EDA and ML – A Perfect Pair for Large-Scale Data Analysis

Ryan Hafen, Terence Critchlow
Pacific Northwest National Laboratory
Richland, WA USA
Ryan.Hafen@PNNL.gov, Terence.Critchlow@pnnl.gov

In this position paper, we discuss how Exploratory Data Analysis (EDA) and Machine Learning (ML) can work together in large-scale data analysis environments. In particular, we describe how applying EDA techniques and ML methods in a complementary fashion can be used to address some of the challenges faced when applying ML techniques to large, real world data sets, and discuss tools that help do the job. This iterative approach is demonstrated with a simple example of how extracting events from a historical sensor data set was enabled by iteratively identifying and filtering various types of erroneous data.

Keywords-exploratory data analyisis, machine learning, large-data analysis

I. INTRODUCTION

In this short position paper, we discuss how Exploratory Data Analysis (EDA) and Machine Learning (ML) can work together to enable effective large-scale data analysis. One of the significant challenges applying ML to large data sets is creating the training data, evaluating the results, and determining how to refine the data to improve the performance of the algorithm. While solutions to these problems have been established on relatively small data sets, overcoming them in the realm of big-data remains a challenge. Recent advances in statistical computing environments have allowed EDA techniques to transition from small data sets into significantly larger ones. This transition allows EDA to be used conjunction with ML to address these challenges.

Section II provides a working description of EDA and describes a typical EDA analysis environment. In Section III, we describe recent advances that have made EDA feasible for large data analysis scenarios. Section IV describes how EDA and ML can be effectively combined to provide an enhanced analytical environment. Finally, in section V, an example of a large-scale analysis using this combined approach is presented.

II. WHAT IS EDA?

EDA is the interactive, hypothesis-driven exploration of data. In this approach, analysis of the data is driven by a set of questions and/or suspicions, but as the analysis proceeds, results are examined not only to answer the original question, but to potentially gain additional, possibly unexpected, insights about the data – after exploring we have some answers and more questions [1]. It is important to recognize that the exploration is directed, based on the initial questions being asked. "*We explore data with expectations. We revise our expectations based on what we see in the data. And we iterate this process.*" [2]

Because of the importance of the user's direction, EDA is greatly informed by domain expertise and domain-driven models of how data should behave. It often plays an important role in either confirming *a priori* expectations or suggesting new hypotheses based on the data.

As John Tukey, the "father of EDA" points out, "*Exploratory data analysis is an attitude, a flexibility, and a reliance on display, NOT a bundle of techniques.*" [3]. However, in practice, the availability of good tools and techniques is critical to the useful application of EDA to a specific problem. In particular, both numerical and visual methods are typically applied to the data to identify trends and patterns as well as deviations from the expected patterns. This requires tools that support iterative interaction with the data, allows users to easily describe complex analytics, and provides the ability to easily display analysis results.

A. Requirements for an EDA Environment

The quote from Tukey just cited highlights two important aspects of EDA: flexibility and the reliance on display. Any successful EDA environment must support both components.

EDA environments must be flexible. The EDA approach to analysis is inherently iterative. The models or algorithms that capture the salient features of the data are inherently complex, and initial descriptions are never complete. Typically, analysis begins with either simple summaries or displays of the data. These provide insight into the data, leading to new ideas. Algorithms to investigate the validity of these new hypotheses are developed and evaluated. These algorithms lead to new insights, and the process is repeated. Inherent in a successful environment is the ability to rapidly implement the method or methods in order for the new idea to be examined. Thus, a successful EDA environment must provide a high-level, *interactive language for data analysis* that has direct support for complex numerical, statistical, and machine learning methods.

Visualization is critical in an EDA environment. EDA requires a human to drive the data exploration process. It is well documented that humans have an extraordinary ability to identify patterns in complex data when presented with the information through an appropriate visualization. Thus, the most effective way for a human be of value in this loop is to enable easy visualization of the analysis results. To accomplish this, a successful EDA environment must

provide a flexible interface for creating a wide variety of visualizations.

III. TOOLS FOR EDA WITH LARGE DATA

EDA principles have been successfully used for many years [4], and several popular environments, including R [5] and MatLab [6], have been developed. These environments allow users to easily and interactively explore their data using scripting languages developed exclusively to express complex mathematical and statistical algorithms. R is particularly popular because it is an open source project, allowing the broad research community to contribute thousands of specialized analysis packages. The supporting ecosystem makes it easy to identify, import, and use algorithms that perform every imaginable analysis task.

Until recently, R – and similar EDA environments – have been limited to relatively small data sets because they require the ability to analyze the entire data set in memory. To facilitate the iterative process of EDA with large data, it is important that methods can be applied at scale so that results can be received in a reasonable amount of time. Fortunately, recent research has advanced the current state of the art, making EDA on multi-TB data sets feasible.

In this section, we introduce the R and Hadoop Integrated Programming Environment (RHIPE) [7] that provides this environment, and discuss some research efforts aimed at scalable analytic and visualization methods based on this platform.

A. RHIPE

RHIPE (pronounced hree-pay') is a merger of the R statistical programming environment and the Hadoop distributed processing system. RHIPE allows an analyst to carry out analysis of complex big data wholly from within R, using Hadoop to handle the distributed computing. RHIPE provides a dynamic language for analysis of big data.

RHIPE is tightly integrated with Hadoop, providing support for many Hadoop features, such as combiners, custom partitioning, distributed cache, traditional input/output formats such as map and sequence files, text files, and HBase, as well as support for adding custom jars for other input formats. Data is serialized using protocol buffers, making it easy to share data with other applications.

The compute paradigm for Hadoop is MapReduce. MapReduce provides a versatile high-level parallelization to solve many data-intensive problems through use of user-specified Map and Reduce functions, including many statistical and machine learning algorithms [8]. In this approach, an algorithm is split into a map component – which performs an initial data processing step and outputs a set of key-value pairs – and a reduce component – which performs a second processing step on all values with the same key. This two-step approach is relatively general, and has been widely adopted by the data analysis community.

However, many algorithms that fall into the MapReduce paradigm are not practically feasible, such as algorithms that require several iterations or require large amounts of state information to be saved between iterations [8]. Further, many algorithms do not fit into the MapReduce paradigm at all, for example certain graph analysis cannot be easily translated to self-contained map and reduce steps, but rather require communication across tasks.

Divide and Recombine (D&R) is a new statistical approach to the analysis of large complex data [9] that seeks to overcome some of the limitations of MapReduce. This approach is similar to MapReduce in that data is divided into subsets, analytic methods are applied to each subset in an embarrassingly parallel manner, and the outputs of each method are recombined to form a result for the entire data. The key difference is in how the data subsets are generated. Specifically, subsets are created such that the results of applying a method independently to each subset can be recombined (e.g. through averaging model coefficients) to achieve a very good approximation to a global model fit. Similar ideas are being researched [10]. The challenge is to develop subsetting algorithms that will produce good results while being sufficiently simpler than the analysis algorithm being approximated. Assuming that the data can be effectively subset, D&R allows a data analyst to apply almost any existing statistical or visualization method to large complex data. Current research in D&R is to develop "best" division and recombination procedures for various analytic methods.

B. Visualization Databases

As previously noted, EDA requires the ability to visualize the analytical results. Additionally, for comprehensive analysis of the data , it is important to be able to visualize the data in detail. One approach to this is to subset the data and examine a few subsets in detail. But this may not be sufficient to identify patterns which would span subsets or occur infrequently. In these cases, it is useful to apply a visualization method to every subset, creating a collection of "small multiples" [11] which spans the entire data set. Viewing a collection of small multiples is effective in revealing repetitions or changes, while allowing an analyst to see the data at finer levels of detail. This visualization approach has been an effective tool for many years and is built in to many successful visualization software packages, such as the lattice package for the R statistical computing environment [12] and the underlying trellis framework [13]. A collection of small multiples is referred to as a single display. Since many displays are created during analysis, an effective way to manage them is required. This is the role of a visualization database (VDB) [14].

This capability is particularly important when working with large data sets since as the size or complexity of the data increases, the number of small multiples increases beyond the capacity of the human mind to simultaneously capture the details and larger scale trends or patterns. If these displays are stored and available for additional analysis, it is possible to have the computer determine which displays might be interesting [15]. This is accomplished through calculating diagnostic quantities, or cognostics [16], for each panel to provide a rank for its potential usefulness. Cognostic quantities can be simple statistical quantities (range, standard deviation, model coefficients) or can be tailored to the visualization at hand to differentiate among

large numbers of panels with respect to context-relevant attributes.

IV. USING EDA AND ML TOGETHER

There has been significant research in the area of distributed algorithms for machine learning, leading to an impressive collection of tools for large-scale data analysis. Unfortunately, difficulties can arise when using these tools in practice. In particular, with numerical methods alone, it can be difficult to:

- identify and understand incorrect or anomalous records within the data set that may be negatively affecting the ML algorithm
- determine or understand the effectiveness of the model that that ML algorithm has learned, and how to change the model to provide better results
- interpret the scientific meaning or impact of the algorithm results

As outlined below, EDA can be used to address these concerns, forming a complementary cycle where EDA and ML are performed iteratively.

A. Data Preparation and Identification of Incorrect Data

EDA naturally complements ML in the preprocessing and cleaning stage. Environments suited to EDA, as outlined previously, can be very helpful with tasks like getting the data in the proper format, getting a "feel" for the data, cleaning the data, performing feature selection, and determining appropriate directions to take. Often these tasks are pushed to the background, with all importance placed on the learning stage. However, in the experience of practitioners, these tasks constitute 80% of the effort that determines 80% of the value of the ultimate results [17]. These tasks are typically thought of as a one-time up-front cost, but in practical applications, they are ongoing efforts, with ML algorithm results bringing new issues to light at each iteration. In section V, we illustrate this point with an example.

B. Understanding the Effectiveness of the ML Algorihtm

Often the overall classification accuracy or prediction error metrics from a ML algorithm do not tell the whole story. When focusing only on these metrics, we run the risk of choosing the best performer from a class of poor performers. Injecting EDA techniques into the process can help provide insight into what regions of the design space are performing well or poorly, and spark ideas for improvements to ML algorithms. A little bit of strategy from the human (EDA) can go a long way in helping the immense tactical power of the machine (ML).

The benefit of EDA, however, is based on the ability of the analyst to guide the process. This benefit is minimized when the analyst is unable to guide the process, for example because they are unfamiliar with the domain or unclear what they are looking for. In this case, an unsupervised approach is likely to generate a more useful model. This type of approach is also likely to be more efficient when the features that comprise the model are known in advance, since the analyst would need to develop a similar model from scratch.

As EDA is a rather general approach, it is difficult to prescribe a concrete set of steps that illustrate its use in this context. However, in our experience we have noted the use of exploratory techniques, primarily driven by visualization, in helping to validate assumptions of the ML method being applied, identify transformations that yield a far more parsimonious model, identify interactions missed by the ML method, and leverage domain expertise. The importance of the use of EDA in the ML process is well-summarized by Tukey: "*How do we avoid analysis that the data before us indicate should be avoided? By exploring the data-before, during, and after analysis for hints, ideas, and, sometimes, a few conclusions*." [3]

C. Interpreting Results

When the context of an analysis relates to scientific discovery, the goal is not necessarily the prediction or classification with the most accuracy, but a sound understanding of the phenomena upon which the data is based. In this case, we are most interested in building interpretable models, checking if data supports hypothesized models, and looking for new phenomena. The role of ML in interpretable scientific discovery has been a topic of debate [18], but we see ML methods as completely necessary here, with EDA helping with the interpretation of results.

V. USE CASE

A detailed example of how EDA an ML can be iteratively used to provide novel analytical results is outside the scope of this paper. However, instead a simple use case demonstrates how combining EDA based data cleaning with event extraction can be used in a complementary fashion on a realistic data set: specifically, given power grid sensor data, identify and extract records detailing a specific type of event, an islanding event, out of the data.

The data is a time series of sensor readings, from a network of 38 sensors on a single network, each reporting a set of values thirty times per second. Each record can be thought of as a tuple (t, s, f, v) where t is the timestamp, s is the sensor, f is a flag value indicating the sensor's state, and v is a vector of values reported by that sensor at the specified time.

When operating normally, the network connects all sensors. Given physical constraints, each sensor should report closely related values in this case. Islanding occurs when network connections are broken, and the network is divided into two smaller, disconnected networks. In this case, the sensors may report significantly different values if they are on different partitions. A model was easily developed that could identify islands within a simple test data set. Unfortunately, when the model was applied to the entire data set an unusually high number of islanding events were detected.

EDA was used to determine why the model was not performing as expected: the sensor data stream had extremely high occurrences of erroneous data, leading the model to generate incorrect results.

Starting with an initial collection of statistics, EDA was then used to identify three distinct types of errors within the data set: previously unknown flags (f) which indicated sensor errors, sensors producing a single value, and sensors generating white noise. The initial collection of erroneous records was identified when error conditions, previously unknown to local experts, were identified based on an analysis that identified a 100% correlation between f and v. The second set of erroneous records was identified when v was unchanged for a length of time determined by a statistical analysis of the data, flagging repeated sequences of a length exceeding the extremities of the tail of a fitted geometric distribution. The final set of records was defined by those records where there is no significant autocorrelation between values on a given sensor across a small timescale. The interested reader is directed to [19] for additional insight into this analysis. .

Determining the specific statistical models that identified the erroneous data required iteratively computing and analyzing statistical measures against the underlying data set. This would typically begin with an analysis computed against the entire data set using RHIPE, which would take between 10 and 30 minutes. From there, we would identify subsets of the data that appeared to be of interest – for example, records with a specific flag value. These data subsets were then extracted from the entire data set and analyzed individually to determine if there was a pattern of interest (e.g. a specific flag always had a certain value). Once a potential pattern was identified, and defined as a model, the model was validated by running it against the entire data set and analyzing all instances identified by the model. Once the model was validated, it could be used to either filter the data stream in real time to improve down-stream analysis tasks or filter the archived data as appropriate for off-line analysis.

After each filter was developed, the event detection model was re-run. The results from each run were then analyzed using EDA to determine why there were more events than expected. This iteration continued until the results produced by the event detection model on the filtered data was validated.

In our application, EDA was particularly useful because it was impossible to describe in advance the patterns that needed to be identified. For example, it would be extremely unlikely to know in advance that sensors will occasionally generate data that looks like white noise. Without this pattern being known in advance, it is unlikely that training sets for supervised learning algorithms will include sufficient data for an algorithm to determine the pattern. Similarly, if the pattern that represents the event is unexpected, it may be outside the scope of what traditional machine learning algorithms may be able to identify. Many traditional algorithms would not generate correlation measures both along a time series and across different time series, which would have missed identifying some of our erroneous records. Finally, in high-dimension space expert insights can be critical in reducing the search space to a manageable set of alternatives. For example, knowing that our network should not remain constant for an extended period of time, despite constant values happening frequently, led to the identification of the appropriate geometric distribution analysis that identifies when a sensor is stuck.

VI. CONCLUSIONS

Traditional machine learning algorithms are effective at identifying, classifying, and categorizing large-scale data sets using complex statistical models. Unfortunately, training and validating these models requires access to data subsets that accurately represent the larger data set. Ensuring the model has sufficient information to learn everything it needs to know is a challenge unto itself. Exploratory data analysis techniques can be used to address these concerns. Historically, such interactive techniques have been excluded from this domain because they were limited to small data sets. This has changed with new, scalable EDA frameworks such as RHIPE, which combine the flexibility of R and the scalability of Hadoop. Based on our experience iteratively validating event detection models, we believe that the application of EDA techniques in combination with ML provides the opportunity to overcome the limitations inherent in both approach and provides better analytical results.

REFERENCES

[1] L. Wilkinson. The Impact of Tukey's Exploratory Data Analysis, *Chicago chapter of the American Statistical Association Spring Conference*, May 5, 2000.

[2] L. Wilkinson, A. Anand, and R. Grossman. High-dimensional visual analytics: Interactive exploration guided by pairwise views of point distributions. *Visualization and Computer Graphics, IEEE Transactions on*, 12(6):1363–1372, 2006.

[3] J. W. Tukey. We need both exploratory and confirmatory. *American Statistician*, pages 23–25, 1980.

[4] J. W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, Reading, Massachusetts, U.S.A., 1977.

[5] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012. ISBN 3-900051-07-0.

[6] MATLAB version 7.10.0. Natick, Massachusetts: The MathWorks Inc., 2010.

[7] RHIPE: www.rhipe.org

[8] C. T. Chu, S. K. Kim, Y. A. Lin, Y. Y. Yu, G. Bradski, A. Y. Ng, and K. Olukotun. Map-reduce for machine learning on multicore. *Advances in neural information processing systems*, 19:281, 2007.

[9] S. Guha, R. Hafen, J. Rounds, J. Xia, J. Li, B. Xi, and W. Cleveland. Large complex data: divide and recombine (D&R) with rhipe. *Stat*, 1(1):53–67, 2012.

[10] Kleiner, A., Talwalkar, A., Sarkar, P., & Jordan, M. I. Bootstrapping big data. *Big Learn,* 2011.

[11] E. Tufte, N. Goeler, and R. Benson. Envisioning information, volume 21. Graphics Press Cheshire, CT, 1990.

[12] D. Sarkar. *Lattice: multivariate data visualization with R.* Springer Verlag, 2008.

[13] R. A. Becker, W. S. Cleveland, and M.-J. Shyu. The visual design and control of trellis display. *Journal of Computational and Graphical Statistics*, 5(2):123–155, 1996.

[14] S. Guha, P. Kidwell, R. P. Hafen, and W. S. Cleveland. Visualization databases for the analysis of large complex datasets. *In Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009.

[15] J. Friedman and W. Stuetzle. John W. Tukey's work on interactive graphics. *Annals of Statistics*, pages 1629–1639, 2002.

[16] W. S. Cleveland. The Collected Works of John W. Tukey: Graphics 1965-1985, volume 5. Chapman & Hall/CRC, 1988.

[17] T. Dasu and T. Johnson. *Exploratory data mining and data cleaning*. Vol. 442. Wiley-Interscience, 2003.

[18] L. Breiman. "Statistical modeling: The two cultures (with comments and a rejoinder by the author)." *Statistical Science* 16.3 (2001): 199-231.

[19] R. Hafen, T. Gibson, K. Kleese van Dam, and T. Critchlow. *Data Mining Applications with R*, chapter Power Grid Data Analysis with R and Hadoop. Elsevier, In print.