Predictive models for multidimensional data when the resolution context changes

José Hernández-Orallo[1], Nicolas Lachiche[2], and Adolfo Martínez-Usó[1]

[1] DSIC, Universitat Politècnica de València
Camí de Vera s/n, 46022, València, Spain
{jorallo,admarus}@dsic.upv.es
[2] ICube, Université de Strasbourg, CNRS
300 Bd Sebastien Brant - BP 10413, F-67412 Illkirch Cedex, France
nicolas.lachiche@unistra.fr

Abstract. Multidimensional data is systematically analysed at multiple granularities by applying aggregate and disaggregate operators (e.g., by the use of OLAP tools). For instance, in a supermarket we may want to predict sales of tomatoes for next week, but we may also be interested in predicting sales for all vegetables (higher up in the product hierarchy) for next Friday (lower down in the time dimension). While the domain and data are the same, the *operating context* is different. We explore two approaches for multidimensional data when predictions have to be made at different levels (or contexts) of aggregation. One approach, which we have called same-level modelling, is based on creating a cube through attribute aggregation at the desired level for the dimension hierarchies, *retraining* the model each time the multidimensional context changes. A second opposite approach, lowest-level modelling, creates one, more reusable, model for the data at the lowest level and then the predictions are *reframed* by aggregation. This important distinction between *aggregate-train-predict* and *train-predict-aggregate* is analysed in terms of advantages and disadvantages, and when they are expected to differ. We perform an experimental analysis with several datamarts and introduce new plots to better compare models under different contexts.
Keywords: multidimensional data, operating context, aggregation, OLAP cubes, quantification.

1 Introduction

Most existing algorithms in machine learning only manipulate data at an individual level (flat data tables), not considering the case of multiple abstract levels for the given data set. However, in many applications, data contains structured information that is multidimensional (or multilevel) in nature, such as retailing, geographic, economic or scientific data. The multidimensional model is a widely extended conceptual model originated in the database literature that can be used to properly capture the multiresolutional character of many data sets [10]. Multidimensional databases arrange data into fact tables and dimensions. A fact table includes instances of facts at the lowest possible level. Each row represents a fact, such as "The sales of product 'Tomato soup 500ml' in store '123' on day '20/06/2014' totalled 25 units". The features (or fields) of a fact table are either measures (indicators such as units, Euros, volumes, etc.) or references to dimensions. A dimension is here understood as a particular variable that has predefined (and hopefully meaningful) levels of aggregation, with a hierarchical structure. Figure 1 shows several examples of dimensions and hierarchies. Using the hierarchies, the data can be aggregated or disaggregated at different granularities. Each of this set of aggregation choices for all dimensions is known as a *data cube* [4]. This approach provides an easy understanding and offers flexibility for visualisation (aggregated tables and cubes). OLAP technology, for instance, has been developed to handle large volumes of multidimensional data in a highly efficient way, and moving through the space of cubes by the use of roll-up, drill-down, slice&dice and pivoting operators.

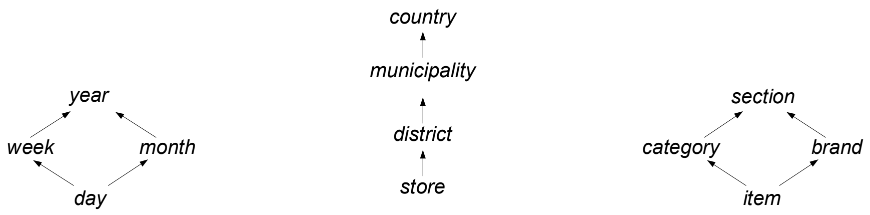

Fig. 1. Examples of dimensions and their hierarchies. Left: Time dimension, Middle: Location dimension, Right: Product dimension.

Despite the success of multidimensional schemas and its widespread use for data warehouses for about two decades, a full integration of machine learning and multidimensional datasets has not taken place. Even in business intelligence tools, which aim at integrating data warehouses, OLAP technology and data mining tools, the usual procedure is to select a cube using an OLAP query or operator, and derive a view from it. Next, this 'minable view' is transferred to the data mining tool to apply machine learning or statistical techniques to this flat, traditional view of the data.

When we analyse the problem more carefully, we see that the main issue for a successful integration is that we would like to use off-the-shelf machine learning techniques but taking full potential of the hierarchical information. Data mining models[3] are not designed to take hierarchical attributes. Consequently, we need to do something different whenever the cube we want to predict for changes. In other words, the predictions for tomatoes and weeks will be obtained in a different way than the predictions for vegetables and Fridays. These two situations represent *operating contexts*. In principle, a model that has been obtained for one context cannot be *directly* applied to a different context.

This leads us to two major alternatives. Either we learn one model for each operating context and apply it for that level of aggregation, or we learn one,

[3] From now on we will use the word 'model' to data mining models, such as a predictive model, and not for conceptual data models any more.

more versatile, model at the lowest operating context (highest resolution) and we aggregate their predictions, as in a quantification problem [8,1,2]. In this paper we analyse these two approaches systematically. In order to do that, we need to find better ways of abstracting and plotting the operating context, such that we can decide for which operating contexts one machine learning technique behaves better than others for both approaches.

The rest of the paper is organised as follows. Section 2 formalises the notion of multidimensional context and properly defines the two main approaches that we will study: the same-level (retraining) approach and the low-level (reframing) approach. Section 3 discusses how datamarts have to be understood when models are required to predict some of the measures of the fact table, and whether the training data has to be populated with zero-valued rows. Section 4 presents the datamarts and techniques that will be used for the experiments. Section 5 introduces the concept of *reduction ratio*, as an abstraction of the multidimensional context. This facilitates the representation of results in what we call *multidimensional context* (*MDC*) plots. Section 6 analyses the results of the same-level and low-level approaches, discussing whether there are general patterns about which method can be best for some contexts. Section 7 discusses some related work. Section 8 closes the paper with some take-away messages and some future work.

2 Multidimensional contexts. Same-level vs. lowest-level

We consider a multidimensional data set D (or datamart) of schema $\langle X, Y\rangle$ where $X = \{X_1, \ldots, X_d\}$ is the set of d dimensions (used as predictor attributes or features) and Y, which is the *target attribute* (one measure or indicator that can be numeric or nominal). Without loss of generality if we had more indicators we could define another datamart with a different target attribute. We use D_A to denote the projection of dataset for attribute A. Note that datasets and projections are multisets (i.e., they can have repeated values). Each dimension X_i has an associated hierarchy $h(X_i)$ of m_i elements or *levels* $\{X_i^{(1)}, \ldots, X_i^{(m_i)}\}$ with a strict partial order $<$. In this paper we will assume that hierarchies are linear, so the partial order becomes a total order from the lowest level $X_i^{(1)}$ to the highest level $X_i^{(m_i)}$. This is not a strong restriction, as a non-linear dimension can be converted into several linear dimensions (one for each possible pathway in the lattice), but simplifies our notation and the understanding of the procedures. For instance, if $X_2 = \mathsf{location}$, as in Figure 1 (middle), we have $X_2^{(1)} = \mathsf{store}$, $X_2^{(2)} = \mathsf{district}$, $X_2^{(3)} = \mathsf{municipality}$ and $X_2^{(4)} = \mathsf{country}$ with $\mathsf{store} < \mathsf{district} < \mathsf{municipality} < \mathsf{country}$ and their transitive closure. We will consider that the top level m_i for every hierarchy is $\mathsf{all\text{-}i}$, such that for every $l \in h(X_i)$, $l < \mathsf{all\text{-}i}$. Non-hierarchical attributes are just special cases, by just considering that $m_i = 2$ (the bottom and the top $\mathsf{all\text{-}i}$ level). These dimensions then just become regular attributes but with the possibility of aggregating them to the top level all.

Each level $X_i^{(j)}$ of a hierarchy $h(X_i)$ has an associated domain $\mathcal{X}_i^{(j)}$, which can be nominal or numeric. We will assume that there are no levels with the same name in the same or different hierarchies. In this way, if the name of a level is name then we can just refer to the level by $X^{(\mathsf{name})}$ and the associated domain by $\mathcal{X}^{(\mathsf{name})}$. For instance, the domain of the level $\mathsf{country}$ for dimension $\mathsf{location}$, i.e., $\mathcal{X}_2^{(4)}$, or $\mathcal{X}^{(\mathsf{country})}$, might be the set with values $\{\mathtt{UK}, \mathtt{Spain}, \mathtt{France}\}$. For every pair of consecutive levels $X_i^{(j)}$ and $X_i^{(j+1)}$ in a hierarchy we define a regrouping function ϕ_i^j between the values of $X_i^{(j)}$ to the values of $X_i^{(j+1)}$. For instance, $\phi_2^3(\mathtt{Valencia}) = \mathtt{Spain}$. We denote by $\phi_i^{j:k}$, with $j \leq k$ the successive application of ϕ from j to k, i.e., $\phi_i^{j:k}(v) = \phi_i^k(...\phi_i^{j+1}(\phi_i^j(v))...)$. Given a value v at a level $X_i^{(k)}$ of the dimension i, we denote by $\perp(v)$ the set of all the values at the lowest level of that hierarchy that belongs to, i.e., $\{w \in \mathcal{X}_i^{(1)} \mid \phi_i^{1:k}(w) = v\}$. For instance, $\perp(\mathtt{Valencia})$ would be all the stores of all the districts of Valencia.

Definition 1. *A multidimensional operating context or resolution is a d-tuple of levels $\langle l_1, \ldots, l_d \rangle$, with each $l_i \in h(X_i)$. A multidimensional context determines the level for every dimension of the dataset.*

Using a multidimensional context, we define a selection of the data set D:

Definition 2. *A selection of D at a context $\langle l_1, \ldots, l_d \rangle$ with values $\langle v_1, \ldots, v_d \rangle$ is defined as follows:*

$$\sigma_{[l_1=v_1,\ldots,l_d=v_d]}(D) \triangleq \{\langle x_1, \ldots, x_d, y \rangle \,|x_1 \in \perp(v_1), \ldots, x_d \in \perp(v_d)\}$$

For instance, if we have three dimensions $X_1 = \mathsf{product}$, $X_2 = \mathsf{location}$ and $X_3 = \mathsf{time}$, and Y is representing units, we could select all the facts for the context $\langle \mathsf{item}, \mathsf{municipality}, \mathsf{year} \rangle$ with values $\mathtt{tomato}$, $\mathtt{Valencia}$ and $\mathsf{2013}$ respectively with $\sigma_{[\mathsf{item}=\mathtt{Tomato}, \mathsf{municipality}=\mathtt{Valencia}, \mathsf{year}=\mathsf{2013}]}(D)$.

Finally, we can define an aggregation operator as follows:

Definition 3. *Given an aggregation function, agg, as a function from sets to real numbers, the aggregation of a datamart D for a context $\langle l_1, \ldots, l_d \rangle$ is:*

$$\gamma^{agg}_{[l_1,\ldots,l_d]}(D) \triangleq \{\langle x_1, \ldots, x_d, z \rangle \mid x_1 \in \mathcal{X}^{(l_1)}, \ldots, x_d \in \mathcal{X}^{(l_d)}, \\ z = agg(\{y| \,\langle v_1, \ldots, v_d, y \rangle \in \sigma_{[l_1=x_1,\ldots,l_d=x_d]}(D)\})\}$$

The above aggregation is extended for unlabelled datasets with no y attribute.

For instance, $\gamma^{sum}_{[\mathsf{item}, \mathsf{municipality}, \mathsf{year}]}(D)$ returns all the tuples for each possible combinations of values at the level item in the dimension $\mathsf{product}$, at the level $\mathsf{municipality}$ in the dimension $\mathsf{location}$ and at the level year in the dimension time, where the output variable is constructed by summing all the y of the corresponding rows according to the hierarchies.

Given the above notation, now we consider a predictive problem from X to Y. For instance, how many tomatoes we expect to sell in Valencia next week? Assuming we have a training dataset, how would we train our model? As a first idea, it seems reasonable to aggregate the training data using the γ operator above for the context $c = \langle \mathsf{item}, \mathsf{municipality}, \mathsf{week} \rangle$, producing a model M that will be applied to the deployment data with the same context. However, if some time later we are interested in predicting sales for all vegetables for next Friday, what would we do? We could aggregate the training data for the context $c' = \langle \mathsf{category}, \mathsf{municipality}, \mathsf{day} \rangle$, learn a new model M' and predict for the deployment data. This is what we see in Figure 2 (top). We refer to this approach as the *Same-Level* (SL) approach or the *retraining* approach.

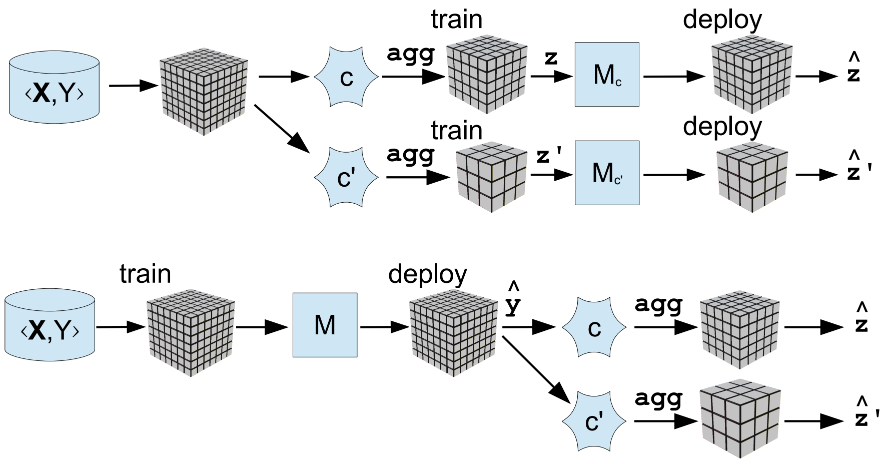

Fig. 2. Retraining (*Same-Level* approach) vs. reframing (*Lowest-Level* approach) for two different multidimensional contexts c and c'. Retraining (top) needs to convert the training data for the two contexts c and c' and then aggregating the output into z or z' respectively. Two models M_c and $M_{c'}$ are learnt (one for each context) at the same level the predictions must be done. Reframing (bottom) shows how the training data is used just once at the lowest level to create a single model M that is applied to different operating contexts c or c' by aggregating the outputs appropriately.

But an alternative approach goes as follows. Consider that we train a predictive model M for the lowest level in D. Once a new multidimensional appears, we apply the model to the deployment data and *aggregate the predictions*. With this approach, one model is used for every possible context. This is illustrated in Figure 2 (bottom). We refer to this approach as the *Lowest-Level* (LL) approach or the *reframing* approach.

So the difference is between aggregate-train-predict or train-predict-aggregate. What method is best? Before analysing this in terms of prediction error, we discuss here the implications of each method in real practice. A datamart with d dimensions and an average hierarchy depth of n, has $d \times n$ operating contexts. This means that the *retraining* approach will require many models to be created and validated, which may be infeasible in many contexts. For instance, in Section 4, where the datamarts for our experimental analysis are described, the number of operating contexts are 48 (GENOMICS), 84 (AROMA) and 96 (CARS). Note that we also consider the case where the dimension attribute has been rolled up completely (the level all-i)).

3 Learning tasks, measure properties and mean models

While multidimensional data occurs in many different scenarios, we will focus on data that originates from datamarts or has been converted into a multidimensional schema. This focus is motivated by our aim of better integrating data warehouses and OLAP tools with machine learning.

The first thing we need to consider is the kind of machine learning tasks that are common with this data. The way the information is arranged in a multidimensional schema, with a fact table containing *measures* suggests that many machine learning tasks, especially predictive ones, are usually focussed on predicting the measures. For instance, if facts are sales, consumptions, failures, usages, etc., it is common to become interested in predicting some of the measures in these tables (e.g., units, dollars, hours, etc.) from past data. As measures are usually numerical, many problems will turn out to be regression problems. Nonetheless, some measures can be nominal, such as whether a purchase has been satisfactory or not. In that case, however, the measure becomes a percentage, i.e., a number, when we aggregate, so binary nominal measures can also be taken as numbers.

The time dimension is found in most datamarts. In a predictive scenario, the time dimension becomes slightly special: predictions are about future facts, so training is usually performed with available data up to a given time and the model is then used to extrapolate from that point on (next week, next month, next year, depending on the resolution).

Finally, there is an important issue that has to be discussed before building predicted models using multidimensional data. Some multidimensional schemas only represent positive data. For instance, in a supermarket datamart, we may have that 3 units of product X (e.g., bottles of shampoo) were sold in supermarket Y on day Z. This is going to appear as a row in our fact table. However, if no units are sold on day Z', is this row going to appear with a zero? Depending on the source of the data (OLAP tool, `group by` SQL query, or other), these rows may be present or not. The degree of sparseness may also be an important reason why they are not present, as only including the positive cases may imply a significant space economy. In any case, it would be a great mistake not to consider them, as only using positive facts would imply a very strong bias. As a result, we will need to ensure that these zeros are present. If they are not, we will need to create them, even if the dataset grows very significantly. Nonetheless, it is not always the case that a row that is not present has to be associated with a zero. For instance, if facts are opinion surveys over a population, the rows in the fact table will represent those that have taken the survey and the measures will be their answers. For the rest of individuals in the population, the measures are unknown, but not zero, as they never took the survey.

Another important issue about multidimensional schemas is whether the measures we want to predict are *additive*, *semi-additive* or *non-additive*. A measure is *additive* when, for any dimension and any set of values S at level j that we want to aggregate up to level $n+k$, the summation of these values using any partition of S is equivalent, i.e., gives consistent results. For instance, units sold in a supermarket aggregate well for all dimensions, i.e., the result is independent of the way it has been aggregated. However, percentages do not aggregate well, as the denominator is not known when performing the aggregation. Therefore percentages are *non-additive*. Finally, the term *semi-additive* measure is used for those measures that aggregate well for some dimensions but not for others. For instance, measures that accumulate or depend on the state, such as stock levels are usually semi-additive.

The aggregated function that is used for aggregating datamarts, as in Definition 3, does not have to always be $sum(S) \triangleq \sum_{s \in S} s$. For instance, it could be an average, $avg(S) \triangleq \frac{sum(S)}{|S|}$. Some functions just work for some measures. For instance, consumption (e.g., in kWh) can be aggregated by averaging it. However, we have to be very careful about how this aggregation is performed. For instance, if we have three households with consumption 3, 5, 10 for day D, and the first two are in region A and the last one is in region B, we have that the average consumption is 4 for region A and 10 for region B. If we further aggregate, we get an overall average consumption of 7, which is wrong, as the true average consumption is $(3+5+10)/3 = 6$. This means that *avg* is not *composable*. This is important in terms of the approaches we explore in this paper (see Figure 2), as both of them (retraining at the same level and reframing from the lowest level) do not require the associativity property.

In regression tasks, we usually look at a baseline method that consists in averaging the values for the training data and apply these values systematically during deployment. This is known as the *mean* or *constant* model. Two baseline approaches are possible with the *Lowest-Level* and the *Same-Level* philosophies.

Definition 4. *Given a training data T with measure Y and a deployment data D, the* MEAN *model for measure Y at the* lowest level*, denoted by* LL.MEAN*, and deployed at a context $\langle l_1, \ldots, l_d \rangle$ is defined as follows. We first calculate $\overline{Y} \triangleq avg(T_Y)$, the average of the measure Y for the whole training dataset T. Now, for the deployment data D we add a new attribute $\hat{Y}$, which is set to $\overline{Y}$ for all the rows in D, giving a new dataset $\hat{D}$. Finally, given an aggregation function agg, we now calculate the predictions for a context $\langle l_1, \ldots, l_d \rangle$ as $\hat{D}^{\Delta} \triangleq \gamma^{agg}_{[l_1,\ldots,l_d]}(\hat{D})$, which produces pairs $\langle X, \hat{Y} \rangle$ at that context.*

Definition 5. *Given a training data T with measure Y and a deployment data D, the* MEAN *model for measure Y at the* same level *in context $\langle l_1, \ldots, l_d \rangle$, denoted by* SL.MEAN*, is defined as follows. We first aggregate T for that context, i.e., $T^{\Delta} \triangleq \gamma^{agg}_{[l_1,\ldots,l_d]}(T)$. Then we calculate $\overline{Y^{\Delta}} \triangleq avg(T^{\Delta}{}_Y)$, the average of the measure Y for this context. For the deployment data D we also aggregate the original data as $D^{\Delta} \triangleq \gamma^{agg}_{[l_1,\ldots,l_d]}(D)$. We finally add an attribute $\hat{Y}$ to D^{Δ} by* *setting it equal to $\overline{Y^{\Delta}}$ for every row in D^{Δ}, so producing $\hat{D}^{\Delta}$. This produces pairs $\langle X, \hat{Y} \rangle$ at that context.*

How different are they? It is easy to find examples where they differ. For instance, consider a very small training dataset T with one dimension (product) and just three examples: 3 tomatoes, 5 pumpkins and 10 cakes. If the multidimensional context is category, we have that the LL.MEAN will predict $(3+5+10)/3 = 6$ for every row at the bottom level of the deployment dataset D. If D also has two products in the category `vegetables` and one product in the category `bakery`, then the prediction will be 12 for the first category and 6 for the second category. On the other hand, if we use SL.MEAN we first aggregate and get $3+5 = 8$ for category `vegetables` and 10 for the category `bakery`. The average for this context in T is 9. This means that for D the prediction will be 9 for the first category and 9 for the second category. Note that the total is the same for both approaches, but the predictions are different.

Both approaches only match in very specific situations. For instance, if the context is balanced for each hierarchy it aggregates (i.e., the same number of rows comes from each aggregation) for both the training and deployment datasets, then the aggregation on the deployment dataset of the average at the lowest level for the training dataset and the average of the aggregation at the same level for the training set will match. This happens, for instance, when both training and deployment datasets cover all combinations (either originally or because it has been filled with zeros) and when the context only includes top levels or bottom levels in the hierarchy. In general, however, both approaches are different.

Note that when differ, SL.MEAN seems to be more consistent. It is still a *constant* model for the multidimensional context we are interested in.

4 Experimental setting

The MEAN approach is useful as a baseline, but we of course are interested in the use of machine learning methods to get good predictions. Including the MEAN approach as given by definitions 4 and 5, we have considered four techniques in total. The other three are LRW (linear regression using RWeka in R [11,18]), M5P (regression tree using RWeka) and KNN (package `kknn` in R). For the techniques LRW, M5P and KNN, we can apply reframing or retraining in the same way we did for the MEAN approach in definitions 4 and 5.

Definition 6. *Given a regression technique* TECH*, a training data T with measure Y and a deployment data D, a model learnt for measure Y at the* lowest level*, denoted by* LL.TECH*, and deployed at a context $\langle l_1, \ldots, l_d \rangle$ is defined as follows. We first train a model $M : X \to Y$ for the whole training dataset T. Now, for each row at the lowest level in the deployment data D we apply M. We add a new attribute $\hat{Y}$, and set it to the result of the model for each row, giving a new dataset $\hat{D}$. Finally, given an aggregation function agg, we now calculate the predictions for a context $\langle l_1, \ldots, l_d \rangle$ as $\hat{D}^{\Delta} \triangleq \gamma^{agg}_{[l_1,\ldots,l_d]}(\hat{D})$, which produces pairs $\langle X, \hat{Y} \rangle$ at that context.*

Definition 7. *Given a regression technique* TECH*, a training data T with measure Y and a deployment data D, a model learnt for measure Y at the* same level *in context $\langle l_1, \ldots, l_d \rangle$, denoted by* SL.TECH*, is defined as follows. We first aggregate T for that context, i.e., $T^{\Delta} \triangleq \gamma^{agg}_{[l_1,\ldots,l_d]}(T)$. Then we train a model $M^{\Delta} : X^{\Delta} \rightarrow Y^{\Delta}$. For the deployment data D we also aggregate the original data as $D^{\Delta} \triangleq \gamma^{agg}_{[l_1,\ldots,l_d]}(D)$. We finally add an attribute $\hat{Y}$ to D^{Δ} by setting it equal to the predictions of the model M^{Δ} for each of these aggregated rows, so producing $\hat{D}^{\Delta}$. This yields pairs $\langle X, \hat{Y} \rangle$ at that context.*

Once we have set the four techniques we will use and the two variants (retraining and reframing), we present the datamarts for our experimental analysis:

- **GENOMICS**: This database was meant to be a unified genomic variation repository to allow biologists to perform efficient recovery tasks about genomic variations (mutations) and their phenotype [17]. Originally, this human genome dataset contains genomic data (HGDB) from several public and private research databases, including information about genes, chromosomes, mutations, diseases, etc. structured in 20 (numerical and nominal) attributes. We converted it into a multidimensional datamart, where each fact showed the number of variations according to five different dimensions (hierarchies in parenthesis): SPEC (Eff < All), GENOTYPE (ID < Chrom < All), PHENOTYPE (Name < ICD10 < ICD10.Cat < All), DBANK (Dbnk < All) and DATE (Year). Note that as we use the DATE dimension to split the data we only consider one level here. The number of possible multidimensional contexts is then $2 \times 3 \times 4 \times 2 \times 1 = 48$. Data goes from years 1970 to 2012 and the output variable is the number of variations.
- **AROMA**: This is an artificial dataset constructed from IBM sales information. It contains sales data for coffee and tea products sold in stores across the United States [13]. The data is almost directly converted into a multidimensional datamart where each fact describes the sales of products using two measures (units and dollars, although we will only use dollars as the output variable) according to five dimensions (hierarchies in parenthesis): PROMO (KeyPromo < PromoType < All), CLASS (KeyClass < All), PRODUCT (KeyProduct < All), STORE (KeyStore < KeyMKT < MKT-HQ-City < MKT-HQ-State < MKT-District < MKT-Region < All) and PERIOD (Year). Note that as we use the PERIOD dimension to split the data we only consider one level here. Data goes from years 2004 to 2006 and the number of possible multidimensional contexts is $3 \times 2 \times 2 \times 7 \times 1 = 84$.
- **CARS**: This is a dataset for car fuel consumption and emissions which is created as a reduced representation of [7] (some attributes are removed) in order to construct a datamart. It describes fuel consumption in cars from years 2000 to 2013, being published by the Vehicle Certification Agency (VCA), an Executive Agency of the United Kingdom Department for Transport. The target variable is car fuel emissions (CO_2) and we have six dimensions (hierarchies in parenthesis): CAR (Man.Model.Description < Man.Model < Manufacturer < All), ENGINE (EngineCapacity < All), TRANS (Transmission <

TransType $<$ All), EURO (EuroSTD $<$ All), FUEL (FuelType $<$ All) and TIME (Year). Note that as we use the TIME dimension to split the data we only consider one level here. The number of possible multidimensional contexts is $4 \times 2 \times 3 \times 2 \times 2 \times 1 = 96$. No rows with zeros were added here, as missing cases are just absence of information. The target variable is a ratio, so the aggregation function that makes sense for this dataset is *avg*, which is neither additive nor associative.

We split these datasets into training and test on the basis of a *split-year*, which is 2006 for all the datasets, being the *split-year* included in the test set.

Finally, we clip the predictions of all methods to 0 if they are negative, as in the three datamarts the measures cannot be negative. This is important for methods that could potentially predict negative values such as M5P or LRW.

5 Multidimensional context plots

We now start by analysing the results. The first thing to decide is the error measure. As the three datamarts have led to regression problems, we may use the Mean Squared Error (MSE). However, for the two datamarts that use *sum* as aggregating function, the magnitude of the error will be much higher for highly aggregated contexts, and the values will be difficult to compare. As an alternative, we could use the Squared Error (SE), i.e., without averaging by the number of examples. This measure is interesting in our scenario, as the higher the aggregation the higher the magnitudes but the number of rows decreases, so the magnitudes will be comparable. This is not completely true, as squared error usually penalises errors of high magnitudes (by squaring them), so the contexts on the upper levels of the hierarchy will still have higher values for SE.

For instance, we can plot SE (as shown on the y-axis) and the contexts sequentially on the x-axis, as in Figure 3.

Yet another option is to use the Absolute Error (AE). A good way of getting rid of all these dilemmas is to divide SE (or MSE) by the SE (or MSE) of the MEAN model. Interestingly, in a classical regression setting, the MSE of the MEAN model equals its error variance. So, actually, what we are doing is to show the MSE by some kind of error variance. We use the SL.MEAN model, as it ensures that it is constant for the deployment multidimensional context.

Definition 8. *The normalised squared error (NSE) of a method* MET *is defined as* $\frac{MSE(\mathsf{MET})}{MSE(\mathsf{SL.MEAN})}$.

If we use NSE everything is normalised so we just see whether they are better or not than the mean model. In fact, in regression quantification (which is actually what we are doing here), a similar measure, known as VSE (page 18 of [2]) is used, which is the SE divided by the error variance.

However, the main problem of Figure 3 is that the order of the dimensions and levels is not very meaningful, and plots look very bumpy. As an alternative we summarise the degree of aggregation of a context by the following indicator:

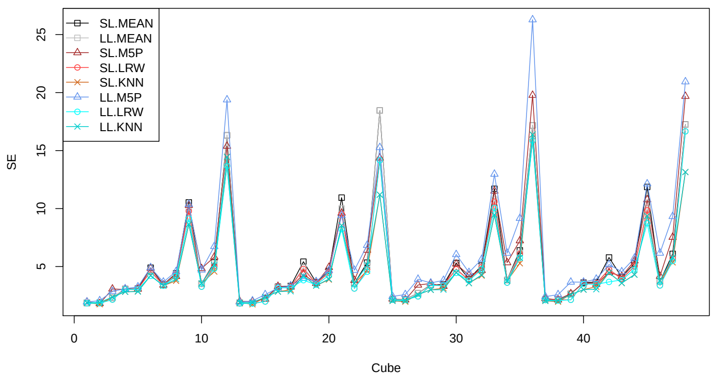

Fig. 3. *SE* values for the GENOMICS dataset. Comparison of a retraining approach (SL) with a reframing approach (LL) for several techniques: MEAN, M5P, LRW and KNN. Note here that the x-axis of the graph shows the 48 cubes sequentially as they are generated by iterating over the dimensions.

Definition 9. *Given a deployment dataset D and a multidimensional context $c = \langle l_1, \ldots, l_d \rangle$, we define the Reduction Coefficient (RC) as follows:*

$$RC \triangleq 1 - \frac{\log(|D^{\Delta}|)}{\log(|D|)} \quad \text{where} \quad D^{\Delta} = \gamma^{agg}_{[l_1,\ldots,l_d]}(D). \tag{1}$$

RC goes from 0 (no aggregation, i.e., the bottom level) to 1 (all rows are collapsed into one row). Note that the two different contexts may have similar (or even equal) RC. The RC is a very useful way of locating each point in the plot according to how much aggregated the context is. This allows for more regular plots and a common scale [0,1] for all datasets. Combining the NSE with RC, we have the multidimensional context plots:

Definition 10. *A multidimensional context (MDC) plot represents RC on the x-axis for all the possible multidimensional contexts in the datamart with NSE in the y-axis.*

Figure 4 shows the *MDC* plots for the GENOMICS (top), AROMA (middle) and CARS (bottom) datamarts. These new plots show how we have different variabilities in results as we move to a higher aggregations (RC closer to 1). For instance, higher variability can be shown for GENOMICS whereas this variability decreases for AROMA, being fairly constant for CARS. It is important to stress at this point that curves or lines in the graphs do not represent any possible or meaningful interpolation between the points. However, we keep these lines in the graphs in order to provide to the reader a much easier way of following the evolution of each method than just with points. In addition, as a matter of better distinguishing between SL and LL strategies, those methods applied on a SL basis have been plotted using “heat” colours whereas those ones on a LL basis have been plotted using “cold” colours.

For GENOMICS, some methods behave much better than the mean (e.g., the KNN methods or the LL.LRW), but there are also some methods that behave much worse. For AROMA, we observe a different behaviour, as there is higher variability in the middle, for values of RC around 0.5. We see that KNN only behaves well for high aggregations. The CARS datamart does not exhibit the same behaviour since it is very different from the other two. We do not have the sparseness, and this allows the models to make better predictions. In fact, as the *Lowest-Level* context is not full of zeros, the *Lowest-Level* model works well for many techniques even for high values of RC. We can see that LL.KNN is a very good technique overall for this dataset.

The sparseness of both GENOMICS and AROMA is probably responsible of the poor results of some methods (worse than the MEAN method). As most regression techniques are designed to minimise the squared error, when there are many zeros, the models will tend to predict values very close to 0. For the *Lowest-Level* approaches, any bias in these predictions will accumulate further up and will lead to high error. For the *Same-Level* approaches, things are different as the models are learnt from aggregated data, and many rows will be aggregated into single rows with measures that are no longer zero.

Finally, even if in the previous section we discussed that LL.MEAN and SL.MEAN would give different results in general, we see that both plots are almost identical for most contexts. Nevertheless, in AROMA, there are contexts (from 0.3 to 0.9) where LL.MEAN is much worse (almost three times worse) than the SL.MEAN.

6 Discussion

The *MDC* plots are useful to see the operating contexts where some techniques are better than others. The usefulness of these plots is then further justified as it is clear that no technique and no approach is consistently better for all operating contexts. As usual, this is dataset-dependent, and the metrics and plots we have introduced in the previous section can be very useful to assess models and to perform model selection. Nonetheless, it may be interesting at this point to make a summary in terms of retraining vs. reframing. One option could be to define an envelop measure (graphical representation) for *MDC* plots, assuming that the distribution of contexts is uniform w.r.t. RC. While this may be useful as a single metric, we take a different approach below.

Table 1 shows the pairwise comparison results between LL and SL for the three datasets. Again, for the GENOMICS and AROMA datasets we see that the SL approach is better. Only LL for LRW and KNN, seem to be at least comparable to their SL counterparts. A very different picture happens for CARS.

Another possible reason for the behavioural differences of each methodology is the *learnability* of each dataset. The CARS dataset is a machine learning benchmark; as such one could expect that the output depends on the inputs variables and therefore a model can be learnt. We do not have such guarantee for the AROMA dataset or for the GENOMICS dataset.

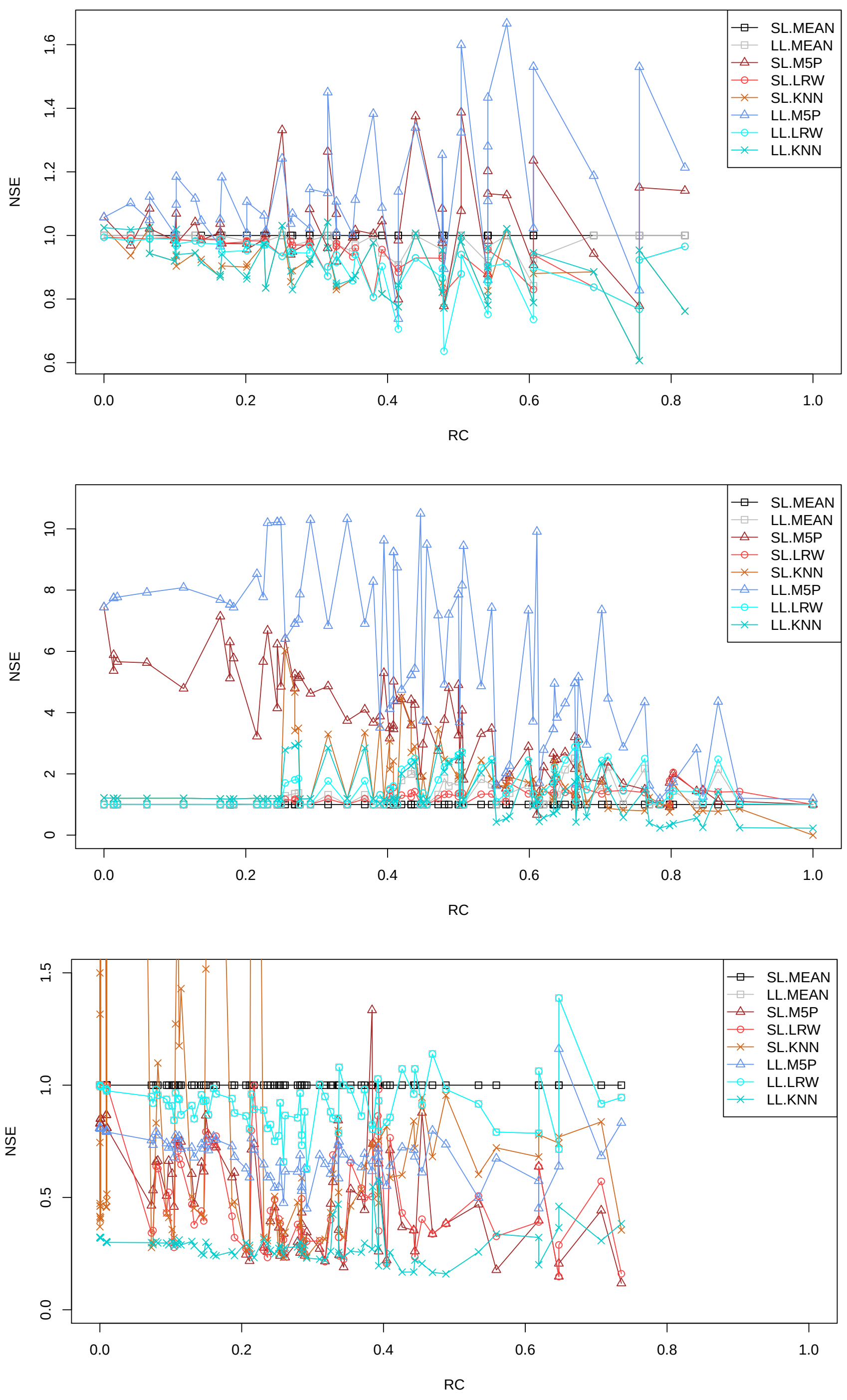

Fig. 4. *MDC* plots for GENOMICS (top) AROMA (middle) and CARS (bottom) datasets. Comparison of a retraining approach (SL) with a reframing approach (LL) for several techniques: MEAN, M5P, LRW and KNN. The y-axis shows *NSE*. This axis has been re-scaled for CARS in order to yield a more detailed view of all methodologies, leaving SL.KNN out of focus. Note that only the AROMA dataset reaches $RC = 1$ (all rows collapsed into one) since in GENOMICS and CARS the time dimension is not completely collapsed (there are several years in the test set). This does not happen in AROMA because it only has one year in the test set.

	GENOMICS		AROMA		CARS	
	LL	SL	LL	SL	LL	SL
MEAN	0.97	1.00	1.31	1.00	0.93	1.00
M5P	1.14	1.03	5.81	3.44	0.71	0.57
LRW	0.91	0.94	1.56	1.24	0.93	0.60
KNN	0.90	0.89	1.36	1.74	0.29	1.31
Overall	0.98	0.97	2.51	1.86	0.71	0.87
N.Cubes	79	56	77	243	224	150

Table 1. Pairwise comparison between LL and SL for the GENOMICS, AROMA and CARS datasets. The first four rows of the table show the average of the NSE measure (per context) for each strategy. The 'Overall" row shows the overall average. Finally, the 'N.Cubes' row shows how many cubes of the corresponding strategy win, aggregating all methods. This last row does not consider those cubes where LL and SL draw.

7 Related Work

As we mentioned in the introduction, the efforts for a full integration of data mining and OLAP tools have not been as common as originally expected. There are, though, some significant contributions for descriptive models. For instance, multidimensional association rules were firstly introduced in [14] and, since then, some related approaches have appeared in areas such as hierarchical association rules, subgroup discovery, granular computing [15] and others [5].

'Prediction cubes' [6,5], despite the term 'predictive' in their name, actually perform subgroup discovery or exploratory mining [19], where we want to have a metric (e.g., predictive accuracy) for a model on a given subset of the data (a cell in a cube) and see whether some cells have different metrics than others (hence being special). It is important to note that "Prediction Cubes" are not meant to aggregate outputs. They are not actually used to make predictions at several resolution levels of values that are unknown. In fact, they always work with a labelled test set to which they compare to get the metrics.

When looking at predictive modelling, the usual approach in the literature has been the *same level* approach (i.e., generating a view for the resolutions at hand). There is no versatile model that can work for the whole hierarchy in every dimension. A significant exception is the area of multilevel modelling (MLM) [3,9], also known as hierarchical (linear) modelling (HLM) [20], among other names. This is an extension of linear, and non-linear, models such that the variables are measured at different levels of a global, usually linear, hierarchy. The first and key difference between a multilevel modelling problem and a multidimensional problem is hence that in the latter all the measurements take place at the lowest level (e.g., they come from facts in a multidimensional data warehouse). However, in multilevel modelling, measurements may take place at any level. As a result, in multilevel modelling, putting all the variables at the lowest level does not make sense, as it means that some of the input variables would have to be disaggregated (or repeated). The second difference is that in multilevel modelling hierarchies apply to *all* attributes. In other words, there is an orthogonal hierarchy, which can be applied to each attribute, depending on the level at which the value has been measured. So it is not actually applicable to a multidimensional database, where each attribute can be aggregated independently. The third difference is that in multilevel modelling the predictions are still made generally at the lowest level. In a multidimensional setting we want predictions at whatever level of aggregation. In addition, multilevel modelling has usually been addressed by linear (and occasionally non-linear) regression models with several assumptions about normality, homoscedasticity, independence, etc. Because of the differences, there are only a few attempts that have been done to apply multilevel models to multidimensional data. For instance, in [16], multilevel models are applied to a datamart. However, we still see a separate concept hierarchy that is applied to all dimensions, instead of having a particular hierarchy for each dimension, as usual in datamarts and OLAP tools, so it is not actually a multidimensional database.

As a result, the problem of having several hierarchies, one for each dimension and seeing the problem (including predictions) at any possible resolution, is new. Also there is no general approach about how to apply any data mining technique to this kind of problem (and not only linear regression models or non-linear variants). So, the multidimensional approach presented in this paper is more general in at least these two aspects.

8 Conclusions and future work

Multidimensional data is a rich and complex scenario where the same task can change significantly depending on the level of aggregation over some of the dimensions. This is the 'multidimensional context'. From this notion of context, we have seen that there are two options whenever the context changes: to adapt or reframe an existing model to a new context (LL) or to retrain a new model for the context (SL). This dilemma between reframing and retraining has been systematically explored in this paper. We have also identified different types of datamarts, in terms of sparseness and aggregating function, and we have introduced new plots and metrics to analyse the behaviour of several techniques when the multidimensional context changes. Our approach is very general, and applicable to any set of off-the-shelf machine learning techniques.

From our experimental results, we see that the best choice is dependent on the dataset and the learning technique that is used. Nonetheless, there is still a simple take-away message. If the datamart is sparse, SL could work better. Otherwise LL is a much better option. The use of *MDC* plots in any case is recommended, in order to spot the contexts where a pair of approach and technique is better than the rest. Finally, resources are an important criterion for how to proceed, as retraining a model again and again may become infeasible for some applications, and reframing a single, versatile model may be a much better option in cost-effective terms.

A more sophisticated approach to select between SL and LL would be to perform a cost-benefit analysis of reframing against retraining, if the costs of retraining were available. Also, the assumption of a uniform distribution could be changed by other distributions. For instance, if the dataset is too big, learning at the lowest-level may have a high computational cost or even be infeasible. Also, when dimensions are not linear, we may consider that all the possible pathways do not have the same probability.

This work suggests many avenues for future work. One area for improvement is the choice of other kinds of plots and metrics. For instance, the squared error on the y-axis could be replaced by relative measures, so that comparison between datasets is easier. This is evident in Figure 4, where the y-axis could have the same scale instead of the different scales we see now. The use of the logarithmic transformation for the reduction coefficient is just an arbitrary choice to set it between 0 and 1 but other possible (and perhaps better justified) options could be considered. The second area we are undertaking is a modification of the LL approach where the aggregation function is substituted by a quantification procedure [8,1]. As quantification is able to correct some aggregation problems, we hope some quantification techniques (especially those for regression using crisp regression models [2] or soft regression models [12]) to be beneficial for the LL approach. A third idea to consider is disaggregation (using frequency counts), as it may be more robust for some contexts to work at an upper level and then disaggregate. Of course, it does not make sense to learn a single model at the *Highest-Level* and disaggregate it for any other context. Instead, we could break the dichotomy between LL (or *Highest-Level*) and SL generating models as a subset of the most representative levels (covering the context space) and using these models to aggregate and disaggregate accordingly. The set of predictions could be used as an ensemble, hopefully leading to better results. Finally, even if the approaches analysed in this paper are general to work for any off-the-shelf machine learning techniques, there may be room for improvement if specific techniques are developed for the multidimensional setting: multidimensional KNN, multidimensional decision trees and multidimensional Naive Bayes.

Acknowledgements

This work was supported by the Spanish MINECO under grants TIN 2010-21062-C02-02 and TIN 2013-45732-C4-1-P, and the REFRAME project, granted by the European Coordinated Research on Long-term Challenges in Information and Communication Sciences Technologies ERA-Net (CHIST-ERA), and funded by MINECO in Spain (PCIN-2013-037). We thank the reviewers for some comments about a cost-benefit analysis, the use of other possibilities for normalising the metrics, the issue of non-linear hierarchies, etc.

References

1. Bella, A., Ferri, C., Hernández-Orallo, J., Ramírez-Quintana, M.: Quantification via probability estimators. In: IEEE ICDM. pp. 737–742 (2010) 3, 16
2. Bella, A., Ferri, C., Hernández-Orallo, J., Ramírez-Quintana, M.J.: Aggregative quantification for regression. DMKD 28(2), 475–518 (2014) 3, 10, 16

3. Bickel, R.: Multilevel analysis for applied research: It's just regression! Guilford Press (2012) 14
4. Chaudhuri, S., Dayal, U.: An overview of data warehousing and OLAP technology. ACM Sigmod record 26(1), 65–74 (1997) 2
5. Chen, B.C.: Cube-space data mining. ProQuest (2008) 14
6. Chen, B.C., Chen, L., Lin, Y., Ramakrishnan, R.: Prediction cubes. In: Proc. of the 31st Intl. Conf. on Very large data bases. pp. 982–993 (2005) 14
7. Datahub: Car fuel consumptions and emissions 2000-2013 (2013), `http://datahub.io/dataset/car-fuel-consumptions-and-emissions` 9
8. Forman, G.: Quantifying counts and costs via classification. Data Min. Knowl. Discov. 17(2), 164–206 (2008) 3, 16
9. Goldstein, H.: Multilevel statistical models, vol. 922. John Wiley & Sons (2011) 14
10. Golfarelli, M., Maio, D., Rizzi, S.: The dimensional fact model: a conceptual model for data warehouses. Intl. J. of Coop. Information Systems 7, 215–247 (1998) 1
11. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: An update. SIGKDD Explor. 11(1), 10–18 (2009) 8
12. Hernández-Orallo, J.: Probabilistic reframing for cost-sensitive regression. ACM Transactions on Knowledge Discovery from Data, 8(3), to appear (2014) 16
13. IBM Corporation: Introduction to Aroma and SQL (2006), `http://www.ibm.com/developerworks/data/tutorials/dm0607cao/dm0607cao.html` 9
14. Kamber, M., Jenny, J.H., Chiang, Y., Han, J., Chiang, J.Y.: Metarule-guided mining of multi-dimensional association rules using data cubes. In: KDD. pp. 207–210 (1997) 14
15. Lin, T., Yao, Y., Zadeh, L.: Data Mining, Rough Sets and Granular Computing. Studies in Fuzziness and Soft Computing, Physica-Verlag HD (2002) 14
16. Páircéir, R., McClean, S., Scotney, B.: Discovery of multi-level rules and exceptions from a distributed database. In: Proc. of the 6th ACM SIGKDD Intl. Conf. on Knowledge discovery and data mining. pp. 523–532. ACM (2000) 15
17. Pastor, O., Casamayor, J., Celma, M., Mota, L., Pastor, M., Levin, A.: Conceptual modeling of human genome: Integration challenges. LNCS, vol. 7260, pp. 231–250 (2012) 9
18. R Team, et al.: R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria (2012) 8
19. Ramakrishnan, R., Chen, B.C.: Exploratory mining in cube space. Data Mining and Knowledge Discovery 15(1), 29–54 (2007) 14
20. Raudenbush, S.W., Bryk, A.S.: Hierarchical linear models: Applications and data analysis methods, vol. 1. Sage (2002) 14