MarketShare

Operations on Cross-sections(also X-sections, tuples)

Murali Mohan

02/26/2014

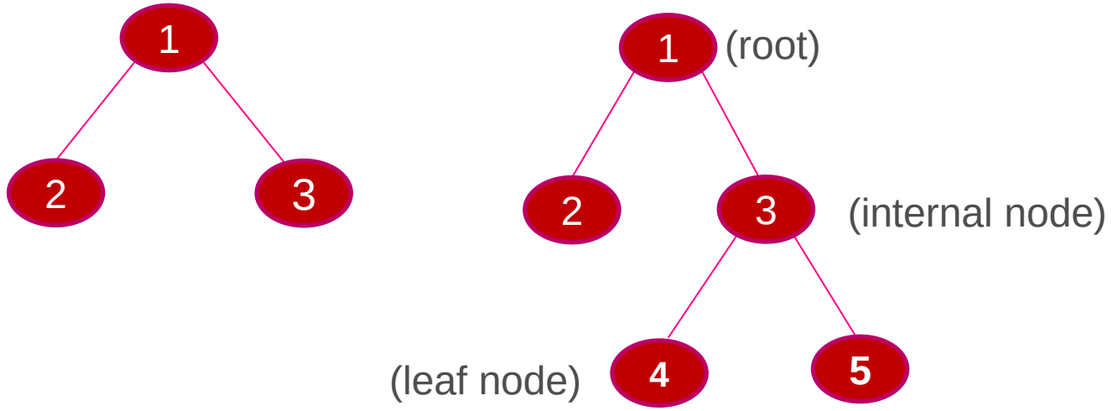
Cross-section

- A k-tuple obtained by the Cartesian Product of k dimensions
- Represents spend value (measure)
- Is specific and unique to each client
- Cardinality |C| of the set C of x-sections is huge

  ($|C| = |D_1| \times |D_2| \times \ldots \times |D_k|$, *where* $|D_i|$ *is the cardinality of the* i^{th} *dimension*)

Cross-sections from Dimension Hierarchies

An example Set C of x-sections obtained from 2 dimension hierarchies



C = { (2,2), (2,4), (2,5), (3,2), (3,4), (3,5), (2,3), (3,3),

(2,1), (3,1), (1,2), (1,3), (1,4), (1,5), (1,1) }

|C| = 3x5 = 15

Relations among measures ($M_{i1..ik}$) of X-sections

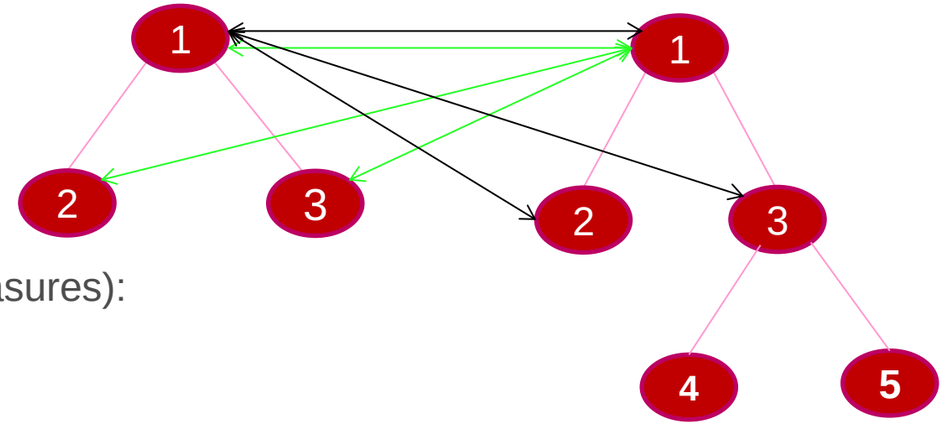
An instance of x-sections obtained from 2 dimension hierarchies

For measures aggregated bottom-up(Aggregated Measures):

$M_{1,1} = M_{1,2} + M_{1,3}$

(or)

$M_{1,1} = M_{2,1} + M_{2,3}$



For measures distributed top-down(Distributed Measures):

$M_{1,2} = w_{1,2} * M_{1,1}$

$M_{1,3} = w_{1,3} * M_{1,1,}$ where $0 <= w_{1,2}, w_{1,3} <= 1$ and $w_{1,2} + w_{1,3} = 1$

$M_{2,1} = w_{2,1} * M_{1,1}$

$M_{3,1} = w_{3,1} * M_{1,1,}$ where $0 <= w_{2,1}, w_{3,1} <= 1$ and $w_{2,1} + w_{3,1} = 1$

Computing Measures

Problem: Given k dimension hierarchies $D_1,D_2,\ldots,D_k$, and measures for a set L of cross-sections obtained by the Cartesian product of sets L_1, $L_2,..,L_k$ of leaf nodes of each dimension hierarchy, compute measures for L', the complement of L in C by aggregation; C being the set of all cross-sections, also called an *n*-dimensional cube. Given measure for the singleton set R containing the cross-section formed by the Cartesian product of the roots of each dimension and distribution functions $f_{i,j}$ for each j^{th} internal node of i^{th} dimension, compute measures for R', the complement of R in C by distribution.

Approach: An iterative method to compute the measures by logically ordering the cross-sections has been discussed. Using the definitions of heights and depths for cross-sections, defined in terms of heights and depths of the dimension hierarchies, the set C is partitioned into linearly ordered disjoint subsets. The cube building procedure works by iteratively computing measures for one subset at a time, deriving values from the measures computed in the previous steps. Computing measures by aggregation and distribution differ in terms of the aggregate and distribute functions used and the overall approach in that the former follows a bottom-up approach and the latter a top-down approach.

Logical Ordering of the Cross-sections Set

- Two ways to order
- In terms of heights of nodes of dimension hierarchies
  - Used to build cubes by aggregation
- In terms of depths of nodes of dimension hierarchies
  - Used to build cubes by distribution

Ordering the Cross-sections set by height

- Using the definition of heights for a dimension hierarchy
- Leaf in a dimension hierarchy has height 0. A non-leaf node has a height of max{children's height} + 1
- Define height *h* of a x-section as the sum of heights of elements(dimensions) in the tuple *($h = h_1 + h_2 + \ldots + h_k$, h_i is the height of the i^{th} dimension in the tuple)*
- Partition C into linearly ordered disjoint subsets, each associated with a unique *h*

Aggregating measures bottom-up using X-section height *h*

- Input: Measures for tuples of height *h* = 0 (tuples from the set L)
- Output: Measures for tuples of height 1 <=*h* <= *h_max* (tuples from the set L')
  - (*h_max* = summation{heights of roots all dimensions})
- Algorithm: Start at h = 1
- For all tuples with height h, do:
  - Let the tuple T be (d_{i1}, d_{i2}, …, d_{ik})
  - Choose a dimension d_{ij} in the tuple that has children (choose by scanning either from the left or the right of the tuple)
  - $M_{i1, i2,\ldots,ik} = \sum_{c_{ij}} (d_{i1}, d_{i2}, \ldots c_{ij}, d_{ik})$, $c_{ij} \in$ {children of d_{ij}}
- Increment h and repeat the above step until *h* > *h_max*

  (Partial cube can be built by directing the algorithm to halt at a target height h_t)
- Proof of correctness and time complexity(to be added later)

Aggregation instance for the example X-section set

h = 3

Measure	21
X-section	(1,1)

(measure at the top reconciles with the sum of spends at the bottom)

Alternative roll-up

h = 2

Measure	6	15	16
X-section	(2,1)	(3,1)	(1,3)

h = 1

Measure	5	11	5	7	9
X-section	(2,3)	(3,3)	(1,2)	(1,4)	(1,5)

h = 0

Measure	1	2	3	4	5	6
X-section	(2,2)	(2,4)	(2,5)	(3,2)	(3,4)	(3,5)

Ordering the Cross-sections set by depths

- Using the definition of depths for a dimension hierarchy
- Root of a dimension hierarchy has depth 0. Depth of a non-root node = depth of it's parent + 1
- Define depth *d* of a x-section tuple as the sum of depths of elements(dimensions) in the tuple *($d = d_1 + d_2 + \ldots + d_k$, d_i is the depth of the i^{th} dimension in the tuple)*
- Partition C into linearly ordered disjoint subsets, each associated with a unique *d*

Distributing measures top-down using X-section depth *d*

- Input: Measures for tuples of depth *d* = 0 (tuples from the set R)

  and distribution functions f_{ij} for each of j^{th} internal node of i^{th} dimension hierarchy

- Output: Measures for tuples of height 1 <=*d* <= *d_max*
  - (*d_max* = summation{maximum depths of all dimension hierarchies})
- Method: Start at d = 1
- For all tuples with depth d, do:
  - Let the tuple T be (d_{i1}, d_{i2}, …, d_{ik})
  - For all tuples ∈ { (d_{i1}, d_{i2}, …c_{ij}, d_{ik}) }, where c_{ij} ∈ {children of d_{ij}}, set measure = $f_{ij}(M_{i1, i2,…,ik})$
- Increment d and repeat the above step until *d* > *h_max*

  (Partial cube can be built by directing the algorithm to halt at a target depth d_t)
- Proof of correctness and time complexity(to be added later)

Distribution instance for the example X-section set

d = 0

Measure	64
X-section	(1,1)

(measure at the top reconciles with the sum of spends at the children, highlighted in red)

d = 1

Measure	16	48	32	32
X-section	(1,2)	(1,3)	(2,1)	(3,1)

d = 2

Measure	24	24	8	8	12	36
X-section	(2,3)	(3,3)	(2,2)	(3,2)	(1,4)	(1,5)

d = 3

Measure	6	18	6	18
X-section	(2,4)	(2,5)	(3,4)	(3,5)

Updates to Cross-sections

- Roll-up and drill-down deltas
- Define virtual graphs with two types of edges: Contributes-To and Obtained-from
- Use Contributes-To edges to roll-up/aggregate deltas and Obtained-from edges to drill-down/distribute deltas

Identifying roll-up/drill down paths for updates

- TBD

Registering incremental updates and periodic commits to persistent store

- Register incremental changes made by the user without actually updating the cross-section
- Apply on-demand transforms to the target x-section (to be viewed by the user)
- Periodically consolidate the incremental changes and update the x-section Set as a background transactional process

(Details TBD)

***