätzlich ist in jedem Knoten des Baumes ein Zeitfenster eingebettet, welches die Häufigkeiten in den letzten k Zeiträumen aufnimmt. Im FP^2-Stream werden hierzu gleitende Fenster eingesetzt.

Weiterhin sind alle Knoten, welche das gleiche Item repräsentieren untereinander über Listen miteinander verbunden. Das entsprechende Item der Header-Tabelle verweist dabei auf das erste Element dieser Liste. Jedes Item der Header-Tabelle enthält zusätzlich ein *Equi-Width Histogramm*, welches einen approximativen Überblick über die Häufigkeitsverteilung der zuletzt eingefügten Werte in die Items gibt. Abbildung 3 zeigt einen Beispiel-FP^2-Stream. Dem Beispiel liegen die in Tabelle 3 beschriebenen Transaktionen zugrunde. Das Attribut a_2 wurde bereits einer Verfeinerung unterzogen.

Input: Menge von Transaktionen D
Output: Menge von Häufigen Itemsets

1 Initialisiere FP^2-Tree als leer;
2 Stelle min und max für jedes Attribut a_i fest und lege die entsprechenden knoten im FP^2-Tree an;
3 Füge die Transaktionen von Batch b_0 in FP^2-Tree ein;
4 Übernehme Knotengrenzen und Häufigkeiten aus dem FP^2-Tree in den FP^2-Stream;
5 **while** *Batch* b_i, $i > 0$ **do**
6 Initialisiere FP^2-Tree mit den Knoten und Intervallgrenzen des FP^2-Stream;
7 Sortiere alle Transaktionen aus aktuellem Batch in FP^2-Tree ein (falls notwendig, füge neue Knoten hinzu bzw. erweitere existierende Knoten);
8 Übertrage alle Knoten aus dem FP^2-Tree in den FP^2-Stream;
9 Führe eventuell notwendige Split Operationen auf dem FP^2-Stream aus;
10 Führe eventuell notwendige Merge Operationen auf dem FP^2-Stream durch;
11 Lese alle häufigen Itemsets aus dem FP^2-Stream aus (FP^2-Growth);

Algorithm 1: FP^2-Stream Algorithmus

4.2 Einfügen neuer Transaktionen

Das Einfügen neuer Transaktionen in den FP^2-Stream erfolgt Batchweise (ein Batch b bezeichnte die Zusammenfassung einer Menge von $|b|$ Transaktionen). Neue Transaktionen werden beim Einfügen nicht direkt in den FP^2-Stream integriert, sondern zuvor in einen FP^2-Tree eingefügt. Der FP^2-Tree entspricht dabei im wesentlichen dem FP^2-Stream, wobei die Knoten jedoch nicht über Zeitfenster verfügen (der FP^2-Tree verwaltet lediglich die zu einem Batch gehörenden Musterhäufigkeiten). Die Implementierung des FP^2-Tree erfolgt in Form von "Schattenknoten", welche in die Knoten des FP^2-Stream integriert werden. Somit fallen für das Anlegen des FP^2-Tree keine zusätzliche Kosten an. Es muss lediglich beim ersten Einfügen einer Transaktion in einen neuen Batch der entsprechende "Schattenknoten" angelegt werden.

Analog dem FP-Stream wird das Einfügen des ersten Batches b_0 getrennt von der Behandlung aller weiteren Batches betrachtet. Zum Zeitpunkt des Einfügens von b_0 in den FP^2-Stream ist kein Wissen über die genaue Verteilung der Stromdaten vorhanden bzw. es stehen lediglich die Informationen aus dem ersten Batch zur Verfügung. Zunächst wird daher für jedes Attribute i ein Item $a_i(min, max)$ derart angelegt, dass min dem minimalen Wert in b_0 und max dem maximalen Wert in b_0 des Attributes a_i im ersten Batch entspricht. Die Items werden anschließend nach der Häufigkeit ihres Auftretens im ersten Batch sortiert und in den FP^2-Tree eingefügt. Häufige Items werden Wurzelnah eingefügt. Da im ersten Batch ein Attribut den kompletten Wertebereich eines Datenstromes überdeckt, kann es ausschließlich durch das vollständige Fehlen von Werten innerhalb von Transaktionen zu Unterschieden in den Häufigkeiten der einzelnen Attribute kommen.

Nachdem der erste Batch erfolgreich eingefügt wurde,

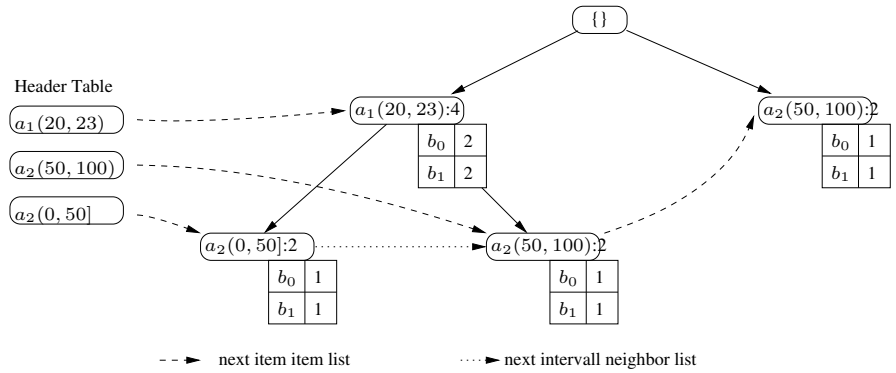


Abbildung 3: Beispiel FP^2-Stream

werden alle weiteren Batches gleich behandelt und folgt nachstehendem Schema:

- Es existiert bereits ein Knoten, welcher das Itemset repräsentiert. Die Frequenz des entsprechenden Knotens im FP^2-Tree wird um 1 inkrementiert.
- Es existiert kein Knoten, welcher das Itemset repräsentiert. Es muss ein neuer Knoten im FP^2-Tree angelegt werden:
  - Der neu anzulegende Knoten wird sowohl auf der linken, als auch auf der rechten Seite von existierenden Knoten eingeschlossen: Als Intervallgrenzen für den neuen Knoten werden die Grenzen der benachbarten Knoten gewählt. D.h. es wird ein Knoten $a_i(r, l')$ zwischen den beiden begrenzenden Knoten $a_i(l, r)$ und $a_i(l', r')$ angelegt.
  - Der einzufügende Wert über- bzw. unterschreitet alle bisher eingefügten Werte: Existiert ein Knoten im FP^2-Tree, welches noch keinem Knoten im FP^2-Stream entspricht und dessen Intervallgrenzen sich derart erweitern lassen, dass es den Wert aufnehmen kann, dann werden die Grenzen dieses Knotens angepasst und die Frequenz entsprechend inkrementiert. Andernfalls muss ein neuer Knoten mit dem neuen Wert als Minimum bzw. Maximum als Grenze angelegt werden.

Wurden ausreichend Transaktionen in den FP^2-Tree eingefügt, dann kann dieser in den FP^2-Stream integriert werden. Zunächst werden hierzu alle Knoten des FP^2-Tree entfernt, welche dem vorab definierten Schwellwert ϵ nicht genügen. Anschließend werden die Häufigkeiten der einzelnen Knoten in den FP^2-Stream übernommen. Hierzu werden in die Zeitfenster aller Knoten des FP^2-Stream neue Zeitslots eingeführt. Sollten im FP^2-Stream Knoten vorhanden sein, welche durch die Integration des FP^2-Tree keine Aktualisierung erfahren, so sind deren Häufigkeiten für diesen Zeitslot 0. Sind während der Verarbeitung eines Batches neue Knoten im FP^2-Tree hinzugekommen, so müssen diese ebenfalls in den FP^2-Stream übernommen werden.

Wurden die Transaktionen eines Batches erfolgreich in den FP^2-Stream eingefügt, dann wird anschließend geprüft, inwieweit sich Items verfeinern lassen um eine möglichst hohe Informationsdichte zu erhalten. Die Verfeinerung ist hierbei ein zweistufiger Prozess. In einem ersten Schritt werden die Items verfeinert, wenn deren Füllgrad zu hoch ist. Der zweite Schritt ist das Mischen von Items, wenn sich diese generalisieren lassen. Beide Prozesse sollen im weiteren beschrieben werden.

4.3 Item-Split

Als wesentliches Kriterium für schlecht approximierte Items wird die Dichteverteilung innerhalb eines Items herangezogen. Sind die in ein Item eingefügten Werte ungleichmäßig verteilt, dann wurden die Grenzen für dieses Item schlecht gewählt. Die Bestimmung der Ungleichverteilung erfolgt dabei über die Schiefe den in der Header Tabelle mitgeführten *Equi-Width-Histogrammen*. Die Schiefe eines Item $a_i(l, r)$ über einem Histogramm lässt sich wie folgt bestimmen:

$$s(a_i(l,r))_n = \frac{\max\{h(a_i(l,r),j)\}_{j=1}^{n} - \min\{h(a_i(l,r),j)\}_{j=1}^{n}}{\sum_{j=1}^{n} h(a_i(l,r),j)}$$

wobei n die Anzahl der Buckets in den Histogrammen bezeichnet. $h(a_i(l, r), j)$ bezeichnet die Häufigkeit im j-ten Bucket des Histogramms für das Item $a_i(l, r)$.

Überschreitet die Schiefe innerhalb des Items $a_i(l, r)$ einen vorab definierten Schwellwert $maxskew$, d.h. $s(a_i(l, r)) > maxskew$, dann wird das Item in zwei disjunkte Items gesplittet. O.B.d.A. werden im folgenden Items immer in Items mit links offenem Intervall zerlegt. Für den Split wird ein Median-basierter Ansatz verwendete, dessen Ziel die Berücksichtigung der realen Dichteverhältnisse innerhalb eines Items ist. Hierzu wird der Median über die einzelnen Buckets der Histogramme bestimmt. Der Split erfolgt anschließend auf Basis des Mittelpunktes des Median-Buckets, d.h. $a_i(l, r)$ wird in die beiden Items $a_i(l, median/2]$ und $a_i(median/2, r)$ zerlegt, wobei $median$ den Median bezeichnet.

Bei einem Itemsplit müssen alle Knoten im Baum, welche dieses Item repräsentieren ebenfalls gesplittet werden. Sind unterhalb eines Knotens, welcher ein Item repräsentiert das gesplitted wird, weitere Knoten, so werden diese Teilbäume kopiert und als neue Teilbäume in die neu entstehenden Knoten eingebunden. Die Häufigkeit des zu splittenden Items wird beim Überführen in die neuen Items halbiert (die genaue Verteilung der Daten in den einzelnen Knoten ist unbekannt, weswegen der Einfachheit halber ein Gleichverteilung angenommen wird). Ein Beispiel soll den Itemsplit verdeutlichen.

Beispiel 2 *Die Häufigkeit des Items $\langle a_1(20, 23]\rangle$ aus dem vorigen Beispiel genügt den definierten Bedingungen. Beim Itemsplit auf Basis der Intervall-Halbierung wird das Item in die beiden Subitems $\langle a_1(20, 21.5]\rangle$ und $\langle a_1(21.5, 23]\rangle$ geteilt. Es entstehen somit die beiden Itemsets $\{\langle a_1(20, 21.5]\rangle, \langle a_2(50, 100]\rangle\}$ und $\{\langle a_1(21.5, 23]\rangle, \langle a_2(20, 100]\rangle\}$ welche jeweils eine geschätzte Häufigkeit von 2 besitzen. Abbildung 4 zeigt den entsprechenden FP^2-Stream aus Abbildung 3 nach dem Split des Items $\langle a_1(20, 23]\rangle$.*

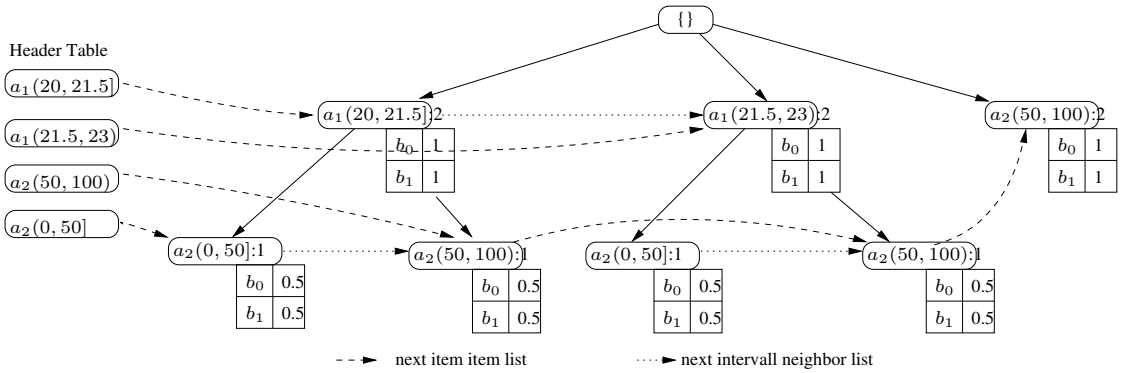


Abbildung 4: Split node $a_1(20, 23]$

Eine beliebige Verfeinerung der Intervalle ist aus Gründen einer effizienten Datenhaltung nicht sinnvoll. Somit ergibt sich die Fragestellung inwieweit Intervalle verfeinert werden. In [11] präsentieren die Autoren eine Untersuchung zur Anzahl an maximal notwendige Partitionen (Basisintervallen), welche für die anschließende Rekombination zu erzeugen sind und dabei einen möglichst geringen Informationsverlust aufweisen. Als Maß für den durch die Generalisierung entstehenden Informationsverlust führen sie die *Partial-Completeness* ein. Als K-Partial-Completeness wird dabei der maximale Support bezeichnet (als K-vielfaches des minimalen Support), den ein Itemset aufweisen darf. Basierend auf der Partial-Completeness haben die Autoren gezeigt, dass im Fall einer Partitionierung in Basisintervalle gleicher Größe, maximal $\frac{2 \cdot \alpha}{minsupp \cdot (K-1)}$. Intervalle notwendig sind. α bezeichnet dabei die Anzahl an quantitativen Attributen. Daraus lässt sich die minimale Intervallgröße für jedes Datenstromattribut a_i ableiten. Zum Zeitpunkt des ersten Erstellens des FP^2-Stream (Batch b_0) wird für jedes Attribut $a_i(min, max)$ bestimmt. Damit ergibt sich die minimale Intervallgröße $minIntSize$ entsprechend

$$minIntSize = \frac{|max - min|}{\frac{2 \cdot \alpha}{minsupp \cdot (K-1)}}. \quad (2)$$

Beim späteren Eintreffen von Transaktionen mit Werten, welche das rechts- bzw. linkseitige Extrema erweitern muss das minimale Intervall entsprechend angepasst werden.

Zusätzlich zu den Knoten müssen auch die in den Knoten eingebetteten Fenster angepasst werden. Jeder der beiden durch den Split neu entstandenen Knoten erhält hierzu die Hälfte der Häufigkeitswerte des Original-Items. Um später beim FP^2-Growth auf einfache Weise das bisherige Split-Verhalten von Items nachvollziehen zu können wird bei einem Split in jedem Zeitfensterslot ein Verweis auf das Fenster angelegt, das bei dem Split abgeteilt wurde. Um weitere Splits bzw. Merges möglichst effizient gestalten zu können verweist das letzte Fenster in der Liste auf das erste Listenelement, so dass ein Ring entsteht. Der Ring ermöglicht anschließend eine schnelle Suche auch ohne die Verwendung einer doppelt verketteten Liste.

4.4 Item-Merge

Items, welche nahezu die gleiche Informationsdichte aufweisen, können ohne weiteren Informationsverlust kombiniert (generalisiert) werden. Die Generalisierung von Items führt somit zu einer kompakteren Datenstruktur.

Zwei Items im FP^2-Stream $a_i(l, r)$ und $a_i(l', r')$ können genau dann zu dem Item $a_i(l, r')$ zusammengefasst werden, wenn für alle Knoten die diese Items repräsentieren folgenden Bedingungen erfüllt werden:

1. Die Items $a_i(l, r)$ und $a_i(l', r')$ sind *direkte Nachbarn*, d.h. es gilt $r = l'$.
2. Benachbarte Knoten, welche die Items $a_i(l, r)$ und $a_i(l', r')$ repräsentieren besitzen den gleichen Präfix, d.h. sie verfügen über den gleichen Elternknoten im FP^2-Stream.
3. $density(a_i(l, r)) \sim density(a_i(l', r'))$

1. stellt sicher, dass es sich bei den Verbundkandidaten um Intervallnachbarn handelt. Ausschließlich direkte Nachbarn können miteinander verbunden werden. 2. garantiert, dass alle von dem Item-Merge betroffenen Itemsets ebenfalls verbunden werden können. 3. stellt sicher, dass die zu verbindenden Items über nahezu den gleichen Informationsgehalt verfügen, d.h. das es durch die Generalisierung zu keinem Informationsverlust kommt.

Algorithmus 1 führt den Merge umgehend nach dem Split aus. Wurde ein Split auf einem Item durchgeführt, dann verfügen die beiden resultierenden Items über die gleiche Dichte. Erst durch das Einfügen neuer Batches setzt sich die Schiefe, welche zuvor für den Split notwendigerweise festgestellt wurde, auch in den neuen Items durch. Aus diesem Grund ist eine sofortige Rekombination zuvor erst geteilter Items für die nächsten k Batches nicht sinnvoll.

4.5 Prunning und Reorganisation

Durch das Splitten von Items wird der FP^2-Stream kontinuierlich vergrößert. So resultiert zum Beispiel der Split des Wurzelknotens in einer Verdoppelung aller Knoten im Baum. Um diesen dennoch möglichst klein und damit die Verarbeitung effizient zu gestalten werden zwei verschiedene Ansätze verfolgt:

- (i) Itemsets, welche nicht mehr häufig sind und in den nächsten Zeitschritten auch nicht mehr häufig werden können, werden aus der Datenstruktur entfernt werden.
- (ii) Es erfolgt eine Reorganisation der Baumstruktur um eine möglichst hohe Kompaktheit zu gewähren (ähnlich dem CPT [12]).

Item-Rekonstruktion und Itemset Exktraktion

Zwar werden im FP^2-Stream häufige Itemsets in einer kompakten Darstellung gespeichert, es wird aber nicht garantiert, dass die Extraktion der häufigen Muster effizient geschieht. Das Herauslösen der Frequent Pattern stellt dabei ein kombinatorisches Problem dar. Mit der Entwick-

lung des FP-Tee präsentierten Han et al. in [5] den FP-Growth, ein Verfahren zur Extraktion aller im FP-Tree gespeicherten häufigen Muster. Im Folgenden wird ein für den FP^2-Stream angepasstes Growth-Verfahren zum extrahieren der häufigen Muster beschrieben.

Bevor die häufigen Itemsets extrahiert werden können sind zwei wesentliche Verarbeitungsschritte notwendig. In einem ersten Schritt wird aus dem FP^2-Stream ein FP^2-Tree extrahiert. Hierzu wird die Summe über alle in dem Fenster eines Knotens enthaltenen Häufigkeiten als aktueller Wert für einen Knoten des FP^2-Tree herangezogen. Aufgrund des kontinuierlichen Teilens der Knoten, unter Annahme einer Gleichverteilung der einzelnen Werte in den Intervallen, handelt es sich bei den Häufigkeiten in den Knoten des FP^2-Stream zumeist nur um approximierte Werte. Lediglich Knoten, welche zuvor nicht gesplittet wurden, weißen genaue Häufigkeitswerte auf. Um dies auch für zuvor geteilte Knoten zu erreichen sind die Fensterslots aller Knoten, welche ursprünglich von dem selben Knoten abstammen untereinander über Ringlisten miteinander verbunden (siehe Item-Split). Unter Verwendung dieser kann die genaue Häufigkeit für den ursprünglichen Knoten wiederhergestellt werden (die Summe über alle Knoten einer Ringliste).

Der durch den Extraktionsprozess erzeugte FP^2-Tree enthält im allgemeinen quantitative Items, deren Support nicht $minsupp$ genügt. Im nächsten Schritt werden durch die Rekombination von benachbarten Intervallen Items erzeugt, welche $minsupp$ genügen. DIn der Literatur lassen sich verschiedene Interessantheitsmaße für das Erzeugen der Items finden [10; 13]. Beim FP^2-Stream bzw. FP^2-Tree findet die Informationsdichte der Items Verwendung. D.h. das Item mit der höchsten Informationsdichte wird so lange mit benachbarten Items (dem jeweils dichtesten direkt benachbarten Item) verbunden, bis der geforderte minimale Support erreicht wird. Werden per Definition mehr als ein häufiges Item pro Datenstromattribut gefordert, dann wird dieses Verfahren auf die verbliebenen nicht-häufigen Items angewandt.

Nachdem der FP^2-Tree erzeugt wurde und die Items den definierten Bedingungen genügen, können aus diesem häufige Itemsets extrahiert werden. Aus Gründen der effizienten Auswertung wird hierzu auf das *Top-Down* Verfahren nach [14] zurückgegriffen.

5 Evaluierung

Im folgenden soll die Funktionsweise des vorgestellten Verfahrens gezeigt werden. Zu diesem Zweck wurde das Verfahren in Form eines Operators in das Datenstrom-Managementsystem AnduIN [6] integriert. Neben einer Vielzahl an einfachen Operationen (Filter, Projektion, Verbund, Aggregation) unterstützt AnduIN zusätzlich auch die Integration komplexer Operatoren in Form von Synopsen-Operatoren. Unter einem Synopsen-Operator wird dabei ein Operator verstanden, welcher sowohl über eine In-Memory Datenzusammenfassung, als auch über notwendige Algorithmen für das Einfügen von Daten und deren Auswertung verfügt. Das hier präsentierte Verfahren wurde als ein solcher Synopsenoperator integriert.

Zunächst soll die grundlegende Funktion des Verfahrens präsentiert werden. Hierzu wurden 3 Datenströme zu je 5000 Datenpunkten erzeugt. Ein Datenstrom entspricht dabei den Daten eines Attributes, d.h. die Analyse erfolgt im weiteren über Attributen. Die erzeugten Werte sind dabei standardnormalverteilt. Abbildung 5(a) zeigt exemplarisch den erzeugten Datenstrom für ein Attribut.

Aus diesen drei Datenströmen sollen nun zusammenhängende Itemsets extrahiert werden, wobei pro Attribut ein Item erzeugt werden soll. Im ersten Test sollen Items bzw. Itemsets extrahiert werden, deren Support $minsupp$ mindesten 0.25 ist. Es wurde mit einer Batchgröße $|b|$ = 100, einer Fenstergröße N von 5, sowie Histogrammen mit 10 Buckets d und einer maximalen Schiefe $maxskew$ von 0.2 getestet. Abbildung 5(b) zeigt beispielhaft die zeitliche Entwicklung des über dem Datenstrom aus Abbildung 5(a) entwickelten Items. Graue Buckets repräsentieren Items (pro Zeitschritt jeweils eine zusammenhängende Box). Die schwarzen Rahmen entsprechen den Knoten im FP^2-Stream, die über diesem Attribut existieren. Weiß gerahmte Buckets sind somit Knoten im FP^2-Stream, welche nicht häufig sind. Graue Balken, die aus mehreren Buckets bestehen sind im FP^2-Stream über mehrere Knoten verteilt.

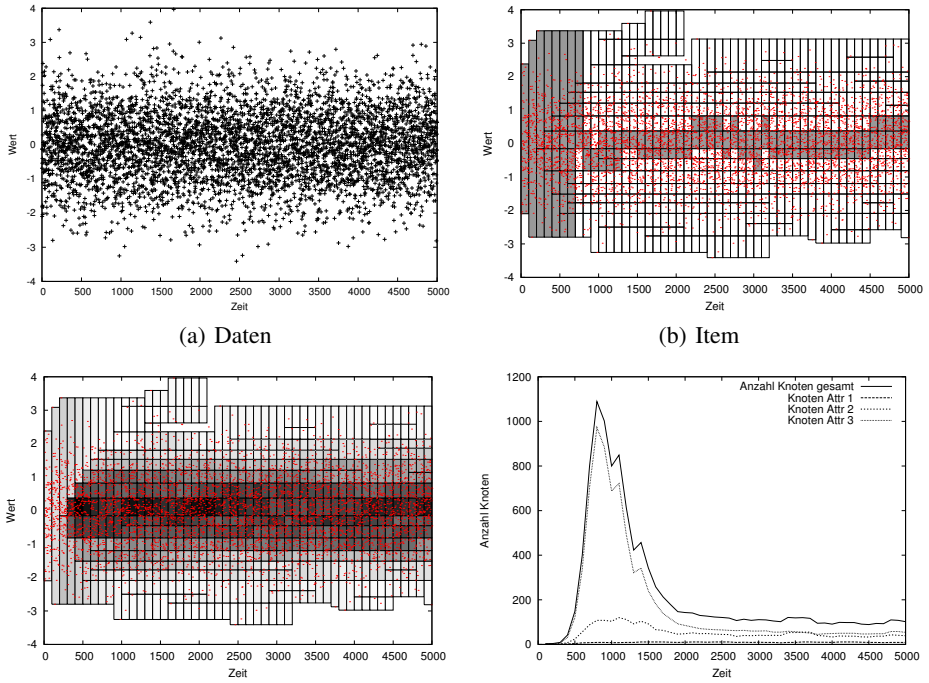
Abbildung 5(b) zeigt sehr gut, dass zu Beginn der Analyse der vollständige Wertebereich durch das Item überdeckt wird. Anschließend fängt der Algorithmus an, den alles überdeckenden Knoten in mehrere Teilknoten zu zerlegen. Aufgrund des Zeitfensters von 5 Batches setzt sich diese Verfeinerung allerdings erst nach 600 Zeiteinheiten durch. Anschließend schwankt das erzeugte Item um den Mittelwert der Standardnormalverteilten Daten.

Elementar für die Menge an Knoten im FP^2-Stream und für die Konstruktion der Items bzw. Itemsets ist das Maß der Informationsdichte. Abbildung 5(c) zeigt für den betrachteten Beispieldatenstrom die Dichte der einzelnen Buckets. Die Bucketgrenzen entsprechen denen aus Abbildung 5(b). Die Abbildung zeigt die mit steigender Knotenverfeinerung zunehmende Informationsdichte.

Die Synopse des FP^2-Stream ist eine Datenstruktur zur effizienten Speicherung von Transaktionen. Das kontinuierliche Verfeinern von Knoten hat allerdings starken Einfluss auf die Größe der Baumstruktur. So führt zum Beispiel ein Splitt in der Wurzel zu einer Verdoppelung aller abhängigen Knoten. Während der initialen Phase muss sich das Verfahren erst an die Daten anpassen. Dies kann eine Vielzahl an Split und Merge Operationen zur Folge haben. In Abbildung 5(d) ist die Zeitliche Entwicklung zu obigen Beispiel dargestellt. Sehr gut zu erkennen ist die Spitze zu Beginn des Verarbeitungsprozesses.

Abbildung 5(d) zeigt außerdem die Anzahl an Knoten pro Attribut im Baum. Erwartungsgemäß steigt diese Zahl mit der Tiefe des Baumes, so dass das Attribut, welches sich auf Blattebene befindet (im Beispiel Attribut 3) durch die höchste Anzahl an Knoten repräsentiert wird.

Durch das Rekombinieren von Items innerhalb des FP^2-Stream werden automatisch auch die entsprechenden Itemsets zusammengeführt. Während der Itemset-Exktraktion werden alle Items aus der Datenstruktur herausgelöst, welche dem minimalen Support genügen. Im Beispiel der standardnormalverteilten Daten sind hierbei zusätzlich 2-Itemsets bzw. vereinzelt auch 3-Itemsets extrahiert wurden. Die Bilder in Abbildung 6 zeigen die zeitliche Entwicklung des 2-Itemsets über den Attributen 0 und 2. Jedes der einzelnen Bilder entspricht dabei dem am Ende eines Batches extrahiertem Itemset. Wiederum sehr gut zu erkennen ist die initiale Phase. Außerdem sehr gut zu erkennen ist das entfernen von Buckets am Rand infolge zu weniger Werte in diesen Regionen. Das entfernen von uninteressanten Knoten ist einer der Gründe, warum sich die Anzahl an Knoten im FP^2-Stream nach dem initialen Anstieg auf ei-



(a) Daten (b) Item

(c) Dichteverteilung (d) Anzahl Knoten

Abbildung 5: Standardnormalverteilte Daten

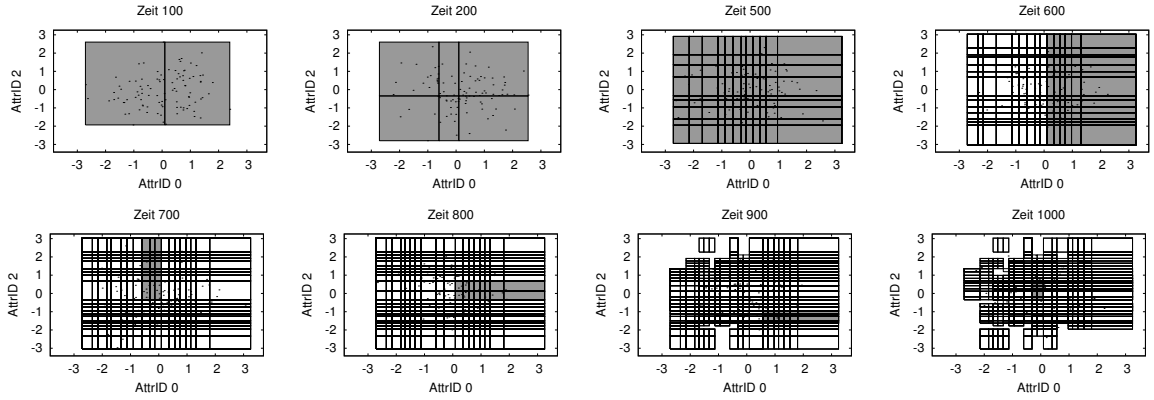


Abbildung 6: Entwicklung des Itemsets s1 und s2

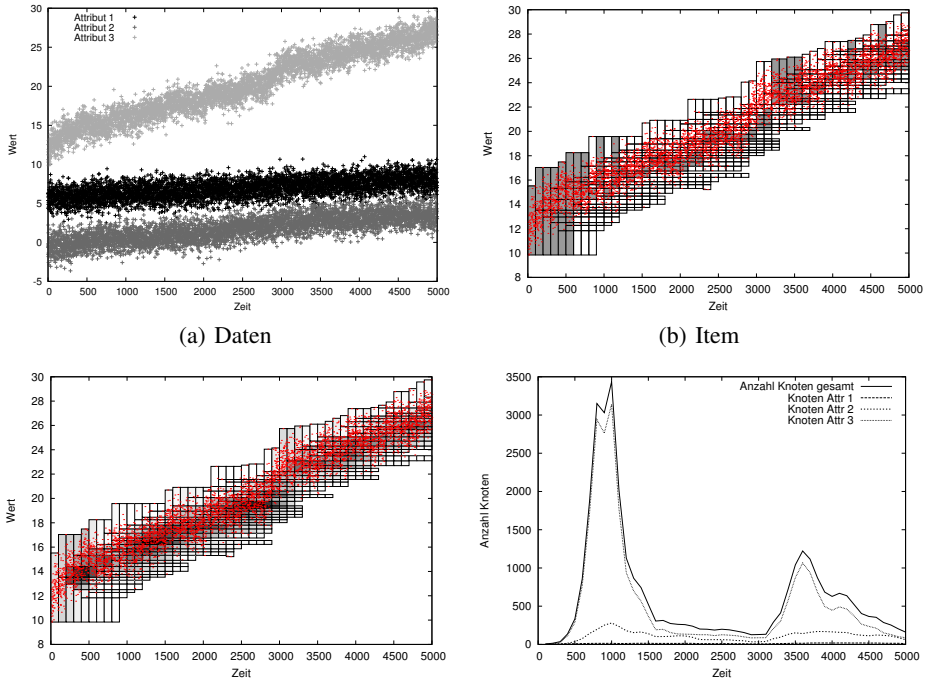
nem deutlich niedrigeren Level einpendelt.

Für den nächsten Test wurden wiederum drei Datenströme erzeugt. Jeder Datenstrom umfasst dabei 5000 Datenpunkte und folgt einem Trend. Zusätzlich wurden die Daten mit normalverteiltem Rauschen überlagert. Abbildung 7(a) zeigt den zeitlichen Verlauf der drei Datensätze.

Das Experiment wurde wiederum mit $|b| = 100$, $N = 5$ und einer maximalen Schiefe $maxskew$ von 0.2 bei 10 Histogrammbuckets durchgeführt. Der geforderte Support betrug 0.15. Abbildung 7(b) zeigt exemplarisch die Entwicklung der Items für ein Attribut. Die notwendige Einschwingphase ist deutlich zu erkennen. Im weiteren Verlauf entwickelt sich das Item mit dem Trend, wobei für den geforderten minimalen Support mehr oder weniger viele Knoten kombiniert werden müssen. Interessant ist die Entwicklung des Items um den Zeitpunkt 1000. Hier sorgen die neu eintreffenden Werte offensichtlich für ein oszillieren zwischen zwei Items, welches sich erst mit dem Teitpunkt 1300 durchsetzen kann.

In Abbildung 7(c) ist die Informationsdichte-Entwicklung des entsprechenden Attributes dargestellt und Abbildung 7(d) zeigt zeitliche Entwicklung der Knotenanzahl im FP^2-Stream. Trotz des kontinuierlichen Trends entwickelt sich die Gesamtknotenanzahl anschließend relativ konstant bei ca. 200 Knoten. Beginnend ab Zeitpunkt 3300 wächst die Knotenanzahl erneut drastisch an. Die Ursache hierfür liegt offensichtlich in einer Änderung der Datencharakteristik während dieser Zeit bei Attribut 3. Trotz dem, dass es sich um trenbehaftet Daten mit normalverteiltem Rauschen handelt, scheint zwischen den Zeitpunkten 3000 und 3500 eine weitere Überlagerung (ähnlich einem Burst) aufzutreten. Diese plötzliche Veränderung hat entsprechend Auswirkungen auf die Knoten und deren Dichten (siehe Abbildung 7(c)), welche Attribut 3 repräsentieren und führen vermutlich zu dem Peak.

Diese ersten Ergebnisse zeigen, dass das Verfahren des quantitativen FP-Mining für die Analyse von Datenströmen prinzipiell geeignet ist. Es wurde gezeigt, dass das eingeführte Maß der Informationsdichte ist für Erzeugen in-



(a) Daten (b) Item

(c) Dichteverteilung (d) Anzahl Knoten

Abbildung 7: Trendbehaftete Daten

teressanter Itemsets. Erst dieses ermöglicht die Analyse über Datenströmen, da es sowohl für das Teilen und Verbinden von Intervallen, als auch für die Rekombination als Schritt der Itemsetextraktion herangezogen wird. Mit dem präsentierten Maß kann auch das Testen aller möglichen Intervallkombinationen verzichtet werden, was einer der wesentlichen Gründe für die Eignung zur Analyse von Datenströmen ist.

6 Zusammenfassung

Das FP-Mining über statischen Daten mit kategorischen Werte zählt heute zu den klassischen Data Mining Verfahren, für welches eine Vielzahl an Lösungen in der Literatur zu finden sind. In der hier vorgestellten Arbeit wurde ein neuer Ansatz für die Identifikation von häufigen Mustern über quantitative Attribute aus Datenströmen präsentiert. Das präsentierte Verfahren kombiniert hierzu bekannte Techniken aus dem Data Mining Bereich mit Verfahren der mehrdimensionalen Indexstrukturen. Neben den notwendigen Verarbeitungsschritten wurde die prinzipielle Funktion anhand von Beispielen gezeigt. In weiteren Arbeiten soll die Funktion des Verfahrens mit realen Szenarien untersucht und evaluiert werden. Das DSMS AnduIN erlaubt die Teilweise Auslagerung von Funktionalität in Wireless Sensor Netzwerke. Eine der zukünftigen Arbeiten ist daher die Migration des vorgestellten Verfahrens hin zu einem In-Network-Operator von AnduIN mit dem Ziel einer möglichst effizienten Verarbeitung.

Literatur

[1] R. Agrawal, T. Imielinski, and A. N. Swami. Mining Association Rules between Sets of Items in Large Databases. In *'93, Washington, D.C., USA*, pages 207–216, 1993.

[2] Y. Aumann and Y. Lindell. A statistical theory for quantitative association rules. In *Journal of Intelligent Information Systems*, pages 261–270, 1999.

[3] T. Fukuda, Y. Morimoto, Sh. Morishita, and T. Tokuyama. Mining optimized association rules for numeric attributes. In *PODS '96*, pages 182–191. ACM, 1996.

[4] M. Halatchev and Le Gruenwald. Estimating missing values in related sensor data streams. In *COMAD*, pages 83–94, 2005.

[5] J. Han, J. Pei, and Y. Yin. Mining Frequent Patterns without Candidate Generation. In *2000, Dallas, USA*, pages 1–12, 2000.

[6] D. Klan, K. Hose, M. Karnstedt, and K. Sattler. Power-aware data analysis in sensor networks. In *ICDE '10*, Long Beach, California, USA, 2010. IEEE, IEEE.

[7] C. K.-S. Leung and Q. I. Khan. Dstree: A tree structure for the mining of frequent sets from data streams. In *ICDM '06*, pages 928–932, Washington, DC, USA, 2006. IEEE Computer Society.

[8] G. S. Manku and R. Motwani. Approximate Frequency Counts over Data Streams. In *2002, Hong Kong, China*, pages 346–357, 2002.

[9] J. S. Park, M.-S. Chen, and P. S. Yu. An effective hash-based algorithm for mining association rules. In *SIGMOD '95*, pages 175–186, New York, NY, USA, 1995. ACM.

[10] G. Piatetsky-Shapiro. *Discovery, analysis and presentation of strong rules*. 1991.

[11] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. *SIGMOD Rec.*, 25(2):1–12, 1996.

[12] S. K. Tanbeer, Ch. F. Ahmed, B.-S. Jeong, and Y.-K. Lee. Efficient frequent pattern mining over data streams. In *CIKM '08*, pages 1447–1448, New York, NY, USA, 2008. ACM.

[13] A. Tuzhilin. On subjective measures of interestingness in knowledge discovery. In *KDD '95*, pages 275–281. AAAI Press, 1995.

[14] K. Wang, L. Tang, J. Han, and J. Liu. Top down fp-growth for association rule mining. In *PAKDD '02*, pages 334–340, London, UK, 2002. Springer-Verlag.