International Journal on Artificial Intelligence Tools
Vol. 31, No. 3 (2022) 2241006 (11 pages)

DOI: 10.1142/S0218213022410068

World Scientific
www.worldscientific.com

Using Kernels to Harness the Complexity of Big Data Applications

Benjamin W. Wah

Department of Computer Science and Engineering
The Chinese University of Hong Kong, Shatin, Hong Kong
bwah@cuhk.edu.hk

Received 14 December 2021
Accepted 10 January 2022
Published 3 May 2022

This paper presents the concept of kernels to address the complexity of solving big-data applications. Their solution strategies often require evaluating domain-dependent subspaces on the big data and selecting the best result. As the data space in these problems is so vast that it is infeasible to scan the data once, we need domain-specific methods for identifying promising and manageable subspaces with a high density of solutions before looking for individual ones. To this end, we introduce kernels to represent some properties of the statistical quality, average density, or probability of solutions in a subspace to prune subspaces with suboptimal kernels. We classify various past approaches based on their analysis methods and illustrate each by an example. Our results confirm that kernels can effectively harness the complexity of solving big-data applications.

Keywords: Kernels; big-data applications; data distributions; pruning strategies, data transformations.

1. Introduction

With the pervasive availability of Internet access, big-data applications have become increasingly popular in recent years. For instance, customer data collected through multi-modal sources and real-time financial data can help guide new investment decisions. Likewise, data collected in massively online open courses can help develop strategies for improving students' learning experience. Google Trends[1] show the popularity of searching for the term "big data" peaked at 77% in late 2014 but has since been overtaken by finding the phrase "machine learning" in late 2016. This change does not reflect less interest in big data but more in using machine learning for solving big-data problems. Since 2010, numerous national projects and interdisciplinary studies have supported research in the big-data area.[2]

According to Wikipedia, big data entails capturing, storing, analyzing, processing, and visualizing large quantities of data within a tolerable amount of time using

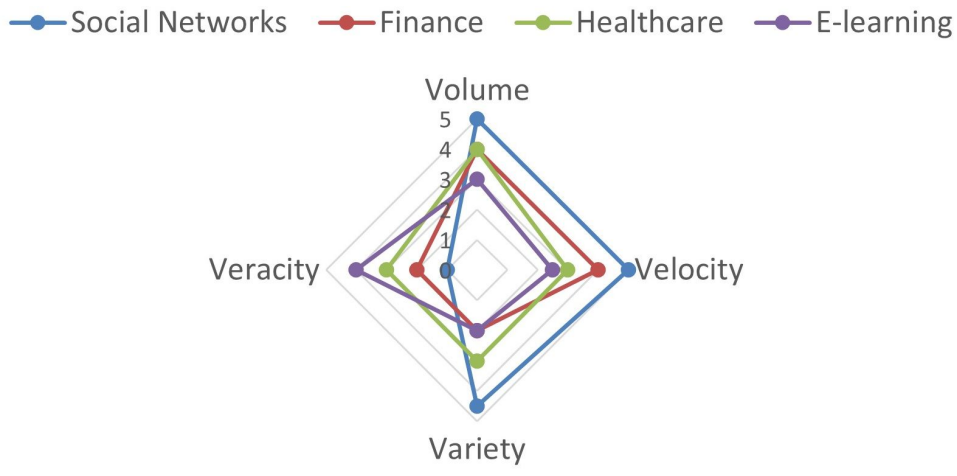

Fig. 1. Characterizing four big-data applications by their volume, veracity, variety, and velocity.

existing hardware/software infrastructures.[3] We can characterize these applications in numerous dimensions, including volume, variety, velocity, veracity, and value.

Figure 1 illustrates the relative degrees of volume, velocity, variety, and veracity for four big-data applications. Social networks have a vast size and type of data changing quickly and with different credibility. In contrast, student data in e-learning has higher veracity, less variety, smaller volume, and lower velocity than big data in social media; and data in finance and healthcare applications have their behavior between social networks and e-learning.

Solving big-data applications is highly diverse due to their domain dependence. We use seven attributes to characterize these diversities: application requirements, data properties, representations, solution goals, algorithms, infrastructure support, and theoretical foundations.

For instance, a big-data problem in astronomy or meteorology often has structured data, well-understood solution algorithms, and adequate infrastructure support than a big-data problem in social networks. The latter application has extensive multi-modal data with diverse representations, ill-defined objectives, and domain-specific solution algorithms. In contrast, a financial big-data problem may rely on quantitative and statistical approaches to minimize risk while maximizing return. Irrespective of the first six attributes, a common difficulty is the lack of a sound theoretical foundation. For example, complexity theory developed for small-data problems requires the data to be limited and manageable, rather than unbounded or not fully accessible using available computing resources. As a result, such theories are not applicable for solving big-data problems.

In short, the seven attributes lead us to conclude on a lack of standard algorithms, infrastructures, and theories for solving big-data applications. Understanding their requirements and properties is critical in harnessing their complexities.

Big-data applications generally have the following three properties.

Property 1. The applications' data space is so vast that it is infeasible to scan the data once and find all solutions or the optimal solution. Hence, we are interested in looking for good but not optimal results that satisfy the requirements.

Property 2. We characterize the search space by some incomplete and heuristically defined problem-dependent attributes. The multi-dimensional space spanned by them may be unbounded, ill-defined, or non-smooth, making it difficult to enumerate all the alternatives exhaustively. Thus, efficient search algorithms will need to traverse the space selectively by exploiting domain-specific properties.

Property 3. The solution points satisfying the application requirements are likely non-uniformly distributed and non-IID (independent and identically distributed) in the big-data space. Moreover, their statistical properties may be non-stationary over time. Hence, it will be hard to use automated methods to acquire the distributions. Moreover, statistical techniques based upon some uniform models of the underlying data may not work well.

These three properties allow us to identify three measures of big-data complexity.

Data complexity refers to the complexity of big-data space itself. With the multi-dimensional attributes defining the unbounded search space (Property 2), we cannot tackle the space complexity of big-data problems like small-data ones. In addition, existing techniques assuming structural regularity do not work well (Property 3). Examples of other complications that make it challenging to characterize data complexity include complex meta-structures with high-order information, multi-modal relationships, and non-IID and non-stationary data distributions.

Computational complexity refers to the complexity of search algorithms for solving big-data problems. Some solvers resort to statistical sampling of the data space or assume some structural properties. However, these do not work well with non-IID or non-stationary distributions (Property 3). Furthermore, because we cannot look for all the solutions (Property 1), the computational complexity of big-data problems is different from that of small-data applications. A possible measure is the time complexity to find the first solution.

System complexity refers to the complexity of hardware/software systems for solving big-data problems. Traditional infrastructures for small-data applications focus on architectures to reduce the computational or space complexity. In contrast, solution algorithms in big-data applications may need to rely on subsets of accessible data and address their dynamic nature and reference localities in their life cycle, including sensing, storage, and computations.

Our observations lead to the understanding that big-data applications have diverse and domain-dependent characteristics, possibly non-stationary statistical

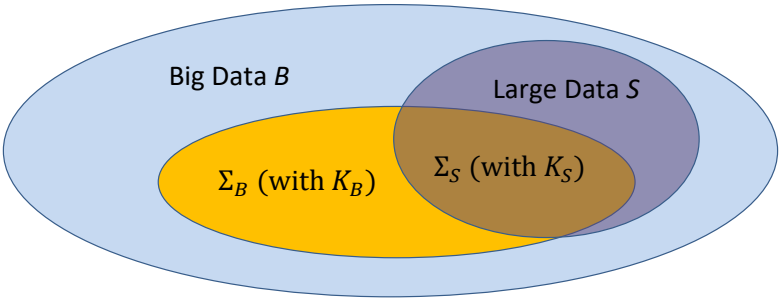

Fig. 2. The relationship between the subspace of search alternatives and its solution set.

properties, and high complexities to store, traverse, and process. To this end, a good search algorithm to look for solutions will need first to identify promising subspaces with a high solution density or a high aggregate quality before drilling down into each.

Figure 2 illustrates a big-data space B whose kernel K_B cannot be computed due to the size of B. For an application on big data B with some objectives, a search algorithm A_P defined on B can determine whether $d \in B$ belongs to its solution set $\sum_B$. Let $\sum_S \subseteq \sum_B$ be the solution set in $S \subseteq B$. For point i, $k_i^S = 1$ if $i \in \sum_S$ and $k_i^S = 0$ otherwise. The solver A_P resorts to a more feasible dimension-reduced subspace S with kernel K_S.

We define subspace S using some application-specific attributes $\bar{x} = x_1, \ldots, x_q$ (Property 2) and their values $\bar{C} = C_1, \ldots, C_q$. Let F_B be a function that maps B to S with N_S points.

$$S = F_B(\bar{x}|\bar{C}) \quad \text{(Subspace defined by mapping function } F_B\text{)} \tag{1}$$

Each point has one or more quality metrics. For an application with one metric, let d_i^S be the quality of $i \in S$. We assume that d_i^S are IID to allow the mean and standard deviation computed.

$$m_S = \frac{\sum_{i=1}^{N_S} d_i^S \, k_i^S}{N_S} \qquad \text{(Mean quality of solutions in } S\text{)} \tag{2}$$

$$\sigma_S = \sqrt{\frac{1}{N_S - 1} \sum_{i=1}^{N_S} (d_i^S \, k_i^S - m_S)^2} \qquad \text{(Unbiased estimate of std. dev.)} \tag{3}$$

Note that we do not require the IID property to apply across subspaces.

When points have a vector of multiple metrics, their vectors are assumed to be IID. We calculate the mean and standard deviation of the primary metric and constrain the remaining metrics. It is also possible to have more general problem-dependent measures, such as weighted means and downside-deviations.

We define the *kernel* K_S in subspace S to represent some properties on the statistical quality, average density, or probability of solutions in S. An example is the tuple of mean and standard deviation of the solution quality in S. We compare subspaces and prune suboptimal ones according to their kernels. The subspaces involved need not have the same IID distribution on their metrics because we do not compare their distributions.

In the next section, we classify three general approaches for finding promising subspaces according to the data distribution supported, pruning methods used, and data transformations employed. Our survey illustrates the reliance on domain-specific information when solving big-data applications.

2. Three Approaches for Finding Subspaces with Large Kernels

This section describes three practical approaches for finding subspaces with a large kernel in big-data space and explains how they help reduce the search complexity (see Table 1). In each, we present the critical domain-specific information leading to a successful solution and illustrate some example applications and their kernels.

2.1. *Structural analysis for finding lower-dimensional structures*

A structural analysis exploits the underlying structure of data entities to allow more efficient spatial traversals. We classify these techniques into structured, semi-structured, and unstructured.

Structured data consists of data with some underlying structural relationships embedded. Examples include relational data, community networks, and knowledge maps. However, such structures may not be apparent initially and may require analysis driven by observations or hypotheses to discover them.

(a) An *observation-driven approach* starts with the application's goal. It then relies on user-guided statistical analysis, limited traversals, reorganization, and experimentation to transform the original data into a new structure. After reorganizing the data by some attributes, the process selects a subspace with the best kernel and verifies its selection on new unseen data.

We illustrate this approach in the collective credit allocation in scientific publications.[4] The application aims to find the appropriate credit for a coauthor's contribution to an article perceived by peers in the scientific community. It uses structured data on all publications and their citation records. The work proposes a credit-allocation algorithm that assigns credit to each citing article and co-cited one. It then validates the algorithm by showing its success in identifying Nobel laureates from those prize-winning papers in Physics, Chemistry, and Medicine. In this application, a subspace refers to a specific organization of the coauthors and articles and their credit assignments. The algorithm proposes a new structure with an improved kernel represented as the success rate in identifying Nobel laureates from the credit assigned to each piece and its coauthor.

Table 1. A summary of types of subspaces with large kernels in big-data analysis.

Method	Description and Examples	Large-Kernel SS
	Structural Analysis	
Structured data analysis	• Relations among structured entities • Clusters, social networks, small-world model, cooperative networks	Macro-level low-dimensional data, motifs, and patterns
Semi-structured data analysis	• Relations among semi-structured data entities in some dimensions/attributes • Social media journal, blogs, twitters, rumors over the temporal dimension	Relationships of objects in a transformed dimension
Unstructured data analysis	• Lower-order relations among entities • Hybrid textural data, images, video, multimedia	Lower-dimensional representation of data-entity relations
	Pruning Analysis	
Random sampling	• Solutions uniformly distributed in space • A simplifying assumption	A random subspace in the big-data space
Heuristic sampling	• Eliminating search subspaces by the heuristic/greedy properties on data relations, distribution, or clusters	Heuristic properties of dominating search subspaces
Dominance sampling	• Pruning suboptimal subspaces by dominance relations without affecting quality • Pareto frontiers in financial data	Pareto boundary of solutions; dominating subspaces
	Transformational Analysis	
Spatial transformation	• Projecting spatial relations of data into a different space to reduce complexity • Social-media data on communities structured by user interests	A projected subspace of data with reduced dimension and complexity
Temporal transformation	• Projecting temporal relations of data into an alternate temporal scale • Propagating social media data over time	A projected temporal subspace of data
Space-time transformation	• Projecting along spatial and temporal dimensions to alternate subspaces • Recent election news over space/time	Projected events in transformed space and time

(b) A *hypothesis-driven approach* starts from a hypothesis based on informal observations and phenomena, proposes a model and a goal using these observations, and verifies the model experimentally and analytically. The objective may be ill-defined initially but refined as the model is more well-defined. An example of the approach is discovering the complex relations among humans in a society. The Small-World Problem[5] starts from a hypothesis and some initial observations on

any two persons in the world knowing each other with a probability distribution. Recent analysis refines and verifies the model to show that the diameter of a small-world model is exponentially smaller than its size N and bounded by $(\log N)^2$.[6] In this development, different subspaces refer to dissimilar ways of modeling the small-world phenomenon, with a kernel on the probability of two people knowing each other in a subspace.

Semi-structured data consists of data entities with relations in one or more dimensions but none otherwise. Examples include social-media journals, blogs, twitters, and financial news that evolve over a temporal dimension but have non-apparent structures otherwise. Further, the evolution may be irregular and unpredictable over time. To illustrate the approach, consider applying temporal scaling on information propagation from one person to another in an evolving social network. The application aims to predict the information propagation dynamics and estimate future propagation probabilities among individuals. The problem is semi-structured because although the temporal dimension is of interest, the application has many other (unstructured) dimensions. In solving the problem, a key observation[7] shows a power-law relationship between the probability a message propagates to two individuals and the length of time since their last interaction. Scaling temporal data according to this relation allows the data to be organized into a uniform sequence with low prediction errors (kernel).

Unstructured data consists of entities without uniform relations; instead, there may be multiple distinct relationships among the objects' subsets. For instance, Facebook has a massive data archive of hybrid textual data, images, video, and multimedia. To illustrate the mapping of unstructured data to aid information retrieval, consider the research on relevance preserving projection and ranking.[8] The application searches an extensive database of web-based images to satisfy some application requirements. The paper proposes to refine text-based search results by mining images' visual content. It offers an attribute extraction algorithm that transforms the original high-dimensional feature space into a lower-dimensional hypersphere space by preserving the images' various structures and relevance relationships. The result is an improved retrieval of relevant images from the database. In this context, the original unstructured space is transformed into a lower-dimensional hypersphere space, and its kernel is the improved retrieval accuracy using the new structure.

In summary, our examples show that discovering a proper structure for solving a big-data problem is domain-dependent and often requires an in-depth understanding of the application. This domain-specific information is essential for reducing the vast number of possible subspaces to a manageable scale. The discovery of a suitable subspace would lead to a kernel with high-quality metrics, rendering the application problem solvable.

2.2. *Pruning analysis for eliminating subspaces with suboptimal kernel*

The basic idea is to use domain-specific data properties to identify subspaces with a dominating kernel and eliminate those dominated ones to reduce search complexity. We consider two general classes here: heuristic and dominance pruning.

Heuristic pruning eliminates search subspaces using heuristic properties on data relations and distributions. It uses partial, experiential, or incomplete information to reduce the search space without relying on a formal analytical foundation. As a result, it is often evaluated on benchmarks, and its performance is not guaranteed on unseen data.

An example application of the approach is on burst topic discovery in microblogs (such as Twitter).[9] The application aims to find high-quality topics from these blogs that are typically short, diverse, and noisy posts, with primarily common and meaningless issues. There are many existing heuristic methods for filtering burst topics from non-burst ones, such as clustering burst features (like words or phrases) and applying existing topic-discovery models to find burst topics. Another approach divides the data stream into time slices and uses a probabilistic model for burst-topic modeling on each interval. Although these heuristics help prune the original big-data space into lower-dimensional subspaces and reveal the main topics in general texts, they are not designed explicitly for microblogs. In this context, our kernel is the quality of burst topics found in a reduced subspace evaluated by benchmarks.

Dominance pruning use a dominance relation[10] between two subspaces S_i and S_j to prune S_j. It is a formal property because with the available information on S_i and S_j and without evaluating all their data points, we can prove that the kernel K_j of S_j cannot be better than K_i. The final dominating kernels found are, therefore, guaranteed to be better as subspaces with suboptimal kernels will be pruned.

For example, consider the search for proper control of a real-time interactive multimedia application to allow users in perceiving high-quality interactions. Examples of such applications include real-time interactive video conferencing and online games. The application is a big-data problem because the space of perceptual quality as a function of controls is vast, ill-defined, and possibly unbounded. In our past work, we observed two monotonicity properties (or dominance relations) based on just-noticeable differences (JND) to reduce the complexity of traversing the subspace of perceptual qualities.[11] The approach allows us to develop an optimal polynomial-time binary-divide algorithm for finding the perceptual quality (kernel) of all the data points in a JND profile. The latter provides a good operating point for interactive multimedia applications at run-time. For example, optimizing the vector of actions in a multi-player online game makes it possible to conceal the virtual delay of multi-player actions. The result allows users to perceive that the game

runs smoother (with less and negligible virtual delay) without compromising the synchronization of action orders.[12]

In general, heuristic pruning is simpler to design but does not guarantee the kernels' quality, and its effectiveness is often evaluated by benchmarks. In contrast, dominance pruning ensures that only suboptimal subspaces will be pruned but requires substantial domain knowledge and understanding in its design.

2.3. *Transformations to project data into new dimensions*

This approach transforms a big-data space into a different dimension with a better kernel. The choice of a transformation is heuristic as well as domain-specific, and its generalizability is limited by the dimension(s) used. There are generally three transformations: spatial, temporal, or a combination of space and time.

Spatial transformation entails projecting the original big-data space into a more structured spatial organization. Since the number of projections is extremely large or infinitely many, the choice is often driven by heuristics or domain knowledge.

For example, consider a weighted graph clustering problem for community detection.[13] This involves detecting users' communities (or nodes with more connections) in an Internet social network (such as Facebook). Using a heuristic observation that overlaps in communities are densely connected, the research maps these overlapping, hierarchically nested, and non-overlapping communities into distinct subspaces. In this context, a subspace represents an embedding of the original data using a specific clustering approach. The subspace with the best clusters has improved kernel with high-quality metrics of users engaged in communities.

Temporal transformation entails the projection of big data into a different temporal scale that allows a better understanding of its entities. Please refer to the example on semi-structured data in Sec. 2.1[7] for applying the approach.

Space-time transformation involves projecting the big-data space along the temporal and spatial dimensions to reduce its subspace. Consider an example application that continuously updates streaming data in persistent sketching over sliding windows.[14] Instead of maintaining all the previous versions of streamed data at every update, the system only holds a sliding window of a small matrix (over space and time) that approximates the entire data set (called persistent data sketch). In this context, each subspace involves a specific space-time transformation of all the previous versions of the streamed data; a suitable subspace has a large kernel that makes it possible to query a significant fraction of all the events over time.

This section has examined various approaches for transforming the original big-data space into a different structure to allow a more efficient traversal of its search space when solving the application problem. Our examples show that the proper transformation techniques are application-specific and require domain knowledge

aided by observations and experiments. Thus, the approach is different from traditional information retrievals (IR)[15] that use domain knowledge to transform the data into meta-data before applying existing retrieval strategies.

3. Conclusions

This paper presents methods for harnessing the complexity of solving big-data applications using kernels defined as the statistical quality metrics of solutions in a subspace evaluated by the solver. Focusing on subspaces is more effective than individual solutions because we can identify better subspaces at a macro scale. Using kernels, we have illustrated three approaches to prune inferior subspaces. Our survey of methods for solving big-data applications shows their dependence on domain-specific information. Such knowledge helps identify promising subspaces searched by the solver, which otherwise would be prohibitive using brute-force enumerations.

Acknowledgments

This project was supported by the National Key Basic Research Program of China (973 Program) No. 2014CB340401.

References

1. Google, Comparing trends on searching big data versus machine learning (2018), https://trends.google.com/trends/explore?date=today%205-y&geo=US&q=big%20data,machine%20learning.
2. X. Jin, B. W. Wah, X. Cheng and Y. Wang, Significance and challenges of big data research, *Big Data Research* **2** (2015) 59–64.
3. Wikipedia, Big data (2018), https://en.wikipedia.org/wiki/Big_data.
4. H.-W. Shen and A.-L. Barabasi, Collective credit allocation in science, *Proc. National Academy of Sciences* **111**(34) (2014) 12325–12330.
5. S. Milgram, The small-world problem, *Psychology Today* **1**(1) (1967) 61–67.
6. J. M. Kleinberg, Navigation in a small world, *Nature* **406** (2000) 845.
7. J. Huang, W.-Q. Wang, H.-W. Shen, G. Li and X.-Q. Cheng, Temporal scaling in information propagation, *Scientific Reports* **4**(5334) (2014) 1–6.
8. Z. Ji, Y. Pang and X. Li, Relevance preserving projection and ranking for web image search reranking, *IEEE Trans. on Image Processing* **24**(11) (2015) 4137–4147.
9. X. Yan, J. Guo, Y. Lan, J. Xu and X. Cheng, A probabilistic model for bursty topic discovery in microblogs, in *Proc. 29th AAAI Conf.* (Austin, TX, 2015).
10. A. Jouglet and J. Carlier, Dominance rules in combinatorial optimization problems, *European J. of Operational Research* **212**(3) (2011) 433–444.
11. J. X. Xu and B. W. Wah, Optimizing the perceptual quality of real-time multimedia applications, *IEEE Multimedia* **22**(4) (2015) 14–28.
12. J. X. Xu and B. W. Wah, Consistent synchronization of action order with noticeable dealy in online games, *ACM Trans. on Multimedia Computing, Communications, and Applications* **8**(1) (2017).
13. J. Yang and J. Leskovec, Overlapping community detection at scale: A nonnegative matrix factorization approach, in *Proc. ACM Int. Conf. on Web Search and Data Mining* (*WSDM*) (Rome, Italy, 2013).

14. Z. Wei, X. Liu, F. Li, S. Shang, X. Du and J.-R. Wen, Matrix sketching over sliding windows, in *Proc. ACM SIGMOD* (San Francisco, CA, 2016).
15. M. Sanderson and W. B. Croft, The history of information retrieval research, *Proc. of the IEEE* **100** (2012) 1444–1451.