Frequent Pattern Mining in Data Streams

Raymond Martin

Agenda

-Breakdown & Review
-Importance & Examples
-Current Challenges
-Modern Algorithms
-Stream-Mining Algorithm
-How KPS Works
-Combing KPS and Apriori
-Future Work
-References

Breakdown & Review

Break Down

Frequent Pattern Mining in Data Streams

We want to turn our data into transaction data sets

We want to find frequently occurring patterns such as itemsets, subsequences, subtrees, and subgraphs

Potentially infinite data set

Not all data is available when we want it

Data must be processed on demand

Frequent Pattern Mining

-Basic data mining concept
-Itemset = a tuple which represents one entity
-Frequent Itemsets = tuples which occur more so than others
-Popular Algorithms for static data:
    -FP-tree
    -Apriori
    -Tree Projection
    -Eclat

The Digital Data Firehouse

“

90% of all the data in the world has been generated over the last two years

ScienceDaily & SINTEF 2013

Data Streams

-High velocity data

-Transactions constantly coming in

-General Practices:

Landmark Windowing - W[i, t]

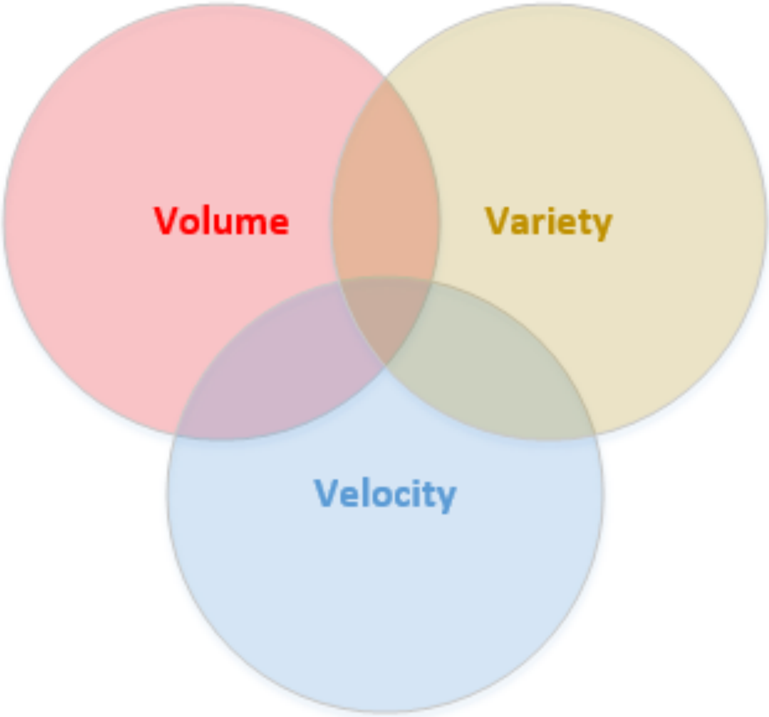
Sliding Windowing - W[t-w, t]

Damped Windowing - W[0,t]*Decay

Tilted Time Windowing - (like sampling)

Sampling - W[?]

Desired Properties



Volume ex. Walmart Scan Data
Velocity ex. Stock Market
Variety ex. Twitter/Facebook Data

The Marriage

-The goal is to find underlying structures and patterns over time.

-Use only one pass over new transactions.

-Must be quick and low on extraneous memory usage

Importance & Examples

Credit Card Fraud

Network Usage & Anomalies

Streaming Services

General Challenges

Memory Efficiency

-There exists exponential patterns and pruning takes time

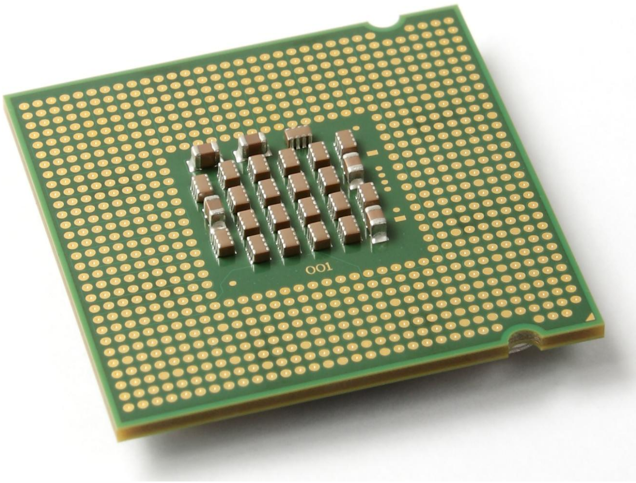
-We want to decrease our out-of-core memory footprint as much as possible

-Memory should be reserved for the current transaction data


Computationally Possible

-We want to have enough time to include each transaction before the next arrives

-Down-closure property for pruning is too expensive



Quality Approximations

-Reduce memory or runtime for better results?

-Where is this equilibrium?

-Do we have to settle for less than perfect?

Modern Algorithms

Algorithms

-Lossy Counting - Manku & Motwani, 2002

-FPDM - Yu et al., 2004

-Moment - Chi et al., 2004

-estDec - Change and Lee, 2003

Lossy Counting

-False Positive Oriented (does not allow for false negatives)

-False Positives have deterministic bounds

-Goes over the entire dataset

-First one-pass algorithm

FPDM

-False Negative Oriented

-Quality assured by Chernoff bound

-Partitions data into equal segments

-Goes over the entire dataset

Moment

-Uses the sliding window approach to data streams

-Wants small sized windows to keep data in main memory

-Related to incremental association rule mining

-Uses a CET Tree structure

-Must have stable itemsets between transactions

estDec

-Uses Lattice data structure

-Depends on the Damped Window model

Specific Challenges

Large Windows

-These slow down when windows/segments cannot be held in main memory

-Out-of-Core data structure take up extraneous memory

Continuity of Data

-The goal is to add to our already existing set of data and itemsets generated

Solution: ‘Stream-Mining’ Algorithm

‘Stream-Mining’

-A false positive approach

-Finds frequent itemsets over the entire stream

-Almost no extraneous memory used

-Does not rely on windowing

-Uses a clever algorithm called KPS

KPS

KPS

K.P.S. - Karp, Papadimitriou, and Shenker

-Designed for finding frequent items with 2 passes of large data sets and minimal out of core memory

-Only input is the threshold desired

-Can be adapted into an approximate one pass

KPS

-Aids in finding 1-itemset (set of all frequently occurring individual elements)

-We want to find elements which occur more often than $N\theta$

N = number of elements

θ = user set threshold ($0 < \theta < 1$)

Extraneous Memory - $O(1/\theta)$

Comparison

-Trivial algos, such as Apriori use O(n) memory where n is the number of elements

-KPS requires $O(1/\theta)$ space

(this is independent from the number of elements in the dataset!)

KPS Generalization

A = [1, 1, 0, 2, 1, 5, 4, 1, 1, 1]

N = 10

θ = .5 (because we want any elements that occurs 50% or more of the time)

Memory = 1/θ = 1/.5 = 2 items

Time = 2n = 2 passes = 20 items

KPS in other cases

-For i-itemsets KPS = $\Omega((1/\theta)*P(L, i))$

-L = length of transaction

-Therefore, as i grows larger, the memory complexity increases (we want to keep i below 2)

Applying KPS

-How do we deal with transaction sequences instead of just items?

-What if we want to find all of the K-Itemsets, and not just the most frequent ones?

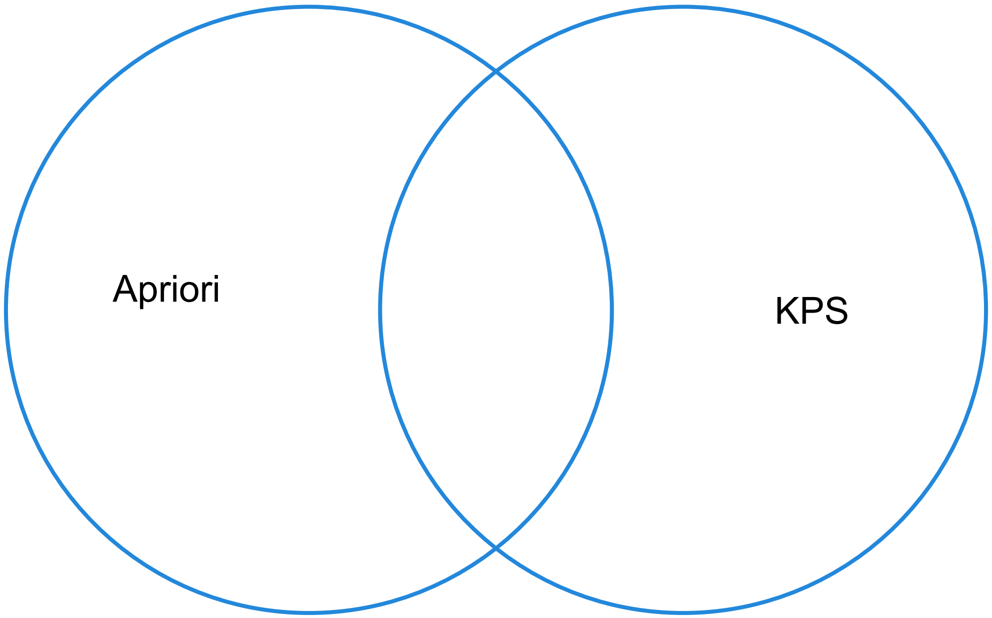
-How do we provide an accuracy bound with only one pass?

KPS + Apriori

A Hybrid Approach

-KPS may be used of i-itemsets where i <= 2

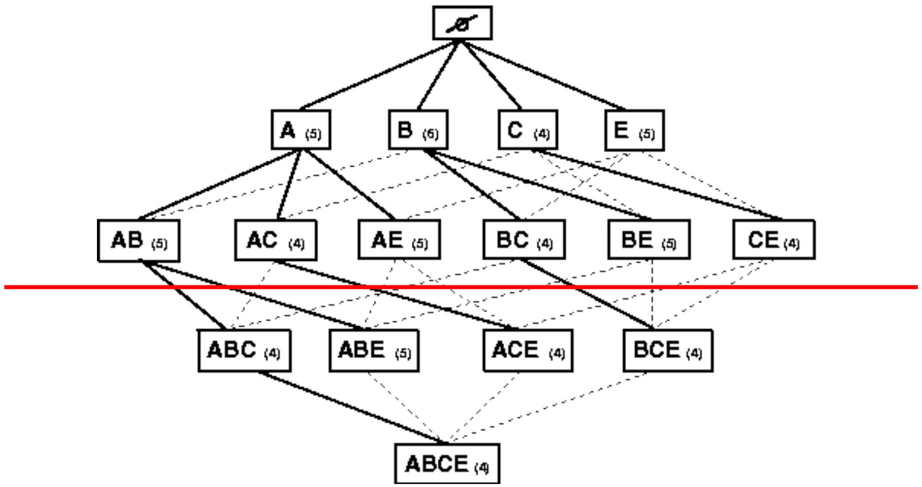
-Apriori is used for i-itemsets where i > 2



I thought Apriori was too slow?

-Apriori uses previous knowledge from i-itemsets -1 to generate the i-itemsets.

-Because 1-itemsets and 2-itemsets have already been generated, Apriori can rely on those figure to determine the K-itemset.



Results

Memory

-Lossy Counting with a 1% support level requires 44MB + disk reads for 1 million elements

-4-20 Million Transactions = 2.5MB of memory to summarize

-This enables mining on mobile or sensor devices with little memory

Relative Performance

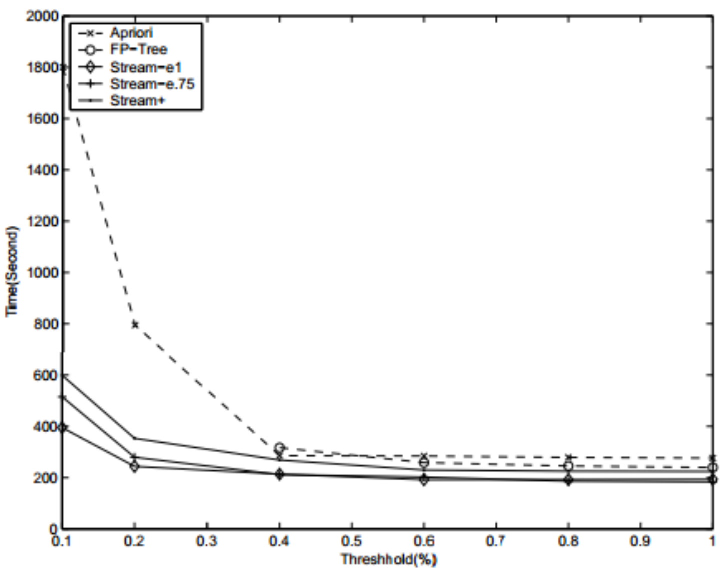


Figure 3. Execution Time with Changing Support Level (T10.I4.N10K Dataset)

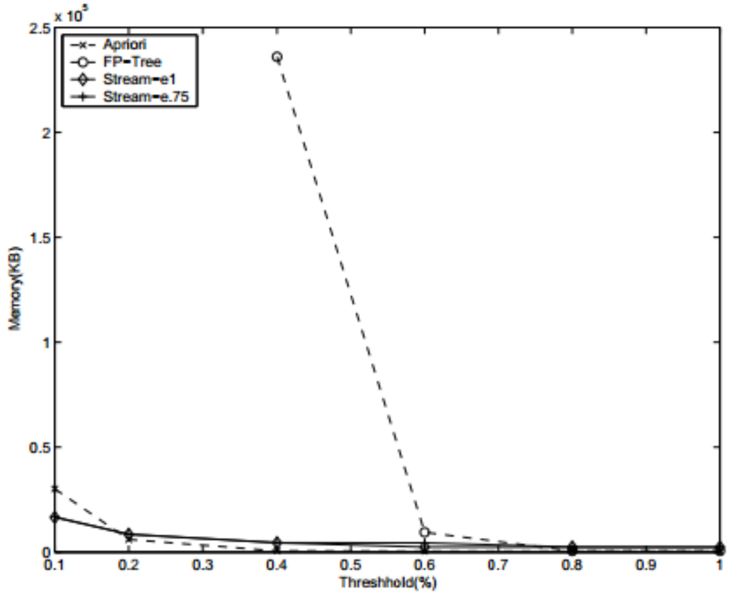


Figure 4. Memory Requirements with Changing Support Level (T10.I4.N10K Dataset)

Future Work & Related Problems

Scalable Algorithms

-SAMOA – Scalable Advanced Massive Online Analysis

-Similar to the function MapReduce provides

-Many of these have false positive issues which cannot be deterministically bound

Distributed Data Streams

-Ex: Weather

-Being able to handle multiple sources of data and applying weight accordingly

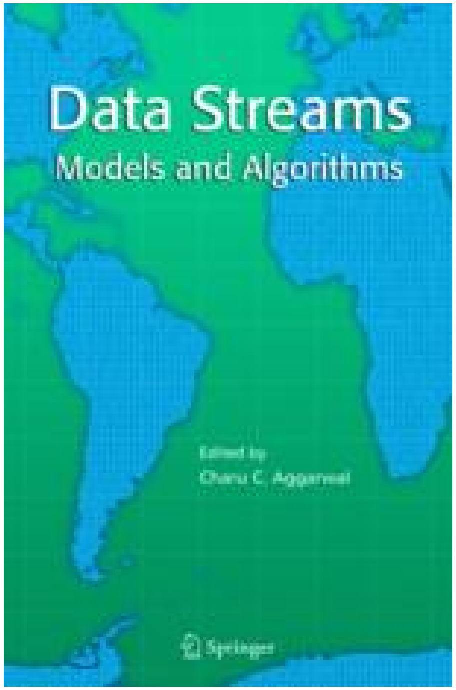
Frequent Temporal Patterns

-Using the sliding window technique

-Uses time-series data to detect changing patterns as the data is received

-Trend Identification and change detection

Stream-Mining Reference



Data Streams: Models and Algorithms
Chapter 4: Frequent Pattern Mining in Data Streams

By Charu C. Aggarwal

References

SINTEF. "Big Data, for better or worse: 90% of world's data generated over last two years." ScienceDaily. ScienceDaily, 22 May 2013. <www.sciencedaily.com/releases/2013/05/130522085217.htm>.

Jin, Rouming, and Gagan Argawal. "Frequent Pattern Mining In Data Streams." *Data Streams*. Boston/Dordrecht/London: Kluwer Academic, 2007. 61-81. Print.

"Frequent Pattern Mining." *Data Mining Articles RSS*. Data Mining Articles, 1 Jan. 2013. Web. 25 Feb. 2015. <http://www.dataminingarticles.com/data-mining-introduction/frequent-pattern-mining/>.

Hahsler, Micheal, John Forest, and Mathew Bolanos. "Introduction to Stream: An Extensible Framework for Data Stream Clustering Research with R." Web. 25 Feb. 2015. <http://cran.r-project.org/web/packages/stream/vignettes/stream.pdf>.

Agrawal R, Imielinski T, Swami A (1993). “Mining Association Rules between Sets of Items in Large Databases.” In Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 207–216. Washington D.C.

Vijayarani S, Sathya P (2012). “A Survey on Frequent Pattern Mining Over Data Streams.” International Journal of Computer Science and Information Technology & Security, 2(5), 1046–1050. ISSN 2249-9555.

Thanks For Listening

Questions?