Wavelets for Querying Multidimensional Datasets

Cyrus Shahabi
University of Southern California, USA

Dimitris Sacharidis
University of Southern California, USA

Mehrdad Jahangiri
University of Southern California, USA

INTRODUCTION

Following the constant technological advancements that provide more processing power and storage capacity, scientific applications have emerged as a new field of interest for the database community. Such applications, termed Online Science Applications (OSA), require continuous interaction with datasets of multidimensional nature, mainly for performing statistical analysis. OSA can seriously benefit from the ongoing research for OLAP systems and the pre-calculation of aggregate functions for multidimensional datasets. One of the tools that we see fit for the task in hand is the wavelet transformation. Due to its inherent multi-resolution properties, wavelets can be utilized to provide progressively approximate and eventually fast exact answers to complex queries in the context of Online Science Applications.

BACKGROUND

OLAP systems emerged from the need to deal efficiently with large multidimensional datasets in support of complex analytical and exploratory queries. Gray et al. (Gray, Bosworth, Layman, & Pirahesh, 1996) demonstrated the fact that analysis of multidimensional data was inadequately supported by traditional relational databases. They proposed a new relational aggregation operator, the Data Cube, which accommodates aggregation of multidimensional data. The relational model, however, is inadequate to describe such data, and an inherent multidimensional approach using sparse arrays was suggested in Zhao, Deshpande & Naughton (1997) to compute the data cube. Since the main use of a data cube is to support aggregate queries over ranges on the domains of the dimensions, a large amount of work has been focused on providing faster answers to such queries at the expense of higher update and maintenance cost. *Pre-aggregation* is the key term here, as it resulted in performance benefits. Ho et al. (1997) proposed a data cube (Prefix Sum) in which each cell stored the summation of the values in all previous cells, so that it can answer range-aggregate queries in constant time. The maintenance cost of this technique, however, can be as large as the size of the cube. A number of following publications focused on balancing the trade-off between pre-aggregation benefits and maintenance costs.

It is not until recent years that the Wavelet Transformation was proposed as a means to do pre-aggregation on a multidimensional dataset. However, most of these approaches share the disadvantage of providing only approximate answers by compressing the data. Vitter, Wang, & Iyer have used the wavelet transformation to compress a pre-processed version of the data cube (1998) or the original data cube (Vitter & Wang, 1999), constructing Compact Data Cubes. Lemire (2002) transforms a pre-aggregated version of the data cube to support progressive answering, whereas in Wu, Agrawal & Abbadi (2000) and Chakrabarti, Garofalakis, Rastogi, & Shim (2000) the data cube is directly transformed and compressed into the wavelet domain, in a way similar to image compression.

A totally different perspective in using wavelets for scientific queries is proposed in Schmidt & Shahabi (2002). Here, the answer to queries posed in scientific applications is represented as the dot-product of a query vector with a data vector. It has been shown (Schmidt & Shahabi, 2002) that for a particular class of queries, wavelets can compress the query vector making fast progressive evaluation of these queries a reality. This technique, as it based on query compression and not data, can accommodate exact, approximate or progressive query evaluation.

MAIN THRUST

What is the Wavelet Transformation?

We will start our discussion by attempting to provide a crude definition of the wavelet transformation, in particular the Discrete Wavelet Transformation (DWT). As the name suggests, it is a transformation of some signal, not too different from other well-known transformations such as Fourier, Laplace, and etcetera. In the context of database applications, the signal is, in general, a multivariate discrete signal that represents a dataset. As a transformation DWT is, essentially, another way to view a signal. The expectance, of course, is that such a view will be more useful and provide more information to the applications in hand.

DWT is lossless, or an orthonormal transformation in signal processing terms, as is the case with the most common transformations. This implies that its effects can be reversed and thus the original signal can be reconstructed in its entirety; a highly desirable property. DWT achieves (lossless) compression by separating the "smooth" part of a signal from the "rough" and iterating on the "smooth" part to further analyze the signal. This is true, provided that the signal is relative smooth, which is the case with real-life datasets and especially with query signals, as we will see.

We can now give the crude definition we promised at the beginning. The Discrete Wavelet Transformation is a lossless transformation that provides a multi-resolution view of the "smooth" and "rough" parts of a signal.

An Example with Haar Wavelets

Haar wavelets are the simplest and were the first to be discovered. The "smooth" version of the signal is produced by pairwise averaging, whereas the "rough" version is produced by pairwise differencing. This is why the Haar wavelet coefficients are called *averages* and *differences* or *details*.

Using signal processing terminology, the "smooth" version of the signal is produced by a *low-pass* filter, which filters out the rough elements. On the other hand, the "rough" version of the signal is produced by a *high-pass* filter, which filters out the smooth elements. Together, these filters are called a *filterbank*, and they produce the smooth and rough views of the signal. DWT is performed by chaining a filterbank on the output of the low pass filter; doing so iteratively leads to the multiresolution view of the signal. A digital filter is simply comprised by a set of coefficients that multiply the input to produce the output. As an example the low-pass Haar filter is comprised by the coefficients $\{1/\sqrt{2}, 1/\sqrt{2}\}$ which multiply input $\{a,b\}$ to produce output $(a+b)/\sqrt{2}$. Similarly, the high-pass filters consists of the coefficients $\{1/\sqrt{2}, -1/\sqrt{2}\}$ which multiply input $\{a,b\}$ to produce output $(a-b)/\sqrt{2}$. We say that the *length* of the Haar filter is 2, as both low-pass and high-pass filters have 2 coefficients and thus require an input of 2 to produce an output. Other wavelets that are generated by longer filters exhibit better performance in terms of separating the smooth and rough elements.

In the example that follows, we will use the filters $\{1/2, 1/2\}$ and $\{1/2, -1/2\}$ to avoid the ugly square roots for illustration purposes. Let us consider a signal of 8 samples (a vector of 8 values) {3,5,7,5,8,12,9,1} and let us apply the DWT. We start by first taking pairwise averages: {4,6,10,5}. We also get the following pairwise differences {-1,1,-2,4}. For any two consecutive and non-overlapping pair of data values a,b we get their average: $(a+b)/2$ and their difference divided by 2: $(a-b)/2$. The result is 2 vectors each of half size containing a smoother version of the signal, the averages, and a rougher version, the differences; these coefficients form the first level of decomposition. We continue by constructing the averages and differences from the smooth version of the signal: {4,6,10,5}. The new averages are {5,7.5} and the new differences are {-1,2.5}, forming the second level of decomposition. Continuing the process, we get the average {6.25} and difference {-1.25} of the new smooth signal; these form the third and last level of decomposition. Note that 6.25 is the average of the entire signal as it is produced by iteratively averaging pairwise averages. Similarly, -1.25 represents the difference between the average of the first half of the signal and the average of the second half. The final average {6.25} and the differences produced at all levels of decomposition {-1.25}, {-1,2.5}, {-1,1,-2,4} can perfectly reconstruct the original signal. These form the Haar DWT of the original signal: {6.25,-1.25,-1,2.5,-1,1,-2,4}. The key is that at each level of decomposition the averages and differences can be used to reconstruct the averages of the previous level.

Lossy compression in the DWT is achieved by thresholding: only the coefficients whose energy is above the threshold are preserved, whereas the rest are implicitly set to 0. If we decide to keep half as many coefficients the resulting wavelet vector contains the 4 highest (normalized by $1/\sqrt{2}$ at each level) coefficients: {6.25,-1.25,0,2.5,0,0,0,4}. Then, the compressed decomposed signal is an approximation of the original: {5,5,5,5,10,10,9,1}