Data Mining with Cubegrades

Amin A. Abdulghani
Quantiva, USA

INTRODUCTION

Much interest has been expressed in database mining by using association rules (Agrawal, Imielinski, & Swami, 1993). In this article, I provide a different view of the association rules, which are referred to as *cubegrades* (Imielinski, Khachiyan, & Abdulghani, 2002) .

An example of a typical association rule states that, say, in 23% of supermarket transactions (so-called market basket data) customers who buy bread and butter also buy cereal (that percentage is called *confidence*) and that in 10% of all transactions, customers buy bread and butter (this percentage is called *support*). Bread and butter represent the body of the rule, and cereal constitutes the consequent of the rule. This statement is typically represented as a probabilistic rule. But association rules can also be viewed as statements about how the cell representing the body of the rule is affected by specializing it with the addition of an extra constraint expressed by the rule's consequent. Indeed, the confidence of an association rule can be viewed as the ratio of the support drop, when the cell corresponding to the body of a rule (in this case, the cell of transactions including bread and butter) is augmented with its consequent (in this case, cereal). This interpretation gives association rules a dynamic flavor reflected in a hypothetical change of support affected by specializing the body cell to a cell whose description is a union of body and consequent descriptors. For example, the earlier association rule can be interpreted as saying that the count of transactions including bread and butter drops to 23% of the original when restricted (rolled down) to the transactions including bread, butter, and cereal. In other words, this rule states how the count of transactions supporting buyers of bread and butter is affected by buying cereal as well.

With such interpretation in mind, a much more general view of association rules can be taken, when support (count) can be replaced by an arbitrary measure or aggregate, and the specialization operation can be substituted with a different "delta" operation. Cubegrades capture this generalization. Conceptually, this is very similar to the notion of gradients used in calculus. By definition, the *gradient* of a function between the domain points x_1 and x_2 measures the ratio of the *delta change* in the function value over the delta change between the points. For a given point x and function f(), it can be interpreted as a statement of how a change in the value of x (Δx) affects a change in value in the function ($\Delta f(x)$).

From another viewpoint, cubegrades can also be considered as defining a primitive for cubes. An *n*-dimensional cube is a group of *k*-dimensional (k<=n) cuboids arranged by the dimensions of the data. A cell represents an association of a measure m (e.g., total sales) with a member of every dimension. The scope of interest in Online Analytical Processing (OLAP) is to evaluate one or more measure values of the cells in the cube. Cubegrades allow a broader, more dynamic view. In addition to evaluating the measure values in a cell, they evaluate how the measure values change or are affected in response to a change in the dimensions of a cell. Traditionally, OLAP has had operators such as drill downs, rollups defined, but the cubegrade operator differs from them as it returns a value measuring the effect of the operation. Additional operators have been proposed to evaluate/measure cell *interestingness* (Sarawagi, 2000; Sarawagi, Agrawal, & Megiddo, 1998). For example, Sarawagi et al. computes anticipated value for a cell by using the neighborhood values, and a cell is considered an exception if its value is significantly different from its anticipated value. The difference is that cubegrades perform a direct cell-to-cell comparison.

BACKGROUND

An association or propositional rule can be defined in terms of cube cells. It can be defined as a quadruple *(body, consequent, support, confidence)* where *body* and *consequent* are cells over disjoint sets of attributes, *support* is the number of records satisfying the *body,* and *confidence* is the ratio of the number of records that satisfy the *body* and the *consequent* to the number of records that satisfy just the body. You can also consider an association rule as a statement about a *relative change* of measure, COUNT, when specializing or drilling down the cell denoted by the *body* to the cell denoted by the *body + consequent*. The *confidence* of the rule measures how the *consequent* affects the support when drilling down the *body*. These association rules can be generalized in two ways:

- By allowing relative changes in other measures, instead of just confidence, to be returned as part of the rule.
- By allowing cell modifications to be able to occur in different directions' instead of just specializations (or drill-downs).

These generalized cell modifications are denoted as *cubegrades*. A cubegrade expresses how a change in the structure of a given cell affects a set of predefined measures. The original cell being modified is referred to as the *source,* and the modified cell as *target*.

More formally, a cubegrade is a 5-tuple *(source, target, measures, value, delta-value)* where

- *source* and *target* are cube cells
- *measures* is the set of measures that are evaluated both in the *source* as well as in the *target*
- *value* is a function, value: *measures* $\rightarrow R$, that evaluates measure $m \in$ *measures* in the *source*
- *delta-value* is also a function, *delta-value: measures* $\rightarrow R$, that computes the ratio of the value of $m \in$ *measures* in the *target* versus the *source*

A cubegrade can visually be represented as a rule form:

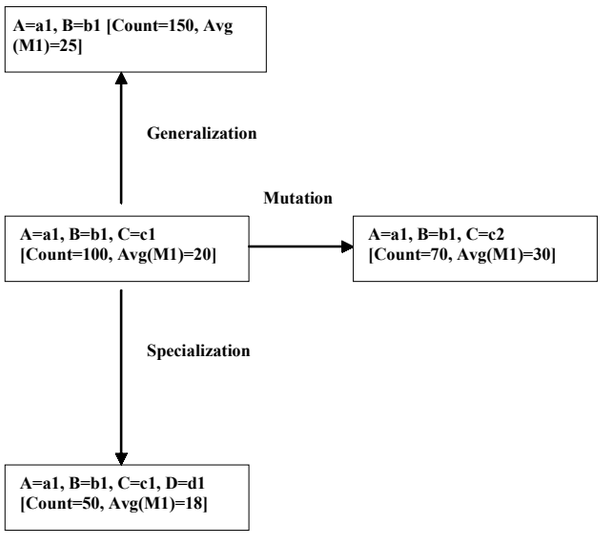
Source $\rightarrow$ target, [measures, value, delta-value]

Define a *descriptor* to be an attribute value pair of the form *dimension=value* if the dimension is a discrete attribute, *or dimension = [lo, hi]* if the attribute is a dimension attribute. The cubegrades are distinguished as three types:

- **Specializations:** A cubegrade is a *specialization* if the set of descriptors of the target are a superset of those in the source. Within the context of OLAP, the target cell is termed a *drill-down* of *source.*
- **Generalizations:** A cubegrade is a *generalization* if the set of descriptors of the target cell are a subset of those in the source. Here, in OLAP, the target cell is termed a *roll-up* of *source.*
- **Mutations:** A cubegrade is a *mutation* if the target and source cells have the same set of attributes but differ on the descriptor values (they are *union compatible,* so to speak, as the term has been used in relational algebra).

Figure 1 illustrates the operations of these cubegrades. Following, I illustrate some specific examples to explain the use of these cubegrades:

Figure 1. Cubegrade: specialization, generalization, and mutation



- (Specialization Cubegrade). The average age of buyers who purchase \$20 to \$30 worth of milk monthly drops by 10% among buyers who also buy cereal.
  (salesMilk=[\$20,\$30]) $\rightarrow$ (salesMilk=[\$20,\$30], salesCereal=[\$1,\$5])
  [AVG(Age), AVG(Age) = 23, DeltaAVG(Age) = 90%]
- (Mutation Cubegrade). The average amount spent on milk drops by 30% when moving from suburban buyers to urban buyers.
  (areaType='suburban') '$\rightarrow$(areaType='urban')
  [AVG(salesMilk), AVG(salesMilk) = \$12.40, DeltaAVG(salesMilk)= 70%]

MAIN THRUST

Similar to association rules (Agrawal & Srikant, 1994), the generation of cubegrades can be divided into two phases: (a) generation of significant cells (rather than frequent sets) satisfying the source cell conditions and (b) computation of cubegrades from the source (instead of computing association rules from frequent sets) satisfying the joint conditions between source and target and target conditions.

The first task is similar to the computation of iceberg cube queries (Beyer & Ramakrishnan, 1999; Han, Pei, Dong, & Wang, 2001; Xin, Han, Li, & Wah, 2003). The fundamental property that allows for pruning in these computations is called *monotonicity* of the query: Let D