Genomic big data and data storage

Thanasis Mitsis, Io Diakou, Katerina Pierouli, Eleni Papakonstantinou*

Laboratory of Genetics, Department of Biotechnology, School of Applied Biology and Biotechnology, Agricultural University of Athens, 11855 Athens, Greece

**Correspondence*: elenippk@aua.gr

Abstract

In the constantly expanding era of big data, the evolution of personalized genomics has posed new difficulties for their efficient analysis as well as storage. Next-generation sequencing technologies as well as protein studies, have been on the forefront of data generation in an explosive manner, pushing the boundaries of storage capabilities, indexing, sharing and scalability. Furthermore, due to these data's non-vectorial nature, machine-learning algorithms face inherent difficulties in their analysis. To address these issues a number of approaches have been developed. In terms of storage and data access the technologies are moving away from relational databases and towards parallel solutions.

Introduction

There exist domains of information where the data is non-vectorial (e.g., sequences in bioinformatics, graphs in sensor networks and social networks). However, a significant percentage of machine learning, data mining and indexing research requires an underlying vectorial representation to function. This has led to a practice of transforming the data to vectorial representation, essentially removing part of the original information, and setting a performance ceiling to analysis methods. Recently, large volumes of data have added a second hindrance to the analysis of non-vectorial information. Not only is it a requirement to keep the original information of this set of data, usually reflected by structural complexity or multi-modality of features, but it is further requested that such data is managed and analyzed, to exploit and learn from this information responsively. In such large volume cases, storing data is a problem, and usually, e.g., in bioinformatics, the practice is to handle batches of data as disposable: only until a first analysis is performed is the data kept. This removes the possibility of re-analysis, if new evidence appears or if a new method of analysis is made available. On the same basis, indexing and search are non-trivial when the instances sought are unstructured. Even though there exist works on, e.g., graph indexing, the problem becomes difficult when there are at least billions of

instances arriving per day and in need of (incremental) indexing. Similarly, in the machine learning and data mining communities there is a movement towards large-scale analysis, e.g., the Mahout Apache Project (Introducing Apache Mahout, 2009). However, these efforts are just beginning to bear fruit and they share one thing in common: they mostly rely on a vectorial representation of the data instances. Only quite recently have some works focused on going beyond such a representation and applying similarity-based methods (less strict than kernel methods) for clustering, classification, and regression. Nevertheless, their outcomes have not been combined with efficient calculation of these similarities and there exists no overall framework combining these achievements.

Recent developments address all of the above problems under a single, common perspective: they consider data in similarity spaces (i.e., spaces where at least one similarity function between instances is defined). From this perspective, the different types of data that are being considered (e.g., graphs, sequences, vectors) can be mapped into such a common space, where new methods for indexing, learning, and mining can directly and efficiently be applied.

Genomic big data

There are two main points of interest apparent in the domain of bioinformatics: next generation sequencing and protein study. Next generation sequencing data have all the characteristics of big data and are invaluable to research, but in practice hard to store for a long time (storage requirement) or analyze in full scale keeping analysis accurate (accuracy requirement). The unprocessed generated data are of the order of at least 30 Gb per day (which can scale by a relevant factor in the case of mapped /processed data) per sequencing machine, considering the typical case of an Illumina HiSeq 2000. Such data comes under the form of short sequencing reads, i.e. short character strings (typically having lengths in the range 75–150); each character represents a nucleotide (which is also called a "base"), and can assume the values of A (adenine), C (cytosine), G (guanine), T (thymine), or N (failure in the base calling process). The nucleotide string is usually accompanied by a corresponding string of ASCII characters, encoding the "quality" (that is, the error probability of the base calling) of each of the nucleotides. Thus, due to their sequential nature, applying vectorial representation in sequencing data, removes important information.

This is a representative case of how a typical sequencing setup works when a resequencing problem is considered. In such a case, a reference (possibly not 100% accurate) for the genome /transcriptome of the organism being sequenced is already known; one has to map the DNA /RNA sequence reads to the reference (i.e., understand where such reads come from in the reference) and find variants present in the genetic code of the specific organisms with respect to the reference.

Depending on the biological application at hand, one might need to perform several tasks on the data, possibly in several steps, with both per-read and global computations required. A typical workflow corresponding to the above use case might be as follows:

- store the reads in compressed searchable form (necessary to avoid excessive storage consumption)
- retrieve (a subset of) the reads based on some criterion, possibly depending on the experiment meta-data (for instance, select all the sequencing reads derived from a given tissue subject to a specific biological condition)
- select/process the reads, for instance: identify all the reads containing long stretches of low-quality nucleotides, and trim/eliminate them
- pattern-match the surviving data, read by read, onto a reference genome
- store the reads and their alignments to the reference genome (that is, the matches found in the genome for each read) in compressed searchable form again.

One should keep in mind that not only huge amounts of data will need to be processed each day, but also that some operations might need to be performed in an incremental way (for instance, the data produced at some point might be used to refine the results obtained from some other data generated previously, implying the reprocessing of a possibly much bigger dataset); this calls for the development of a robust and extensible high-throughput storage/matching/processing system. Many other workflows might be envisaged, but most of them share the same skeleton structure (storage, retrieval, filtering/processing, and final storage of the results).

To some extent sequencing data are intrinsically noisy (they depend on chemical reactions which are intrinsically stochastic in nature), but on the other hand high-throughput sequencing techniques have now reached a high degree of reliability, so sequencing errors are relatively rare. In addition, as mentioned above sequencing machines provide a quantification of the sequencing error at each nucleotide in terms of "qualities", which can be used to pinpoint problematic nucleotides/regions in the read. There is a clear requirement for fast and efficient analysis of whole-genome sequencing data in the emerging era of personalized genomics. Due to the continuous improvements in sequencing technologies, the current scaling of available computing, storage and analysis throughput is far lower than the scaling of the data generation rate. The induced lag between storage and processing potential and the corresponding requirements already poses problems for researchers and companies in the bioinformatics field. Since it is impossible to constantly upgrade one's hardware to keep up with the increased data production rate, the only feasible solution is to devise better algorithms, able to offer a better scaling and use the existing hardware at its full potential. Furthermore, applying machine learning in combination with these similarities might heavily facilitate biological findings. Although at a quick pace, machine learning parallelization is just starting to evolve.

The second biological problem, protein study, is one of the central biological problems for which sequence-level analysis is simply insufficient. It has been several years since protein science first provided compelling evidence that protein structural comparisons based on sequence searches are mostly ineffective: an essential contribution to the determination of the final structure actually comes from non-local structural interactions between different parts of the molecule, which need to be modeled and taken into account (cfr. e.g., homology modeling). The elucidation of the evolutionary relationship between proteins, prediction of protein structure-function and comparative modeling techniques are all based on searches in biological databases containing structural information. Repositories of known biological structures contain non-vectorial data in a large scale, the last count in the RCSB-PDB database, that holds information obtained mostly by X-ray crystallography and NMR studies, numbered a total of about 78,000 structures (Berman et al., 2000; Burley et al., 2018). The situation becomes even more challenging if one considers the huge amount of annotated /unannotated sequencing data, which holds the key to the discovery and characterization of so far unknown proteins, coming into focus from the study of an ever-growing number of species: for instance, the genomic database of NCBI GenBank contains about $320*10^9$ bases in about 200,000,000 sequence records. Bioinformatics applications (like protein-based structure-guided drug development), which represent the basis to applied biomedical research, require the ability of efficiently managing, indexing and searching the existing (and exponentially growing) databases, interrogating structural properties and similarity.

State of the art storage

For several years, under the pressure of increasing volumes of data and due to reduced hardware costs, the view of databases as centralized data access points has become vaguer. Basic paradigms of data organization and storage have been revised to accommodate parallelization, distributability and efficiency. New models and practices are applied for storage mechanics, querying methods and analysis and aggregation of the results. Search has gone beyond the boolean match, being directly linked to efficient indexes allowing approximate matching in domains ranging from string to graph matching. The main points of this progress can be summarized as follows.

From row-oriented representation nowadays the trend is to move to column-oriented representation and database systems (Abadi et al., 2009), which are the evolution of what was called "large statistical databases" in earlier literature (Turner et al., 1979). Column-oriented database systems allow high compressibility per column, by direct application of existing ratio-optimized compression algorithms (Abadi et al., 2006).

Furthermore, there are several threads pulling current database practices away from the relational paradigm. Large-scale storage and access may incorporate dynamic control over data layout. Peer-to-peer (P2P) overlays are also used in distributed stores, exchanging, e.g., index information to contributing nodes in distributed data warehouses,

where also the queries can be executed in a peer-based fashion distributing the processing load. Another alternative, related to large scale analysis is the case of Pig Latin, where an SQL-like syntax is used to provide the dataflow requirements for analysis over a map-reduce infrastructure. Other efforts provide partial SQL support, as is the case of Hive and the corresponding query language, named HiveQL (Doka et al., 2011; Gates et al., 2009; Thusoo et al., 2010). Recently, parallel databases (e.g., Oracle Exadata, Teradata) allowed high efficiency at the expense of failure recovery and elasticity. Newer approaches and versions of these parallel databases integrate a map-reduce approach into the systems to alleviate these drawbacks.

The increased availability of low-cost, legacy computers has brought cloud computing settings to the front line. Shared-nothing architectures, implying self-sufficient storage or computation nodes, are applied to storage settings (Sakr et al., 2011). Alternative cloud ecosystems are based on active data storage, where part of the computational database effort is distributed among the processing units of storage peripherals. Such an example is the case of DataLab where data operations, both read and write, are based on "sets" - essentially named collections of files - distributed across several active storage units (ASUs). Finally, task-focused storage solutions are devised to face problems in bioinformatics, social networks and network monitoring and forensics, showing how much data requirements drive the need for research on storage systems.

Especially in bioinformatics, there exist approaches that combine compressed storage and indexing under a common approach, based on sequence properties and works on indexed string storage (Ferragina et al., 2005; Arroyuelo et al., 2011). There exist cases where the system provides tunable parameters that allow a balance between data reuse and space recovery, by keeping only the data that may be reused in the near future. At this point it must be stressed that there still exist relational databases that are used for high-throughput data storage, an example being the NCBI GEO archive which supports the submission of experimental outputs and provides a set of tools to retrieve, explore and visualize data. However, even in the case of NCBI GEO, the relational nature of the underlying database is used to identify specific datasets and not specific sequences (i.e., instances). In the literature, there are methods such as Sparse Indexing, where sampling and backup streams are used to create indexes that avoid disk bottlenecks and storage limitations. Several works on time series indexing and graph indexing are also reported, beyond full text indexing. These two types of indexes, together with the string (and, thus, sequence) indexes, provide a full artillery of methods that can cope with a great variety of problems and settings.

Graph indexing is under heavy research, due to its applicability on such cases as chemical compounds, protein interactions, XML documents, and multimedia. Graph indexes are often based on frequent subgraphs, or otherwise "semantically" interesting subgraphs (Jiang et al., 2007). There exist hierarchical graph index methods, and hash based ones. A related recent work relies on "fingerprints" of graphs - derived from hashing on cycles and

trees within a graph – for efficient indexing, as part of the "Scaffold Hunter" open source software.

In the case of time series, in order to efficiently process and analyze large volumes of data, one must consider operating on summaries (or approximations) of these data series. Several techniques have been proposed in the literature for the approximation of data series, including Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT), Piecewise Aggregate Approximation (PAA), Discrete Wavelet Transform (DWT), Adaptive Piecewise Constant Approximation (APCA), Approximation (SAX), and others (Fu, 2011). Recent works based on the iSAX algorithm have focused on the batch update process of indexing very large collections of time series and have proposed highly efficiency algorithms with optimized disk I/O, managing to index "one billion time series" very efficiently on a single machine. Another system, Cypress, applies multi-scale analysis to decompose time series and to obtain sparse representations in various domains, allowing reduced storage requirements. This method can efficiently address many statistical queries, overcoming the need to reconstruct the original data.

Conclusion

While we move towards the era of personalized genomics and through the accelerating generation and accumulation of more and more data, the issue of storage, efficient mining and learning has become more prominent. To address this, various approaches have been developed, including parallel databases and graph indexing for storage, as well as the use of parallelization and GP-GPU computing to improve mining and learning in a scalable manner.

References

Abadi, D. J., Boncz, P. A. & Harizopoulos, S. Column-oriented database systems. Vol. 2 (VLDB Endowment, 2009)

Abadi, D., Madden, S. & Ferreira, M. Integrating compression and execution in column-oriented database systems. (Association for Computing Machinery, 2006)

Apache Mahout, developerWorks, 2009. https://www.ibm.com/developerworks/library/j-mahout/j-mahout-pdf.pdf

Berman, H. M. et al. The Protein Data Bank. Nucleic Acids Research28, 235–242, doi:10.1093/nar/28.1.235 (2000).

Burley, S. K. et al. RCSB Protein Data Bank: biological macromolecular structures enabling research and education in fundamental biology, biomedicine, biotechnology and energy. Nucleic Acids Research47, D464–D474, doi:10.1093/nar/gky1004 (2018).

Turner, M. J., Hammond, R. & Cotton, P. A DBMS for large statistical databases. (VLDB Endowment, 1979).

Doka, K., Tsoumakos, D. & Koziris, N. Brown Dwarf: A fully-distributed, fault-tolerant data warehousing system. Journal of Parallel and Distributed Computing71, 1434–1446, doi:https://doi.org/10.1016/j.jpdc.2011.07.008 (2011).

Gates, A. F. et al. Building a high-level dataflow system on top of Map-Reduce: the Pig experience. Vol. 2 (VLDB Endowment, 2009).

Thusoo, A. et al. in 2010 IEEE 26th International Conference on Data Engineering (ICDE 2010). 996–1005.

Sakr, S., Liu, A., Batista, D. M. & Alomari, M. A Survey of Large Scale Data Management Approaches in Cloud Environments. IEEE Communications Surveys & Tutorials13, 311–336, doi:10.1109/SURV.2011.032211.00087 (2011)

Ferragina, P. & Manzini, G. Indexing compressed text. Vol. 52 (Association for Computing Machinery, 2005).

Arroyuelo, D. & Navarro, G. Space-efficient construction of Lempel–Ziv compressed text indexes. Information and Computation209, 1070–1102, doi:https://doi.org/10.1016/j.ic.2011.03.001 (2011).

Jiang, H., Wang, H., Yu, P. S. & Zhou, S. in 2007 IEEE 23rd International Conference on Data Engineering. 566–575.

Fu, T. C. A review on time series data mining. Engineering Applications of Artificial Intelligence24, 164–181, doi:https://doi.org/10.1016/j.engappai.2010.09.007 (2011).

Volume 1: Issue 1

pages 8–14