Modular Neural Networks for Extending OLAP to Prediction

Wiem Abdelbaki[1,2(✉)], Sadok Ben Yahia[1], and Riadh Ben Messaoud[3]

[1] Faculty of Sciences of Tunis, University of Tunis El-Manar, LIPAH-LR 11ES14, 2092 Tunis, Tunisia
sadok.benyahia@fst.rnu.tn

[2] Department of Information Systems, College of Economics Management and Information Systems, University of Nizwa, 616 Nizwa, Nizwa, Sultanate of Oman
wiem.abdelbaki@gmail.com

[3] Faculty of Economics and Management of Nabeul, University of Carthage, 8000 Nabeul, Tunisia
riadh.benmessaoud@fsegn.rnu.tn

Abstract. On-line Analytical Processing (OLAP) represents a good applications package to explore and navigate into data cubes. Though, it is limited to exploratory tasks. It does not assist the decision maker in performing information investigation. Thus, various studies have been trying to extend OLAP to new capabilities by coupling it with data mining algorithms.

Our current proposal stands within this trend. It has two major contributions. First, a Multi-perspectives Cube Exploration Framework (MCEF) is introduced. It is a generalized framework designed to assist the application of classical data mining algorithm on OLAP cubes. Second, a Neural Approach for Prediction over High-dimensional Cubes (NAP-HC) is also introduced, which extends Modular Neural Networks (MNN)s architecture to multidimensional context of OLAP cubes, to predict non-existent measures. A preprocessing stage is embedded in NAP-HC to assist it in facing up the challenges arising from the particularity of OLAP cubes. It consists of an OLAP oriented cube exploration strategy coupled with a dimensions reduction step that reposes on the Principal Component Analysis (PCA). Carried out experiments highlight the efficiency of MCEF in assisting the application of MNNs on OLAP cubes and the high predictive capabilities of NAP-HC.

Keywords: Data warehouse · OLAP · Data mining · Principal Component Analysis · Multilayer Perceptrons · Modular Neural Networks

1 Introduction

Data warehouses are the corner stone in the Business Intelligence (BI) roadmap. They are used to store analysis contexts within multidimensional data structures referred to as *Data Cubes* [1]. They are usually manipulated through On-line Analytical Processing (OLAP) applications to enable senior managers exploring information and getting BI reportings through interactive dashboards.

A. Hameurlain et al. (Eds.): TLDKS XXI, LNCS 9260, pp. 73–93, 2015.
DOI: 10.1007/978-3-662-47804-2_4

Needless to mention that OLAP tools provide efficient solutions to navigate through data cubes. However, it is restricted to exploration tasks. Goil and Choudhary argue that coupling OLAP with data mining techniques increases its efficiency [2], and enables it to assist decision makers in performing advanced knowledge discovery tasks. Since then, several studies put the focus on enhancing OLAP by coupling it with data mining techniques to respond to various analysis purposes, e.g. cube exploration [3] and association rule mining [4].

Nevertheless, despite the fact that, data warehouses should fundamentally contain integrated data [1], generally, data cubes exploration discloses sparse structures within several empty measures. In this respect, empty measures correspond to non-existent facts, reflecting either out-of-date events that did not happen, or future events that have not yet occurred and may happen in the future. Empty measures represent a source of frustration for the enterprise management, especially when strategic decisions need to be taken.

Predicting non-existent measures would consolidate BI reporting. It would even provide new opportunities to BI analysts by enlarging their dashboard picture and empowering them with knowledge on what may occur if non-existent facts had already happened. For instance, it will be very useful to a car Sale Company to predict the potential turnover that a new agency could produce in a new city by the end of next year. This indicator will definitely help the company's management to assess the potential investment.

Despite the fundamental Cood's statement of goal seeking analysis models (such as "What if" analysis) required in OLAP applications since the early 90's [5], most of the recent OLAP products still lack an effective implementation of this feature. Recently, new approaches have been attempting to extend OLAP to prediction capabilities [6,7]. However, to the best of our knowledge; none of them provides BI analysts with explicit values of non-existent measures.

The current work fits within the approaches trying to extend OLAP to advanced abilities by coupling it with data mining techniques. It introduces two main contributions. The first one consists of a novel generalized framework, called *Multi-perspectives Cube Exploration Framework* (MCEF). It is designed to enable the application of classical data mining techniques on OLAP cubes. As for the second contribution, it consists of a measure prediction technique, called *Neural Approach for Prediction over High-dimensional Cubes* (NAP-HC). It is based on Modular Neural Networks (MNN)s and designed under the MCEF formalism.

This paper is organized as follows. In Sect. 2, we expose a state of the art of works related to predictions in data cubes. We introduce and formalize the MCEF in Sect. 3. Section 4 details the formalization of NAP-HC. In Sect. 5, we carry out experiments investigating the effectiveness and the efficiency of our proposals. Finally, Sect. 6 summarizes our contributions and addresses future research directions.

2 Related Work

In recent years, several studies have been addressing the issue of extending OLAP to advanced analysis capacities. They were driven under different motivations

Table 1. Proposals addressing prediction in data cubes

Proposal	Goal	Optimization	Reduction	Measures	Values
Sarawagi *et al.* [3]	Exploration	+	−	−	−
Palpanas *et al.* [8]	Compression	−	+	−	−
Chen *et al.* [9]	Prediction	+	−	+	−
Cuzzocrea [10]	Query approximation	+	+	+	+
Chen *et al.* [11]	Compression	−	+	+	−
Bodin-Niemczuk *et al.* [6]	Prediction	+	−	−	−
Cuzzocrea and Saccà [12]	Privacy preserving	+	−	−	+
Agarwal and Chen [7]	Prediction	+	−	+	−
Our approach	Prediction	−	+	+	+

e.g. discovery-driven cube exploration [3], association rules mining [13], cube compression [11]. Thus, they are based on various concepts and methodologies. In this section, we focus on those having a close linkage with prediction in data warehouses.

Table 1 summarizes the proposals attempting to extend OLAP to prediction. These proposals are detailed according to five main criteria: (1) What is the overall goal of the proposal? (2) Does the proposal include an algorithmic optimization? (3) Does it use a reduction technique? (4) Does it introduce new classes of measures? And (5) Does it provide explicit predicted values of empty measures? We note (+) if the proposal fulfils the criteria, and (−) if in the opposite situations.

Sarawagi *et al.* proposed to assist data warehouse users when exploring data by detecting exceptions [3]. Their approach is based on a log-linear model. Palpanas *et al.* used the principle of information entropy to build a probabilistic model capable of detecting measure deviations [8]. To compress data cubes, Chen *et al.* introduced the concept of *Prediction cubes*, where the score or the probabilities of measures are fetched beside their original values [9]. Prediction Cubes are exploited to build prediction models, which predict low-level measures from high-level pre-calculated aggregates. In [10], Cuzzocrea propose a statistical framework that provides probabilistic bounds on approximate answers. This framework's main goal consists at supporting OLAP applications in overcoming queries' answering, which are considered among the main bottleneck for of OLAP applications. More specifically, it aims at enhancing the accuracy of the approximate answers. To do so, the framework reposes on a sampling technique, which ensures the quality of the approximate answers and generates the probabilistic guarantees on their approximation's degree. On the other hand, to ensure the scalability of the proposal, the author extends it with a previously proposed

data cube dimensions reduction technique [14], based on the Karhunen-Loeve transform [15]. In [11], Chen *et al.* proposed a new type of multidimensional structures called *Regression Cubes*, which contain compressible measures. Regression cube cells indicate measure variations and tendency. Cuzzocrea and Saccà address the computation of privacy preserving OLAP aggregations [12]. Their framework reposes on sampling-based data cube compression. Its strength consists in the fact that the generated privacy preserving aggregates stills allow the evaluation of approximate answers. Agarwal and Chen introduced a new data cube class called *Latent-Variable Cube*, built over a statistical model [7]. It enables the computation of aggregate functions, such as mean and variance over latent variables. Bodin-Niemczuk *et al.* propose to equip OLAP with a regression tree to predict measures of forthcoming facts [6].

Most of the cited proposals recognize that the combination of the important dimensionality and huge volumes of data cubes represent a serious challenge for most of the approaches trying to apply a data mining technique on OLAP cubes. To face up this challenge, some proposals consider a preprocessing stage to reduce the dimensionality effect on algorithm's performance [8,10,11], while some others rather rely on heuristics to optimize implemented algorithms [3,7,9,12]. In our case, we include a PCA-based preprocessing stage in our prediction proposal, which reduces the data cubes dimensionality and generates concentrated, information preserving training sets for the prediction stage.

We notice that all the approaches having different goals than measure's prediction do not provide explicit values for measures. While [3,6,8,10,12] provide approximations of non-existent measures, [7,9,11] introduce new classes of data cubes within new measures generated over the existing ones. Nevertheless, this is totally justified since most of the cited proposals largely meet their main objectives. Among them all, only Bodin-Niemczuk *et al.*'s proposal shares the particular goal of non-existent measure prediction with us [6]. However, the output of this approach is a set of discretized values of the targeted measures. Subsequently, the relevance of the results is strongly depending of the nature and the range of the produced intervals. This issue may not satisfy the analyst who aims an explicit precise decision. The predictive model that we introduce in this paper provides the decision maker with explicit predicted values of non-existent measures, which do not require any further processing.

3 Multi-Perspectives Cube Exploration Framework

3.1 Motivations

Following the success of data warehouse technology, OLAP tools, which are limited to exploratory tasks, are no longer sufficient to meet the increasing needs of OLAP users. Thus, several approaches have been trying to extend OLAP to new abilities by coupling it with data mining techniques to deal with different issues, e.g. cube exploration [3], association rules mining [13], non-existent measures prediction [16,17]. However, most of these proposals consist at specialized solutions, which are tightly related to their particular goals. Thus, even if most

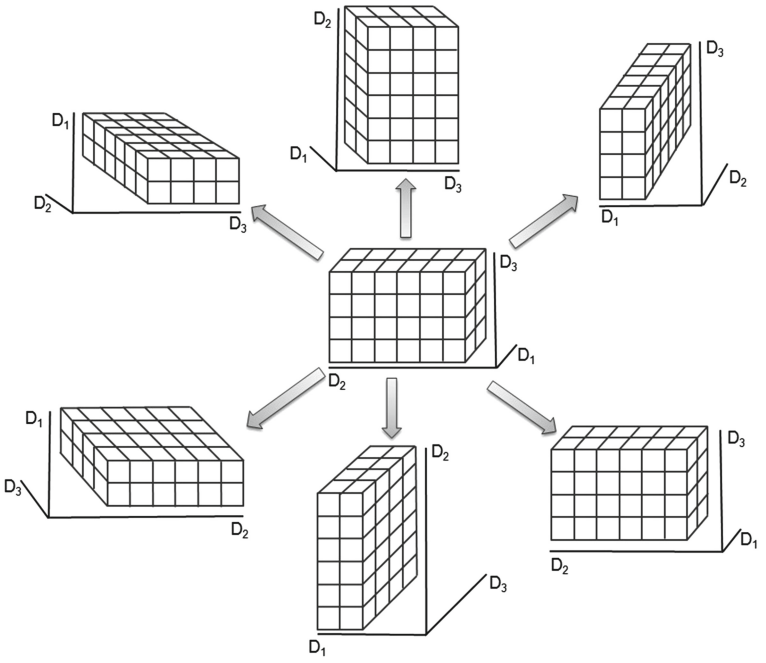

Fig. 1. Possible views of a 3 dimensions cube

of them share the same general motivation of applying data mining algorithms on OLAP cubes, they cannot exploit the already designed formalism of each other's.

We believe that a uniform standardized framework that assists the extension classical data mining algorithms to OLAP cubes' context could turn out to be very useful. Firstly, it offers a uniform ready-to-deploy formalism for the forthcoming proposals aiming to extend OLAP cubes context with mining algorithms. Secondly, and most importantly, it opens the doors for interoperability between the different proposals, since they will be based on the same formalism and handling the same components type. Doing do, the outputs of one proposal could be exploited as the inputs of another. For example, an analysis could start with cube exploration [3], passes through association rules extraction [13] and ends by non-existent measures prediction [16]. Doing so, the analysis would end-up with more efficient reportings. Furthermore, implementing this platform opens the doors for producing software packages that include multiple cube mining algorithms, similarly to *Weka* and *Tanagra* packages [18], which are dedicated to the bi-dimensional context.

On the other hand, the application of cube mining techniques is generally preceded by an in-depth analysis step, which consists of a vertical cube exploration that ends by selecting the most suitable hierarchical levels for the analysis. However, even if it is commonly ignored, horizontal cube exploration, which

consists at the selection of the most convenient dimensions' distribution over cube axes for the analysis, have to be considered.

Actually, each dimensions' distribution across the cube axes generates a different cube view, i.e. data presentation. As illustrated in Fig. 1, multiple data presentations could be obtained from a single three-dimensional data cube, following the dimensions' distribution across its axes. Cuzzocrea and Mansmann state that the efficiency of data representation has an important impact on the data exploration and visualization [19].

Actually, the dimensions' distribution defines the way data is delivered to the data mining algorithm. Therefore, it has a great impact on multiple data mining techniques, especially, the ones that are sensitive to the way data is delivered to them. For these techniques, considering a single dimensions' distribution may promote some dimensions at the expense of others, which causes the loss of the relevant patterns that could be generated over the unexplored views. Nevertheless, most of the researches ignore horizontal cube exploration and limit their analysis to a single dimensions' distribution, usually, implicitly, selected following the user's preferences.

In [20], Ramakrishnan and Chen highlight that mining large datasets requires a principled way to explore the large space of possibilities and alternatives. We further claim that; in order to obtain representative results from rich versatile structures such as data cubes, horizontal cube exploration should be addressed and reinforced.

To concretize these considerations, we design a Multi-perspectives Cube Exploration Framework (MCEF). It is a generalized framework that assists the application of classical data mining algorithm on OLAP cubes, while supporting both horizontal and vertical cube explorations, designed to meet the following goals:

1. Supporting the application of classical data mining algorithms on OLAP cubes;
2. Considering both horizontal and vertical data cubes exploration approaches;
3. Ensuring equitable contributions of the dimensions to the analysis;
4. Preserving the semantics linking members to their respective dimensions and to other dimensions' members;
5. Covering all the possible measures' variations in terms of dimensions' distributions;
6. Enabling BI analyst to define customized analysis contexts.

3.2 Multi-Perspectives Cube Exploration Framework

In this subsection, we thoroughly describe and elaborate the MCEF formalism.

We start by recalling Ben Messaoud *et al.* data cube definitions, which we intend to reuse [13]. Afterwards, we introduce the new definitions required to develop the MCEF formalism.

We start by recalling [13] data cube definitions, which we reuse in MCEF formalization. Let $\mathcal{C}$ be a data cube having the following properties:

- $\mathcal{C}$ has a nonempty set of d dimensions $\mathcal{D} = \{D_i\}_{(1 \leq i \leq d)}$;
- $\mathcal{C}$ contains a nonempty set of m measures $\mathcal{M} = \{M_q\}_{(1 \leq q \leq m)}$;
- $\mathcal{H}_i$ is the set of hierarchical levels of the dimension D_i. $H_j^i \in \mathcal{H}_i$ is the j^{th} hierarchical level of D_i. In Fig. 2, H_1^2 of D_2 is `Product_name`.
- $\mathcal{A}_{ij}$ is the set of members of the hierarchical level H_j^i; $\theta_t^{ij} \in \mathcal{A}_{ij}$ is the t^{th} member of the j^{th} hierarchical level of the dimension D_i. In Fig. 2, θ_5^{22} is `iPod`.

Definition 1 ***Inter-dimensional predicate.*** *Let $\mathcal{D}_a \in \mathcal{D}$ be a nonempty set of p dimensions $\{D_1, ..., D_p\}_{(1 \leq p \leq d)}$ from the data cube $\mathcal{C}$. An inter-dimensional-predicate defines a conjunction of non-repetitive members, i.e., each dimension has a distinct member in the expression. $\Theta^a = (\theta_t^{mi} \wedge ... \wedge \theta_s^{nj})$ is called an inter-dimensional predicate in $\mathcal{D}_a$ if θ_t^{mi} is the t^{th} member of the i^{th} hierarchical level of the dimension D_m and θ_s^{nj} is the s^{th} member of the j^{th} hierarchical level of the dimension D_n, and $\{D_m, D_n\} \in \mathcal{D}_a$.*

In Fig. 2, let $\mathcal{D}_a = \{D_1, D_2\}$ be a set of dimensions of $\mathcal{C}$, a random inter-dimensional predicate Θ^a can of be of the form: $(\langle \theta_1^{11} \in \mathcal{A}_{11} \rangle \wedge \langle \theta_5^{22} \in \mathcal{A}_{22} \rangle)$, e.g. $(\langle \texttt{quarter1} \rangle \wedge \langle \texttt{iPod} \rangle)$.

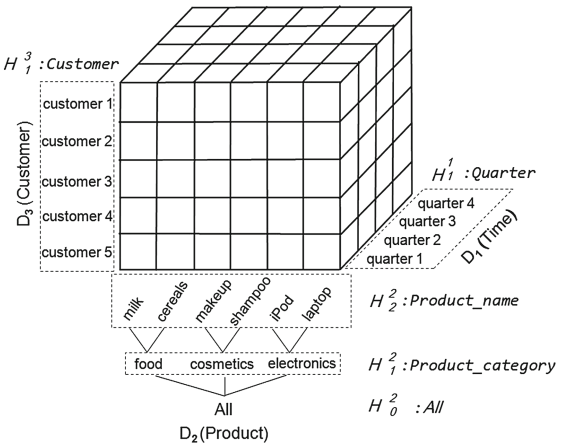

Fig. 2. Example of a data cube

Hereafter, we define the new concepts required to formalize our proposal.

Definition 2 ***Inter-dimensional hierarchical predicate.*** *Let $\mathcal{D}_a \in \mathcal{D}$ be a nonempty set of p dimensions $\{D_1, ..., D_p\}_{(1 \leq p \leq d)}$ from the data cube $\mathcal{C}$. An inter-dimensional hierarchical predicate defines a conjunction of distinct hierarchical levels of non-repetitive dimensions. $\Omega^a = (H_m^s \wedge ... \wedge H_n^t)$ is called an inter-dimensional hierarchical predicate of $\mathcal{D}_a$ if H_m^s is the m^{th} hierarchical level in D_s, H_n^t is the n^{th} hierarchical level in D_t and $D_s \neq D_t$.*

In Fig. 2, let $\mathcal{D}_a = \{D_1, D_2\}$ be a set of dimensions of the data cube $\mathcal{C}$. $\Omega_i^a = (\langle H_1^1 \in \mathcal{H}_1 \rangle \wedge \langle H_2^2 \in \mathcal{H}_2 \rangle)$, which is, $(\langle \texttt{Quarter} \rangle \wedge \langle \texttt{Product\_name} \rangle)$ is a random inter-dimensional hierarchical predicate of $\mathcal{D}_a$.

In the sequel of this paper, let $\mathcal{D}_c = \{D_1, \ldots, D_c\}_{(0 \leq c \leq d-2)}$, $\mathcal{D}_v = \{D_1, \ldots, D_v\}_{(0 \leq v \leq d-2)}$ and $\mathcal{D}_r = \{D_1, \ldots, D_r\}_{(0 \leq r \leq d-2)}$ be three non-empty sets of c, v and r distinct dimensions, respectively; with $c + v + r \leq d$ and Ω^c, Ω^v, Ω^r be three inter-dimensional hierarchical predicates of $\mathcal{D}_c$, $\mathcal{D}_v$ and $\mathcal{D}_r$, respectively.

Let $\Theta^c, \Theta^v, \Theta^r$ be three inter-dimensional predicates in $\mathcal{D}_c$, $\mathcal{D}_v$, $\mathcal{D}_r$, respectively, and let $\Omega^c, \Omega^v, \Omega^r$ be three inter-dimensional hierarchical predicates of $\mathcal{D}_c$, $\mathcal{D}_v$, $\mathcal{D}_r$, respectively.

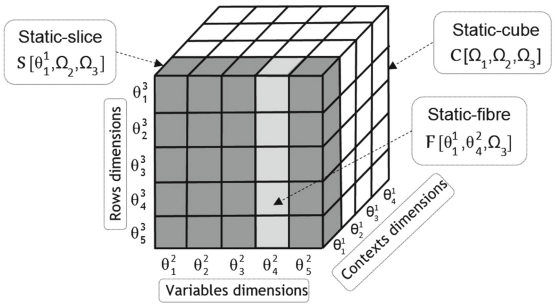

Fig. 3. Static-cube, static-slice and static-fibre

Definition 3 ***Static-cube.*** *We denote by $C[\Omega^c, \Omega^v, \Omega^r]$ a static-cube of a data cube $\mathcal{C}$. It is the fixed distribution of the cells obtained by the application of the OLAP Dice operator on $\mathcal{C}$, following, Ω^c, Ω^v and Ω^r . C is identifiable by the distribution of $\Omega^c, \Omega^v, \Omega^r$, across $\mathcal{C}$ axes.*

The dimensions of C are distributed over three classes; *Contexts* dimensions $\mathcal{D}_c$, *Variables* dimensions $\mathcal{D}_v$ and *Rows* dimensions $\mathcal{D}_r$, which we refer to as MCEF dimensions classes. Each of these classes is designed to ensure a particular role:

- **Contexts dimensions :** The set of attributes generated over these dimensions combination serves in identifying the different analysis subcontexts
- **Variables dimensions :** The set of attributes generated over these dimensions combination is considered as a set of variables.
- **Rows dimensions :** The set of attributes generated over these dimensions combination is considered as a set of observations.

The main goal of the static-cube concept is to depict all the possible dimensions' distributions across the cube axes. An illustrative example is shown in Fig. 3. The latter represents the static-cube $C[\Omega^1, \Omega^2, \Omega^3]$, with $\mathcal{D}_c = \{D_1\}$, $\mathcal{D}_v = \{D_2\}$, $\mathcal{D}_r = \{D_3\}$ as the sets of Contexts, Variables and Rows dimensions, respectively.

Following the BI analyst preference, an analysis could either consider the most relevant static-cube to the analysis or involve the entire set of static-cubes.

On the other hand, a single three-dimensional OLAP cube's slice can generate two distinct bi-dimensional tables, with one representing the transpose of the other. This might cause the confusion of the data mining technique and lead to inconsistent results. To solve this issue, in what follows, we introduce the concept of static-slice, which enables the distinction between the different bi-dimensional tables that can be generated over a single OLAP slice.

Definition 4 ***Static-slice.*** *We denote by* $S[\Theta^c, \Omega^v, \Omega^r]$ *a static-slice of a static-cube* C*. It is the fixed distribution of the cells obtained by the application of the OLAP Slice operator on* $\mathcal{C}$*, following* Θ^c, Ω^v *and* Ω^r*.* S *have the same MCEF dimensions distribution of* C*.*

The concept of *static-slice* is designed to enable browsing static-cubes in a principled way, following the different MCEF classes. For instance, the dark grey coloured cells in Fig. 3 represent the static-slice $S[\Theta_1^1, \Omega^2, \Omega^3]$.

Definition 5 ***Static-fibre.*** *We denote by* $F[\Theta^c, \Theta^v, \Omega^r]$ *a static-fibre of a static-slice* S*. It is the fixed distribution of the cells obtained by the application of the OLAP Dice operator on* $\mathcal{C}$*, following* Θ^c, Θ^v *and* Ω^r*.* F *have the same MCEF dimensions distribution of* S*.*

The concept of *static-fibre* is designed to enable browsing static-slices in a principled way, following the different MCEF classes. As instance, the light grey coloured cells in Fig. 3 represent the static-fibre $F[\Theta_1^1, \Theta_4^2, \Omega^3]$.

Data mining could be classified into two distinct categories. The first one concerns the data mining techniques that are not sensitive to the way data is provided to them. Therefore, they would generate the same mining outcome with the different static-cube. As for the second category, it concerns the data mining that are sensitive to the way data is provided to them, which makes each static-cube a unique dataset. For this type of category, the most optimal scenario that ensures equitability between dimensions consists in involving all the potential static-cubes in the analysis. Then, following the analysis aims, the BI analyst can either combine the obtained results or consider them separately. Still, this solution is very expensive and may turn to be non-effective, especially if in the case of online deployment. The other alternative consists in limiting the analysis to a the most relevant the static-cubes.

4 Neural Approach for Prediction over High-Dimensional Cubes

4.1 Overview

As far as we know, despite their proven performances, Neural Networks (NN)s are not yet exploited in OLAP cubes' context. This is due to multiple factors. First,

NNs generalization capabilities become limited when handling high-dimensional datasets. Second, the computational requirements of NNs increase drastically with the increase of inputs' number, which slows down the learning rates [21]. Third, highly correlated data may corrupt the training phase of NNs and degrade their generalization capability [22].

On the other hand, Modular Neural Networks (MNN)s represent a well-established technique in the field of machine learning. They are generally composed by set of (NN)s called *modules* and a *combiner* system [23]. They are based on the "divide and conquer" principle. They undertake a complex problem, divide it into smaller tasks and distribute them over the modules. Modules can be trained independently or sequentially targeting the same task. While, the combiner system processes their outputs to generate a conclusive analysis result of the entire system.

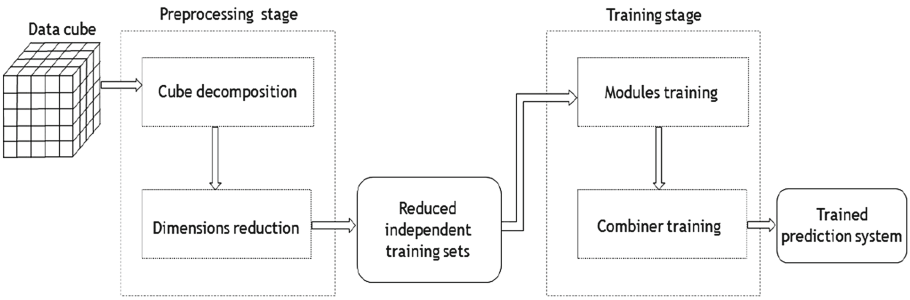

Fig. 4. Overview of NAP-HC architecture

Multiple researches claim that MNNs overcome multiple limitations of single NNs [23–25]. Melin and Castillo state that MNNs are very effective to overcome the problems defined over high-dimensional space and having high complexity [25]. Happel and Murre [26] state that MNNs enable the application of NNs on large-scale data [26]. Gallinari claim that MNNs reduce the model complexity, provide robustness and enable data sources fusion [24]. Sharkey sheds the light on the fact that decomposing a large complex task into modular components makes the system easier to understand and to modify [23]. These factors, make MNNs highly promising candidates to overcome the limitations of NNs with multidimensional large structures, such as data cubes.

Despite the fact that MNNs might resolve multiple problems related to the application on NNs on OLAP cubes, the high dimensionality and the correlated measures still represent thriving challenges that could deteriorate the training process quality. Recently, some studies have been interested in Principal Component Analysis (PCA) to reduce the dimensionality of prediction models inputs [27,28]. The PCA is an exploratory statistical procedure, which aims at transforming the original correlated variables into a smaller set of uncorrelated ones, called *principal components* [29]. Its key idea is to project the initial data

on a new orthogonal subspace to find the linear combinations that define new summarizing variables, which concentrate the largest possible variance of the original ones.

Therefore, we find that the PCA represents a good solution to assist in solving the limitations caused by the important number of inputs and the measures correlation. We intend to follow this trail as a backstage preprocessing step that would ensure the generation of new reduced training sets that preserves the measure variability.

On the other hand, OLAP measures have multiple linear variations following the different axes of the data cube. Considering a single measure variation may make the prediction process fall into the pitfall of promoting a particular set of dimensions at the expense of the other ones. This could generate a prediction model that may not reflect the complete multidimensional context.

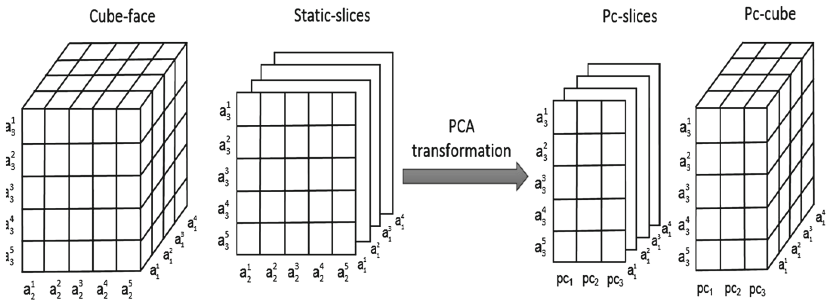

Fig. 5. Overview of the preprocessing stage of a single static-cube

To tackle this issue, we introduce the Neural Approach for Prediction over High-dimensional Cubes (NAP-HC). The NAP-HC's main goal is to overcome the challenges of the application of Neural Networks (NN)s within the context of OLAP cubes. To do so, we design it over the MCEF, which is dedicated to assist the application of classical data mining techniques on data cubes.

The NAP-HC combines the modular aspects of MCEF and MNN to provide a prediction solution that enables the application of NNs on a data cube, while covering all its data presentations. As shown in Fig. 4, NAP-HC is carried out in two major stages. The first one is a preprocessing stage, which is divided, in its turn, into two steps. The first one consists in extracting the MCEF substructures. As for the second step, it consists at applying the PCA on the MCEF substructures to transform their correlated attributes into reduced sets of decorrelated principal components. The second stage is a prediction one, which considers each reduced dataset, obtained over the first stage, as the learning set of an independent NNs module. Then, it trains a NNs combiner system, which considers the outputs of each module as its own inputs, to come out with a unique predicted measure of each targeted cell.

To sum up, the NAP-HC overarching goals are as follows:

1. Generating reduced, information preserving training sets from the original data cube;
2. Adapting NNs to the multidimensional structure of data cubes;
3. Predicting explicit values of non-existent measures;
4. Assessing predicted measures with quality indicators.

4.2 Preprocessing Stage

The main goal of this stage is to generate concentrated, independent, information-preserving data subsets, which can be exploited later as the training sets of the independent modules. As illustrated in Fig. 4, it is based on two main steps. The first one consists in decomposing the complex multidimensional data cube domain into a set of linear sub-domains, defined by the MCEF substructures. As for the second step, it is a dimensions reduction step, which consists in applying PCA on the obtained MCEF substructures.

The NAP-HC exploits MCEF as a modular principled cube explorations technique. First, the dimensions are distributed over three mutually exclusive sets, following the analysis's goals. Each of these sets plays a different role as one of MCEF dimensions classes. Then, all the possible MCEF classes' combinations are considered to define and extract potential static-cubes, which consist of distinct data presentation following the dimensions' distribution over its axes.

The second step of the preprocessing stage is illustrated in Fig. 5. It is a dimensions reduction and data transformation step. Its main goal is to reduce the attributes of each static-cube Variables dimensions and to transform its members into a reduced, concentrated set of principal components. It starts by extracting sequentially each static-cube static-slices by sequentially applying MDX queries.

Static-slices are not dynamic such as classic OLAP slices, so they can be directly considered as disjunctive tables with Variables dimensions as attributes' dimension and Rows dimensions as instances' dimension. The PCA is then applied sequentially on the static-slices, to generate a new type of slices, which we refer to as *pc-slice*. Each pc-slice shares the same Rows and Contexts dimensions' sets with its associated static-slice. However, its Variables dimensions are replaced with a new dimension, referred to as *pc-dimension*. It has the set of retained principal components as attributes. As for the obtained factorial coordinates, they are stored as the values of the pc-slices.

The set of pc-slices generated over the static-slices of the same static-cube, are gathered as a new multidimensional structure that we call *pc-cube*. Actually, a pc-cube is a static structure associated to one particular static-cube. Unlike regular OLAP cubes, pc-cubes are not dynamic and do not support OLAP operations. Their role consists in providing an organized storage solution for the obtained factorial coordinates to track their membership to the original cube cells. They are trackable from the data cube through a new type of measure, which we call *pc-measure*. It is an indexation measure that links each cell to its adequate

factorial coordinates in the pc-cubes. It is embedded within the original cells' measures after the application of PCA on each static-slice.

The usages of these new PCA oriented concepts provide an efficient storage solution. It enables discarding each static-cube from the main memory, as soon as its associated pc-cube is generated. The storage of pc-cubes is less expensive than the storage of static-cubes, since they represent their reduced version. Doing so, the preprocessing stage provides reduced, decorrelated predictors that require a minimum storage cost.

Algorithm 1. Static-cube generation and reduction

Input: Ω^c, Ω^v, Ω^r
Output: The pc-cube Pcc
$C \leftarrow generate\_cube\text{-}face(\Omega^c, \Omega^v, \Omega^r)$;
$Pcc \leftarrow \emptyset$;
$i = 0$;
foreach *nonempty* Θ_i^c *of* Ω^v **do**
  $S_i \leftarrow generate\_slice(\Theta_i^c, \Omega^v, \Omega^r)$;
  $Pcs_i \leftarrow PCA(S_i)$;
  $Pcc \leftarrow Pcc + Pcs_i$;
  $i \leftarrow i + 1$;
$return(Pcc)$;

The static-cube generation and reduction is provided in Algorithm 1. It requires three inter-dimensional hierarchical predicates Ω^c, Ω^v, Ω^r translating the three MCEF classes as inputs and processes as follows:

- The static-cube C is generated according to the inter-dimensional hierarchical predicates Ω^c, Ω^v and Ω^r.
- Each inter-dimensional hierarchical predicate $\Omega_i^c \subset \Omega^c$ is instantiated to the next nonempty inter-dimensional predicate Θ_i^c.
- The static-slice $S[\Theta_i^c, \Omega^v, \Omega^r]$; S_i is then generated.
- PCA is applied on S_i and the obtained factorial coordinates are stored into the pc-slice pcs_i.
- pcs_i is added to the pc-cube Pcc.
- the output of this algorithm is a fully indexed pc-cube, representing the reduced version of the treated static-cube.

We admit that, similarly to of the conventional OLAP preprocessing phases, this preprocessing stage is a time-consuming one. Therefore, we believe that it should be executed in backstage on a regular basis by the end of each periodic data loading of the data warehouse.

4.3 Prediction Stage

The main goal of this stage is to learn from the outputs of the preprocessing stage, which are the pc-cubes, to come out with unique explicit value for each

targeted measure. To the best of our knowledge, PCA has not been yet exploited with MNNs by any previous work.

By virtue of their operation simplicity, their excellent generalization capacity and their ability to approximate any universal function, Multilayer Perceptrons (MLP)s represent one of the popular NNs [30]. Thus, for all the sub-networks that compose our system, we adopt the MLPs architecture. In addition, several theoretical and empirical studies show that a single hidden layer is sufficient to achieve a satisfactory approximation of any nonlinear function [30]. Thus, we associate a three layers MLPs architecture, including a single hidden layer for each sub-network. We also use the gradient back-propagation algorithm [31], that has proven its usefulness in several applications [30,32]. We associate it with the conjugate gradient learning method and the sigmoid activation function.

The prediction system is composed of an interconnection of a set of module-networks and a single combiner-network. The number of module-networks is equal to that of pc-cubes obtained of the preprocessing stage. Each module-network is trained independently. It considers the factorial coordinates as inputs and targets the measure' values. In addition, each module-network has three layers:

1. An input layer, which contains a number of neurons equal to that of of the principal components of the pc-cube associated to the module;
2. A hidden layer, which contains an empirically selected number of neurons;
3. An output layer that contains a single output.

As for the combiner-network, it follows the same architecture as the modules except that its input layer neurons' number is equal to the number of module-networks. It brings together all the module-networks output as its own inputs. Thus, the input vector of the combiner-network is obtained by propagating the factorial coordinates associated to the same cell into all the module-networks. The measure's value of this cell represents the output of the combiner-network. This process is repeated until the combiner-network reaches the convergence status at its turn.

The pseudo-code of the training algorithm is described in Algorithm 2. As inputs, it requires the data cube $\mathcal{C}$, the set of the obtained over the preprocessing stage pc-cubes $\{Pcc\}$ and the Root Mean Squared Error(RMSE) minimum value $RMSE\text{-}min$. For each module-network, NAP-HC starts by selecting a random set of cells as training set from the data cube, $A[]$, and the pc-cube, Pcc, associated to the treated module. For each training cell, the algorithm accesses the pc-measure, pc, and fetches it to get its appropriate factorial coordinates vector, $fc[]$, from the pc-cube. $fc[]$ is then injected into the input layer of the module-network, while targeting the initial measure's value. This process is repeated for each module-network until there are no more training instances or until the RMSE reaches $RMSE\text{-}min$ value. After performing the sequential independent training of all the module-networks, the combiner-network becomes ready to be initialized and trained.

We stress that our approach is not a cube completion technique, i.e. it is not designed to fill all empty measures of a data cube. However, the main goal of

Algorithm 2. Training the prediction system

Input: $\mathcal{C}, \{Pcc\}, RMSE - min$
Output: Trained prediction system

foreach *module* **do**
  $Pcc \leftarrow select\_Pcc(\{Pcc\})$;
  $module \leftarrow initialize(module)$;
  $A[] \leftarrow generate\_random - cells(\mathcal{C})$;
  while $((A[] \neq \emptyset)$ ***and*** $(RMSE(module) < RMSE - min))$ **do**
    $m \leftarrow get\_measure(\mathcal{C}, A[])$;
    $pc \leftarrow get\_pc - measure(Pcc, A[])$;
    $fc[] \leftarrow get\_factorial - coordiantes(Pcc, pc)$;
    $propagate(module, fc, m)$;
    $back - propagate(module, fc, m)$;
    $adjust(module)$;
$combiner \leftarrow initialize(combiner)$;
$A[] \leftarrow generate\_random - cells(\mathcal{C})$;
while $((RMSE(combiner) < RMSE - min)$ ***and*** $(A[] \neq \emptyset))$ **do**
  $combiner - input[] \leftarrow \emptyset$;
  foreach *module* **do**
    $m \leftarrow get\_measure(\mathcal{C}, A[])$;
    $pc \leftarrow get\_pc - measure(Pcc, A[])$;
    $fc[] \leftarrow get\_factorial - coordiantes(Pcc, pc)$;
    $combiner - input[] \leftarrow combiner - input[] + propagate(module, fc, m)$;
  $propagate(combiner, combiner - input[], m)$;
  $back - propagate(combiner, combiner - input[], m)$;
  $adjust(combiner)$;
$return(Trained\ prediction\ system)$;

Table 2. Static-cubes description

Static-cube	Contexts	Variables	Rows	# retained components
C_1	Location	Education	Origin	3
C_2	Location	Origin	Education	4
C_3	Education	Location	Origin	10
C_4	Education	Origin	Location	4
C_5	Origin	Location	Education	12
C_6	Origin	Education	Location	4

NAP-HC is to promptly come-out with a predicted value of any empty measure upon the request of the BI analyst.

5 Experimentation

We implemented an experimental prototype of our approach, in Java, on a running on Microsoft Windows 7 with Intel Core 2 Duo, 2 GHz of CPU processor, 4 GB main memory workstation. We used *Microsoft SQL Server Analysis Services 2008*(SSAS) as an OLAP server. We performed our experiments on the database *American Community Surveys 2000–2003*[1], after adapting it to the OLAP context. It is a real-life database of the U.S.A census that concerns the population samples treated between 2000 and 2003.

5.1 Analysis Context

We consider a four dimensions data cube; `Location`, `Origin`, `Education` and `Time`, with 3.8 million facts. The `Location` dimension contains the geographic data of the census. `Origin` dimension contains information about the racial structure of the U.S.A population. `Education` dimension contains information on the education levels reached by the subjects of the census. We aim at predicting the number of people of a certain race, according to their cities and their levels of education in 2003.

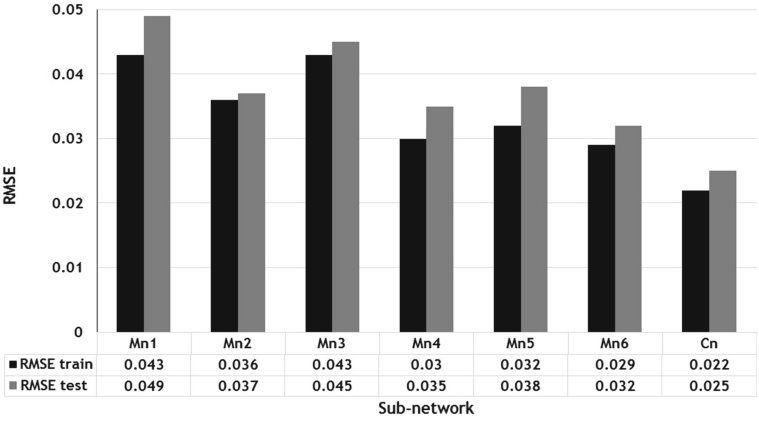

Fig. 6. Prediction quality

To be able to analyze and compare the different static-cubes outcomes, we limited each MCEF class to one dimension `Location`, `Education` and `Origin` and we selected the member `2003` of `Time` dimension. This led to the generation of six static-cubes as summarized by Table 2. We selected the hierarchical levels *`Location`*, *`Education`* and *`Origin`*, respectively. These levels include 51, 14 and 10 members, respectively. We investigated the measure `person-count`.

[1] American Community Surveys is accessible from the official site IPUMS-USA (Integrated Public Use Microdata Series); http://sda.berkeley.edu.

We elaborated a predictive system that faithfully represents our proposed architecture. After the application of the preprocessing stage, we ended up with the 6 pc-cubes from which we retained different numbers of principal components described in Table 2. As for the prediction stage, we have set the number of hidden neurons of each sub-network's hidden layer to the half of its inputs. We used the 10-fold cross-validation technique and the Root Mean Squared Error (RMSE) as a quality indicator. For accuracy reasons, more specifically, to avoid the impact of the random weights initialization of NNs, we ran all the experiments five times and provided the resulting means of RMSE and execution time in this section.

5.2 Prediction Quality

Figure 6 illustrates the prediction performances for all the sub-networks that compose our predictive system. We notice that RMSE values vary remarkably from of a module-network to another one. This is justified by the particularity of the different data structures of each pc-cube. We find that the two module-networks that provide the largest RMSEs, and thus the worst prediction quality, are Mn_1 and Mn_3. We note that these two module-networks consider `Origin` as their Rows dimension.

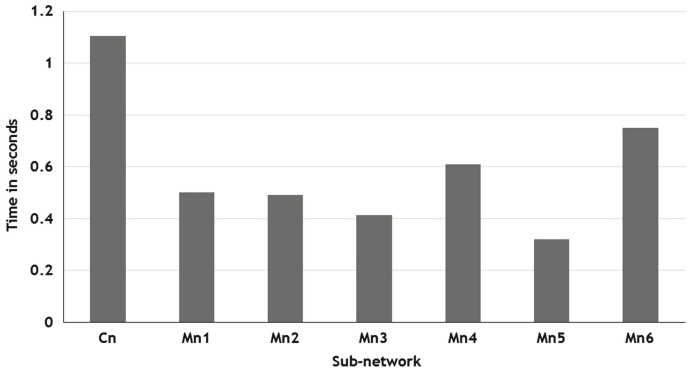

Fig. 7. Training time consumption

We recall that in our proposal, the number of available instances in a training set for a module-network is defined by the number of the Rows dimensions' members. In our case, `Origin` dimension is the poorer dimension in terms of members' number (10 members). Subsequently, the module-networks that consider it as their Rows dimension have the smallest number of training instances. The poor prediction quality can be due to that this number has been not sufficient to ensure the module-network learning. Inversely, we found that Mn_4 and Mn_6, which consider `Location` as their Rows dimension, produce the smallest RMSE values among all module-networks.

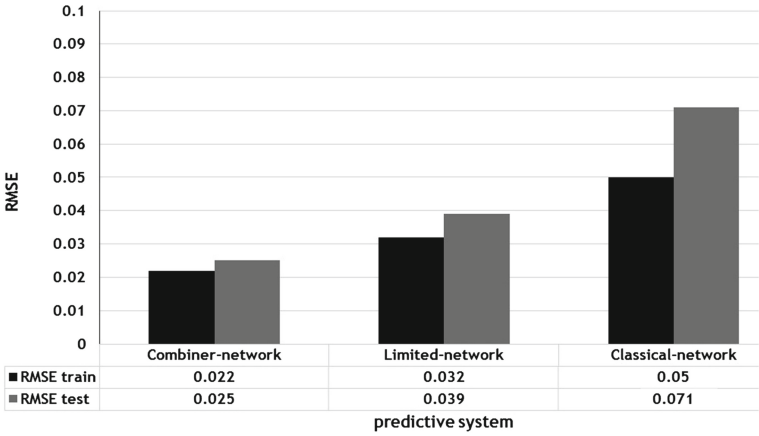

Fig. 8. Performances of the different prediction systems

Interestingly enough, we find that the obtained RMSEs values are generally acceptable. Still, the best prediction performance is achieved by the combiner-network. It surpasses all the module-networks in the training and the test phases. This confirms the efficiency of the modular architecture in generating better prediction by combining the knowledge of all the module-networks. Since, each module-network has become an expert in its particular cube perspective, joining the modules into an ensemble make them compensate each other's limitations through the combiner-network, which combines all static-cubes bi-dimensional knowledge and convert it into a multidimensional one.

As shown on Fig. 7, the training time varies from a sub-network to another one. The most time requiring module-networks are Mn_3 and Mn_5. This is due to the fact that they consider `Location`, which is the richest dimension in terms of members' number, as Variables dimension, which led to retaining a larger number of principal components after PCA application. Consequently, these module-networks have the largest number of input and hidden neurons among all module-networks, what makes them require a larger number of pc-measures accesses to fetch the factorial coordinates. Moreover, their important number of neurons leads to more complex computations, and thus they consume more time to converge.

Furthermore, we find that the combiner-network is the most time consuming among all sub-networks. This is explained by the fact that at each turn of its training phase, it has to access to all the module-networks' principal components to obtain its own input vector.

5.3 Novel Architecture Contributions

In the previous experiment, we found that several module-networks provide very modest performances compared to the other ones. Following these results, a logical question arises: How would the system perform if we eliminate the

module-networks that produce the worst results from the analysis? To answer this question, we designed and trained an additional prediction system, which we call `Limited-network`. The latter is similar to NAP-HC, except that it does not involve Mn_1 nor Mn_3, i.e. the module-networks that performed the least efficient predictions through the previous experiment. Surprisingly enough, as shown in Fig. 8, `Limited-network` provided worst performances than `Combiner-network`. This can be explained by the fact that even if the eliminated module-networks are not useful to perform the prediction task individually, they have a positive role in enhancing the combiner-network knowledge about the multidimensional structure.

To further investigate the contributions of the modular architecture, we trained another system that follows the classical MLPs architecture, which we refer to as `Classical-network`. It considers all the factorial coordinates indexed by the pc-measure of a particular cell as the inputs of one single large MLP. In other words, it merges all the training subsets into a unique large one. As shown in Fig. 8, `Classical-network` provided the worst performances among all the studied architectures. The naïve fusion of the training sets caused the loss of the particularity of the information obtained over each static-cube. Moreover, the large number of inputs limited the MLP generalization abilities. This confirms the positive contributions of MCEF and the efficiency of the combination of MCEF and MNNs.

6 Conclusion and Perspectives

In this paper, we encouraged the exploitation of machine learning techniques to extend OLAP to advanced abilities. The key idea of our proposal is that, these sophisticated techniques can be exploited, in the context of OLAP cubes, even with the challenges raised by their important dimensionality and volumetry. First, we proposed a generalized cube exploration framework, designed to assist the application of machine learning algorithms on OLAP cubes. Then, we exploited it to propose a novel MNNs solution called NAP-HC, which predicts non-existent measures' values over OLAP cubes.

NAP-HC makes use of enhanced procedure to solve the constraints raised by applying sensitive techniques as NNs on a complex data structure like OLAP cubes. It relies on two main stages. A preprocessing one that exploits MCEF to explore the data cube in a principled way, and generate reduced information preserving training subsets by applying PCA on the MCEF substructures. As for the second stage, it exploits the outputs of the first one to train a MNN.

The experimental study proved the efficiency of MCEF in enabling the application of classical data mining algorithms on OLAP cubes. Further, it demonstrated the good prediction performances of NAP-HC and helped to get further insights of the different results obtained over the sub-systems, which form the global model. Furthermore, it compared our proposal against the classical MLPs architecture and confirmed the successful combination between MCEF and MNN.

In future work, we plan to include a framework that explains the reasons of non-existent measures occurrences, similarly to that of [33], which is performed

on classical bi-dimensional data. Cuzzocrea and Mansmann state that multidimensional visualization tools provide more comprehensive analysis for multiple cube mining tasks, including discovering new knowledge from large volumes of multidimensional data [19]. Therefore, we intend to equip the NAP-HC with a visualization tool to assist the prediction phase. We also would like to involve the hierarchical structure of data cubes in our system. This way, we could exploit the different levels of aggregation to predict lower/higher-levels facts. Finally, we believe that modeling a theoretical relation between the reduction and the prediction stages could be very useful to optimize our proposal.

References

1. Inmon, W.H.: Building the Data Warehouse. QED Information Sciences Inc, Wellesley, MA, USA (1992)
2. Goil, S., Choudhary, A.: High performance multidimensional analysis and data mining. In: Proceedings of the High Performance Networking and Computing Conference (SC'1998), Orlando, Florida, US, November 1998
3. Sarawagi, S., Agrawal, R., Megiddo, N.: Discovery-driven exploration of OLAP data cubes. In: Schek, H.-J., Saltor, F., Ramos, I., Alonso, G. (eds.) EDBT 1998. LNCS, vol. 1377, p. 168. Springer, Heidelberg (1998)
4. Ben Messaoud, R., Loudcher-Rabaseda, S.: Olemar: An on-line environment for mining association rules in multidimensional data. In: Advances in Data Warehousing and Mining, vol. 2. Idea Group Publishing (2007)
5. Codd, E.F., Codd, S.B., Salley, C.T.: Providing OLAP (on-line Analytical Processing) to User-analysts: An IT Mandate. Codd and Date, Inc., Manchester (1993)
6. Bodin-Niemczuk, A., Ben Messaoud, R., Rabaséda, S.L., Boussaid, O.: Vers l'intégration de la prédiction dans les cubes OLAP. In: EGC. (2008) 203–204
7. Agarwal, D., Chen, B.C.: Latent OLAP: Data cubes over latent variables. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data. SIGMOD 2011, pp. 877–888. ACM, New York (2011)
8. Palpanas, T., Koudas, N., Mendelzon, A.: Using datacube aggregates for approximate querying and deviation detection. IEEE Trans. Knowl. Data Eng. **17**, 1465–1477 (2005)
9. Chen, B.C., Chen, L., Lin, Y., Ramakrishnan, R.: Prediction Cubes. In: Proceedings of the 31st International Conference on Very large Data Bases. VLDB 2005, pp. 982–993 (2005)
10. Cuzzocrea, A.: Providing probabilistically-bounded approximate answers to non-holistic aggregate range queries in OLAP. In: Proceedings of the 8th ACM International Workshop on Data Warehousing and OLAP. DOLAP 2005, pp. 97–106. ACM, New York (2005)
11. Chen, Y., Dong, G., Han, J., Pei, J., Wah, B.W., Wang, J.: Regression cubes with lossless compression and aggregation. IEEE Trans. Knowl. Data Eng. **18**, 1585–1599 (2006)
12. Cuzzocrea, A., Saccà, D.: Balancing accuracy and privacy of OLAP aggregations on data cubes. In: Proceedings of the ACM 13th International Workshop on Data Warehousing and OLAP. DOLAP 2010, pp. 93–98. ACM, New York (2010)
13. Messaoud, R.B., Rabaséda, S.L., Boussaid, O., Missaoui, R.: Enhanced Mining of Association Rules from Data Cubes. In: Proceedings of the 9^{th} ACM International Workshop on Data Warehousing and OLAP (DOLAP'2006), pp. 11–18. ACM Press, Arlington (November 2006)

14. Cuzzocrea, A.: Overcoming limitations of approximate query answering in OLAP. In: 9th International Database Engineering and Application Symposium. IDEAS 2005, pp. 200–209 (July 2005)
15. Jain, A.K.: Fundamentals of Digital Image Processing. Prentice-Hall Inc, Upper Saddle River, NJ, USA (1989)
16. Abdelbaki, W., Ben Messaoud, R., Ben Yahia, S.: A neural-based approach for extending OLAP to prediction. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2012. LNCS, vol. 7448, pp. 117–129. Springer, Heidelberg (2012)
17. Abdelbaki, W., Ben Yahia, S., Ben Messaoud, R.: NAP-SC: a neural approach for prediction over sparse cubes. In: Zhou, S., Zhang, S., Karypis, G. (eds.) ADMA 2012. LNCS, vol. 7713, pp. 340–352. Springer, Heidelberg (2012)
18. Witten, I., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Elsevier, Morgan Kaufmann, Burlington (2005)
19. Cuzzocrea, A., Mansmann, S.: OLAP visualization: models, issues, and techniques. In: Wang, J. (ed.) Encyclopedia of Data Warehousing and Mining, 2nd edn, pp. 1439–1446. IGI Global, Hershey, PA (2009)
20. Ramakrishnan, R., Chen, B.C.: Exploratory mining in cube space. Data Min. Knowl. Disc. **15**(1), 29–54 (2007)
21. Azam, F.: Biologically inspired modular neural networks. Ph.D. thesis, Virginia Polytechnic Institute and State University, Virginia, USA (2000)
22. Bishop, C.: Neural Networks For Pattern Recognition. Oxford University Press, Oxford (1995)
23. Sharkey, A.J. (ed.): Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems, 1st edn. Springer-Verlag New York Inc, Secaucus, NJ, USA (1999)
24. Gallinari, P.: The Handbook of Brain Theory and Neural Networks. MIT Press, Cambridge, MA, USA (1998)
25. Melin, P., Castillo, O.: Modular neural networks. In: Hybrid Intelligent Systems for Pattern Recognition Using Soft Computing. Studies in Fuzziness and Soft Computing, vol. 172, pp. 109–129. Springer, Heidelberg (2005)
26. Happel, B.L., Murre, J.M.J.: The design and evolution of modular neural network architectures. Neural Netw. **7**, 985–1004 (1994)
27. Tshilidzi, M.: Computational Intelligence for Missing Data Imputation, Estimation, and Management: Knowledge Optimization Techniques. IGI Publishing, Hershey, PA (2009)
28. Wang, Z., Xu, J., Lu, F., Zhang, Y.: Using the method combining PCA with BP neural network to predict water demand for urban development. In: Proceedings of the 2009 Fifth International Conference on Natural Computation. ICNC 2009, pp. 621–625. IEEE Computer Society, Washington (2009)
29. Hotelling, H.: Analysis of a complex of statistical variables into principal components. J. Educ. Psychol. **24**(7), 498–520 (1933)
30. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Netw. **2**(5), 359–366 (1989)
31. Rumelhart, D., McClelland, J.: Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Foundations. Computational Models of Cognition and Perception. MIT Press, Cambridge (1986)
32. Haykin, S.: Neural Networks: a Comprehensive Foundation. Prentice Hall, Prentice Hall International Editions Series (1999)
33. Ben Othman, L., Ben Yahia, S.: Yet another approach for completing missing values. In: Yahia, S.B., Nguifo, E.M., Belohlavek, R. (eds.) CLA 2006. LNCS (LNAI), vol. 4923, pp. 155–169. Springer, Heidelberg (2008)