Another Algorithm to Find ALL Frequent Itemsets in 2 Passes (most of the time)

Toivonen's Algorithm

Slides adapted from
Prof. Jeffrey D. Ullman, Stanford University &
Dr. Alexander (Sasha) Apartsin, Tel-Aviv University

Recall: The Simple Algorithm

- Take a random sample of the market baskets.
- Run a-priori or one of its improvements (for sets of all sizes, not just pairs) in main memory, so you don't pay for disk I/O each time you increase the size of itemsets.
- Use as your support threshold a suitable, scaled-back number.
  - Example: if your sample is 1/100 of the baskets, use $s/100$ as your support threshold instead of s.

Simple Algorithm – Option

- Optionally, verify that your guesses are truly frequent in the entire data set by a second pass.
- But you don't catch sets frequent in the whole but not in the sample.
  - Smaller threshold, e.g., $s/125$ instead of $s/100$, helps catch more truly frequent itemsets.
    - But requires more space.
- This algorithm belongs to the "Monte Carlo" type of Randomized Algorithms, i.e.
  - It always gives you an answer BUT, with small probability, the answer can be WRONG

Toivonen's Algorithm

- One pass over a small sample and one full pass.
- Will NOT produce False Positive or False Negative
- Small Probability of Not able to give you any Conclusive Answer
  - In this case, RERUN the algorithm with another different small sample until an Answer is reached
  - Expected number of Runs to reach a conclusion is a small constant
- This type of algorithm belongs to the "Las Vegas" type of Randomized Algorithms, i.e.,
  - If it gives you any answer, the answer is always 100% correct. However, with small probability, it may NEVER give you any answer.

Toivonen's Algorithm

STEP 1:

- Start as in the Simple Algorithm to construct candidate frequent itemsets from a Small Sample, BUT lower the threshold slightly for the sample.
  - Example: if the sample is 1% of the baskets, use $s/125$ as the support threshold rather than $s/100$.
  - Goal is to avoid missing any itemset that is frequent in the full set of baskets.

Toivonen's Algorithm

Define Immediate Subset of Itemset:

- Form a Subset by deleting a single item from the Itemset

STEP 2:

Construct the Negative Border which consists of:

- Itemsets, each of which is:

  NOT frequent in the Sample

  BUT

  ALL of its Immediate Subsets are frequent in the Sample

***By definition, the empty set {} is always frequent*

=> Any non-frequent singleton set in the sample will always be included in the Negative border

Example of Negative Border

- Items $\{A, B, C, D, E\}$
- Frequent itemsets $\{A\}, \{B\}, \{C\}, \{D\}, \{B, C\}, \{C, D\}$
  IN THE SAMPLE
- Negative border
  - {E},{A,B},{A,C},{A,D},{B,D}

Toivonen's Algorithm

STEP 3: Go through the FULL set of data to count the occurrences of

- (1) The frequent itemsets identified in the Sample

AND

- (2) The itemsets which belong to the Negative Border in the Sample

Toivonen's Algorithm

Outcomes:

- If No member of Negative Border is frequent in the whole dataset THEN
  - Frequent Itemsets in the Sample are the only possible candidates for Frequent Itemsets in the whole dataset
    - Note that the count of each of these candidates are already collected and verified during STEP 3 to determine if they are truly frequent itemsets in the whole
- If ANY member of the Negative Border is frequent in the whole THEN
  - No Conclusion and we have to repeat the process with another Sample until a Conclusion is reached

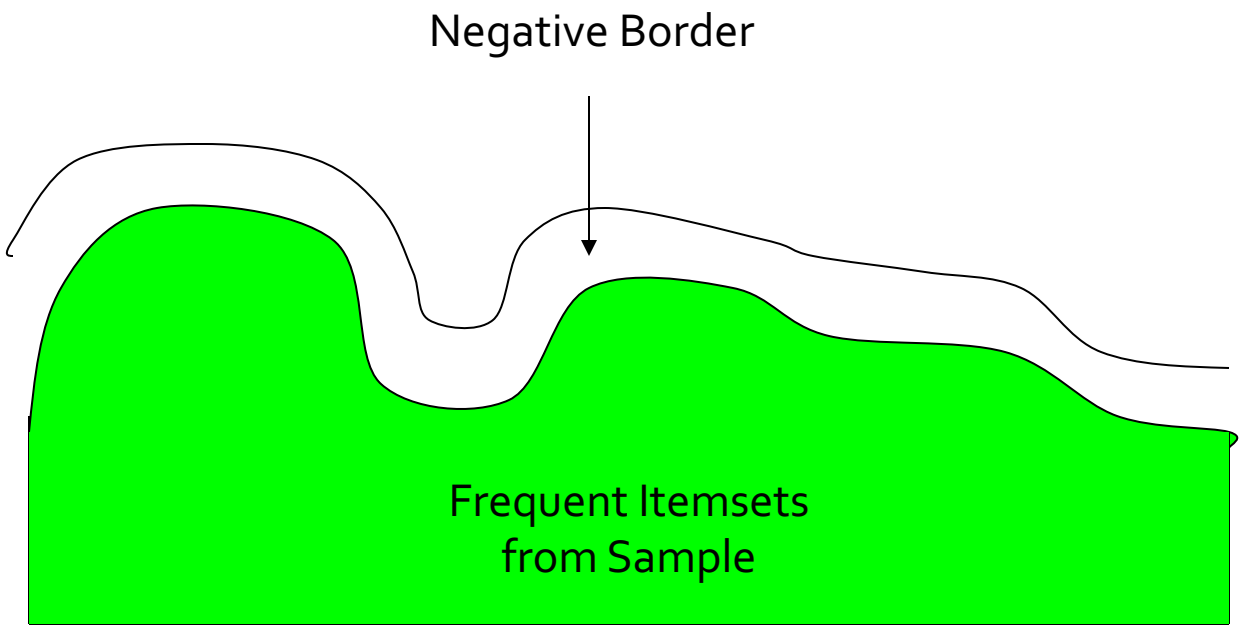
Picture of Negative Border

...

tripletons

doubletons

singletons



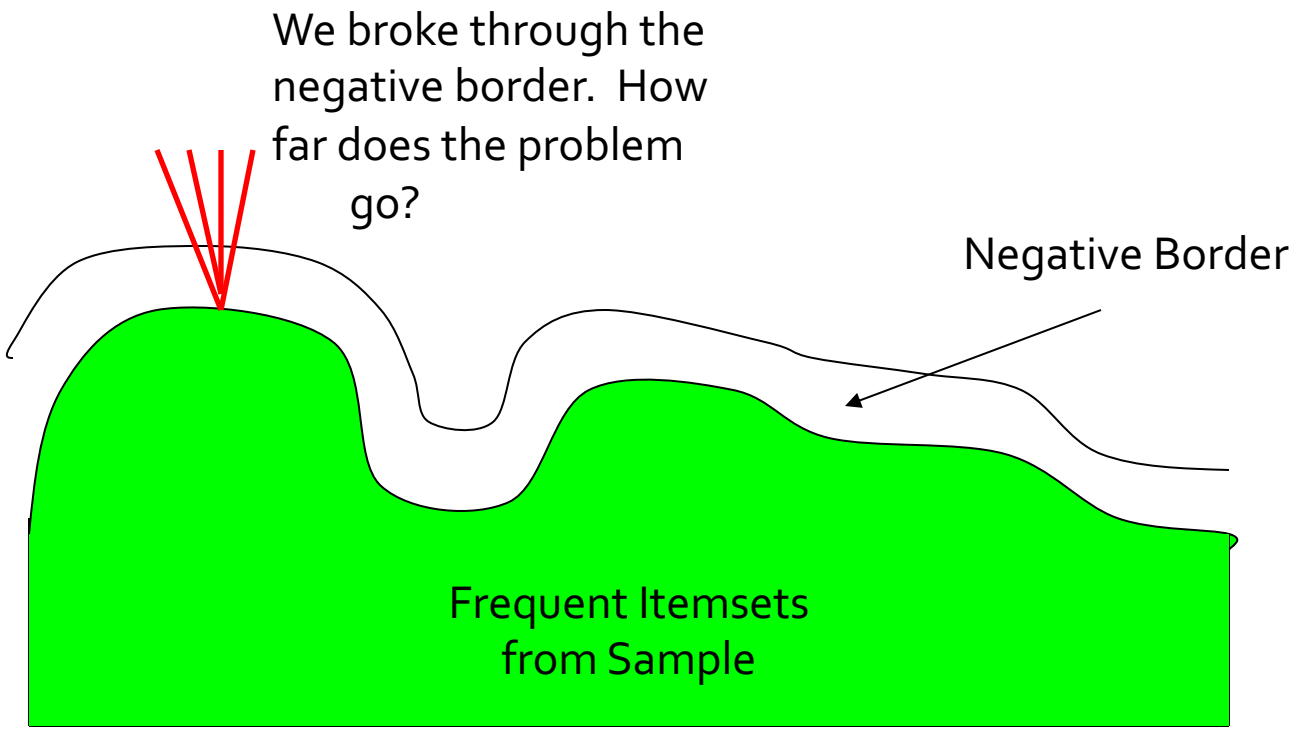
If Something in the Negative Border Is Frequent . . .

...

tripletons

doubletons

singletons



Analysis (1)

- Toivonen's Algorithm Never produce False Positive because
  - Candidate Frequent Itemsets are ALWAYS checked against the Whole dataset
- Assume a False Negative is produced

=> There is at least one False Negative, that is:

- Frequent in the Whole

AND

- NOT Frequent in the Sample

AND

- NOT in Negative Border (o.w. the algorithm would have said that there is no conclusive answer).

Analysis (2)

- Let itemset *S* be frequent in the whole BUT NOT in Negative Border and NOT frequent in the Sample
- Let *T* be a smallest subset of *S* that is not frequent in the sample. (N.B. *T* can be = *S* itself)
- *T* is frequent in the whole (because *S* is frequent & monotonicity property).
- *T* is in the negative border because
  - *T* is not frequent in the sample and
  - Each of T's immediate subsets are frequent in the sample (due to our definition that *T* is the "smallest…")
- T is therefore both frequent in the whole and in the Negative Border
- Toivonen's algorithm should not have produced a conclusion => Contradiction => S cannot exist !

One more Observation

- If you execute a-priori on the sample, in main memory, then every candidate k-set is either a frequent k-set or it is in the negative border.
- And the entire negative border can be found this way.
- So you get the negative border “for free.”