Very Fast Estimation for Result and Accuracy of Big Data Analytics: the EARL System

Nikolay Laptev[1], Kai Zeng[2], Carlo Zaniolo[3]

University of California, Los Angeles
USA

[1]nlaptev@cs.ucla.edu
[2]kzeng@cs.ucla.edu
[3]zaniolo@cs.ucla.edu

Abstract— **Approximate results based on samples often provide the only way in which advanced analytical applications on very massive data sets (a.k.a. 'big data') can satisfy their time and resource constraints. Unfortunately, methods and tools for the computation of accurate early results are currently not supported in big data systems (e.g., Hadoop). Therefore, we propose a nonparametric accuracy estimation method and system to speed-up big data analytics. Our framework is called EARL (Early Accurate Result Library) and it works by predicting the learning curve and choosing the appropriate sample size for achieving the desired error bound specified by the user. The error estimates are based on a technique called bootstrapping that has been widely used and validated by statisticians, and can be applied to arbitrary functions and data distributions. Therefore, this demo will elucidate (a) the functionality of EARL and its intuitive GUI interface whereby first-time users can appreciate the accuracy obtainable from increasing sample sizes by simply viewing the learning curve displayed by EARL, (b) the usability of EARL, whereby conference participants can interact with the system to quickly estimate the sample sizes needed to obtain the desired accuracies or response times, and then compare them against the accuracies and response times obtained in the actual computations.**

I. Introduction

In today's fast-paced business environment, obtaining results quickly represents a key desideratum for 'Big Data Analytics' [8]. Well developed theory and extensive experiments from statistics show that in analyzing large data sets, performing careful sampling on the data and computing early results from such samples provide a fast and effective way to obtain approximate results within the prescribed level of accuracy for most applications. However, while the need for approximation techniques grows with the size of the data sets, general methods and techniques for handling complex tasks are still lacking in both MapReduce systems and parallel databases that claim 'big data' as their forte.

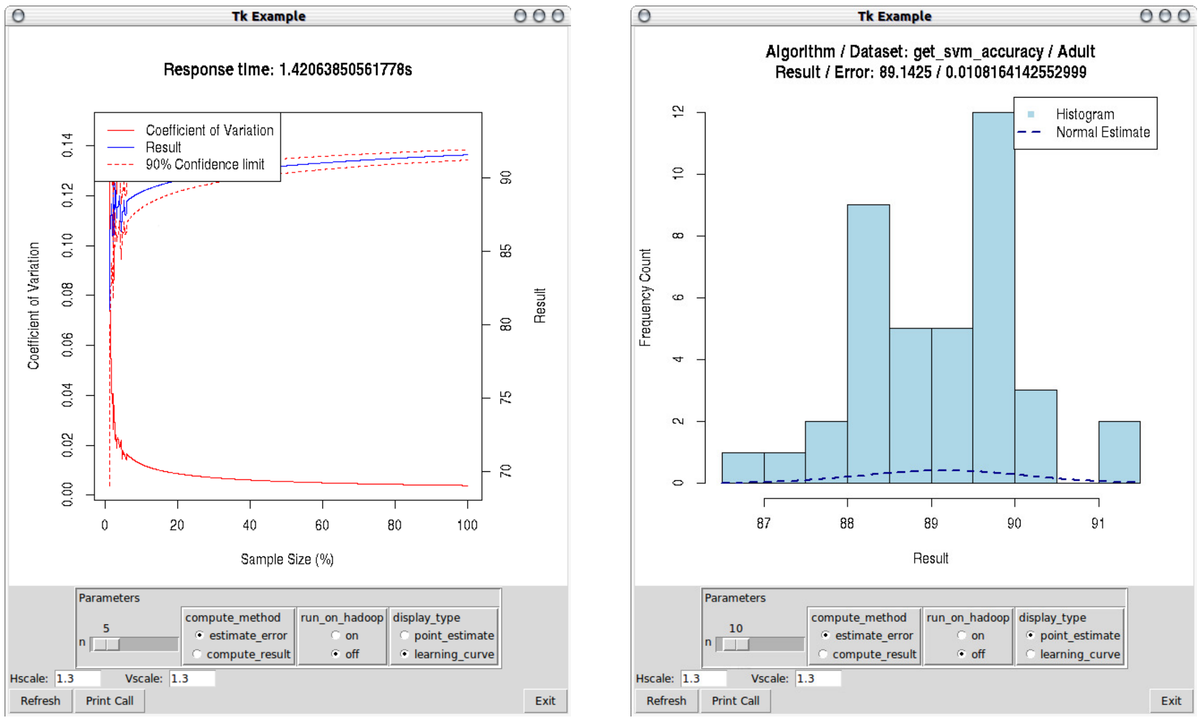
Our Early Accurate Result Library framework (EARL) [9] has been designed and developed to provide this much needed functionality thus bridging the gap between the mushrooming data-sizes and the response time requirements. To achieve this, we explore and apply powerful methods and models developed in statistics to estimate results and the accuracy obtained from sampled data [16], [4]. We propose a method and a system that optimize the work-flow computation on massive data-sets to achieve the desired accuracy while minimizing the time and the resources required. In this demonstration we will present the EARL prototype[1] and its intuitive GUI interface which helps the user through the successive phases of 'big data' analytics. This prototype represents also the starting point for a formal study of the tool usability we are planning to conduct. The early approximation techniques presented are also important for fault-tolerance, where only a portion of the data is available and the error estimation is required to determine if node recovery is necessary.

As shown in the paper [9], in order for sampling to be effective at reducing latency two challenges must be addressed in error estimation: *efficiency* and *scalability*. To address these challengers, EARL relies on the bootstrapping technique described in [16] which works for arbitrary analytical functions. This technique relies on resampling methods, where a number of samples are drawn from s. The function of interest is then computed on each sample resulting in the sampling distribution used for assigning measure of accuracy to sample estimates. When computing an analytical function in EARL, a uniform sample, s, of stored data is taken; then, based on the confidence of error prediction, this is enlarged as necessary to construct the learning curve from which the required sample size can be determined.

Improving on this general framework, EARL addresses the efficiency challenge via delta maintenance techniques and the scalability challenge via Hadoop integration. Delta maintenance techniques reuse the results across samples of different sizes, while the integration with the popular MapReduce framework allows for exploiting the inherent parallelism of bootstrapping. Furthermore EARL's simple API allows for easy specification of mining algorithms that take full advantage of our framework.

The conference participants will experience EARL's ease of use first hand. The user interacts with the EARL system via a three step process: (i) configuration, (ii) required sample size estimation and (iii) computation of the actual results. In (i) the user uses EARL's API to specify her own algorithm, or choose one from the existing library. The dataset and the execution mode (distributed or local) is also chosen during the configuration stage. In (ii) the learning curve is first derived from a small dataset and then used to predict the sample size

[1]EARL will be released for experimental and non-commercial use at [1].



Response time: 1.42063850561778s

Parameters: n 5; compute_method: [✓ estimate_error] compute_result; run_on_hadoop: on [✓ off]; display_type: point_estimate [✓ learning_curve]; Hscale: 1.3 Vscale: 1.3

Algorithm / Dataset: get_svm_accuracy / Adult
Result / Error: 89.1425 / 0.0108164142552999

Parameters: n 10; compute_method: [✓ estimate_error] compute_result; run_on_hadoop: on [✓ off]; display_type: [✓ point_estimate] learning_curve; Hscale: 1.3 Vscale: 1.3

Fig. 1. An example of EARL estimating the classification error for the Adult training dataset using SVM. (left) The user begins by starting with a small sample and analyzing the learning curve then, (right) the user picks the sample size for the desired error bound and computes the result. In the case of SVM, the resulting model is saved in a binary file, which can be then loaded in R and used on testing data.

needed to achieve the desired accuracy. Estimations made by extrapolating the learning curve allow users to estimate errors and results for sizes of data that make the actual computation impractical under realistic time and resource constraints. Finally, in (iii) the algorithm is executed (on Hadoop or in local mode) over the determined sample size. Steps (ii) and (iii) are shown in Figure 1. For more details on the above steps refer to our full paper in [9].

EARL presents many exciting opportunities. While the recent move to a DSMS might require that the old data mining algorithms be replaced by new ones specifically designed for data streams, our recent results suggest that this is not often the case. In fact, recent results on early approximate answers from massive data sets [9] show that accurate mining results can be obtained with samples as low as 1%, whereby satisfactory response times are often achievable on massive data streams by using the same algorithms as those used on stored data.

Organization Section II briefly describes the demonstration we are proposing. Section III provides an overview of our system. Section IV presents related work and Section V draws our conclusion.

II. Demonstration Description

To demonstrate the functionalities of the EARL workbench, we apply it to a well known supervised learning algorithm SVM [7], which is known for its accurate classification but expensive running time. We then proceed with the MIC [15] algorithm, which has been recently proposed for detecting associations in large data. The actual demonstration is organized in two phases: (i) a brief introduction to the main system functionalities, in which the presenter shows a simple case of our scalable error estimation framework highlighting the core components of the system, and (ii) a "hands-on" phase in which the public is invited to *directly* interact with the system and test its capabilities by visually inspecting the results produced by the mining algorithms derived from samples of the data.

In the presentation of the system functionalities we will show how the system interface, shown in Figure 1 guides the user through the the two-step analytics process: (i) Learning curve prediction and (ii) Point-error estimation. In (i) for the particular mining task and the given sample of the training data, the user analyzes the predicted learning curve and picks the point on the curve that is satisfactory to her error threshold. In (ii), once the desired sample size has been determined from the learning curve, the user uses the point-prediction to compute the result, at the same time displaying the final accuracy.

In the second part of the demo, the conference participant will experience EARL "hands-on" by adjusting its various configuration parameters. Using EARL's configuration, the user is able to select the target mining algorithm, and the training data. The "hands-on" session will consist of experimenting with EARL using two very different mining algorithms: (i) SVM and (ii) MIC. In (i) the participant will be able to choose from a variety of training datasets ranging from our private twitter hashtag dataset to the well-known UCI-Machine Learning repositories [2] to train the SVM model to predict various features of the datasets on sample of the data given the desired error threshold. After training the model, the demo participant will load the computed model into *R* and visually inspect it using the testing data. In (ii) we will use the MIC algorithm to compute an approximate hashtag association score between all tags in our Twitter dataset which contains over 2 Million hashtags. For the MIC demo, we will precompute a set of MIC scores M using EARL for various sample sizes and various error bounds. We will provide to the demo participant a web interface where she can search for a hashtag and see the strongly correlated hashtags along with the error bound hierarchy for the various M.

Therefore, our demonstration will present a major step forward in solving a very important problem in data-mining: *How to deliver accurate early results for a variety of mining algorithms without making any assumptions about the underlying data distribution*. By quickly providing the error bound to the user, the workflow becomes interactive giving the user more control over the exploration of her data.

III. Overview of EARL

This section briefly summarizes the theoretical background of EARL presented in [9]. The key concept in EARL is the one of *efficient* and *scalable* bootstrapping which is used for error estimation. By *efficient* we mean that our approach uses delta maintenance techniques to avoid recomputing results when enlarging the sample size. By *scalable* we mean that EARL provides a way to run the error estimation computation over Hadoop utilizing the inherent parallelizability of bootstrapping. Furthermore EARL provides a simple API which the developer can use to specify her own mining algorithm and let EARL take care of the mechanics of providing the error estimates.

A. Estimating Accuracy

In EARL, error estimation of an arbitrary function is done via resampling. By re-computing a function of interest many times, a result distribution is derived from which both the approximate answer and the corresponding error are retrieved. EARL uses a clever delta maintenance strategy that dramatically decreases the overhead of computation. As a measurement of error EARL uses a coefficient of variation (c_v) which is a ratio between the standard deviation and the mean. Our approach is independent of the error measure and is applicable to other errors (e.g., bias, variance).

A crucial step in statistical analysis is to use the given data to estimate the accuracy measure, such as the bias, of a given statistic. In a traditional approach, the accuracy measure is computed via an empirical analog of the explicit theoretical formula derived from a postulated model [16]. Using variance as an illustration let $X_1, ..., X_n$ denote the data set of n independent and identically distributed (i.i.d) data-items and let $f_n(X_1, ..., X_n)$ be the function of interest we want to compute. The variance of f_n is then:

$$var(f_n) = \int \left[f_n(x) - \int f_n(y) d \prod_{i=1}^{n} F(y_i) \right]^2 d \prod_{i=1}^{n} F(x_i) \tag{1}$$

where $x = (x_1, ..., x_n)$ and $y = (y_1, ..., y_n)$. Given a simple f_n we can obtain an equation of $var(f_n)$ as a function of some unknown quantities and then substitute the estimates of the unknown quantities to estimate the $var(f_n)$. In the case of the sample mean, where $f_n = \bar{X}_n = n^{-1} \sum_{i=1}^{n} X_i$, $var(\bar{X}_n) = n^{-1} var(X_1)$. We can therefore estimate $var(\bar{X}_n)$ by estimating $var(X_1)$ which is usually estimated by the sample variance $(n-1)^{-1} \sum_{i=1}^{n} \left(X_i - \bar{X}_n \right)^2$. Unfortunately, the use of Equation 1 to estimate the variance is computationally feasible only for simple functions, such as the mean. Next we discuss a resampling method used to estimate the variance of arbitrary functions.

The *bootstrap* resampling approach [17] provides an accuracy estimation for general functions. This approach does not require a theoretical formula to produce the error estimate of a function. In fact the bootstrap technique allows for estimation of the sampling distribution of almost any statistic using only very simple methods [5]. The estimate of the variance can then be determined from the sampling distribution. The bootstrap technique, however, requires repeated computation of the function of interest on different resamples. The estimate of the variance of the result, σ, produced by this repeated computation is $\sigma^2(F) = E_F(\hat{\theta} - E_F(\hat{\theta}))^2$, where θ is the parameter of interest. The challenge lies in making the bootstrap scalable and efficient.

B. Delta Maintenance when Bootstrapping

The most computationally intensive part of *EARL*, aside from the user's job j, is the re-execution of j on an increasingly larger sample sizes. One important observation is that this intensive computation can reuse its results from the previous iterations. By utilizing this incremental processing, performing large-scale computations can be dramatically improved. Our full paper [9] takes a more detailed look at the processing of two consecutive bootstrap iterations and also discusses the optimization of the bootstrapping (resampling) procedure so that when recomputing f on a new resample s' we can perform delta maintenance using a previous resample s.

C. The Learning Curve

One of the exciting parts of our work is extrapolating the learning curve, which captures the relationship between the size of the sample and the result accuracy. The study of learning curves is important to us because learning curves can be used to estimate how large n must be before the error drops below a user specified level τ. Thus, we are interested in the asymptotic complexity as n becomes large. Vapnik-Chervonenkis (VC) theory [19] states that a random sample of size n leads to a generalization error $O(\frac{d}{n})$ of a function class F where d is a measure of the complexity of F. VC theory presents universal bounds that are distribution independent. In our case, through empirical support, we are able to provide a much tighter empirical estimates of the accuracy error than those given by the VC theory.

The learning curve follows a power-law [10] and thus we estimate the parameters of a learning curve by subsampling and extrapolating. Our method parametrizes the learning curve as an inverse function of the power law: $\sigma(n) = a + m^{-\alpha}$. The unknown parameters of this expression, $a \in R, b, \alpha \geq 0$, are estimated using nonlinear least squares, which estimates the parameters via minimization. The estimation of the parameters is done by evaluating $\sigma(n_i)$ for various i exactly.

D. Sampling over HDFS

In order to provide a uniformly random subset of the original data-set, *EARL* requires sampling. While sampling over memory-resident, and even disk resident, data had been studied extensively, sampling over a distributed file system, such as HDFS, has not been addressed [11]. Therefore EARL provides two sampling techniques: (1) pre-map sampling and (2) post-map sampling. The two sampling techniques allow for sampling before or after sending the input to the Mapper denoted as the *pre-map* and *post-map* sampling respectively. Pre-map sampling significantly reduces the load times, however the sample produced may be an inaccurate representation of the total $< k, v >$ pairs present in the input. Post-map sampling first reads the data and then outputs a uniformly random sample of desired size. Post-map sampling also avoids the problem of inaccurate $< k, v >$ counts. More details on these sampling approaches can be found in [9].

IV. State of the Art

To achieve scalability, EARL uses Hadoop for error estimation. Hadoop, and more generally the MapReduce framework, was originally designed as a batch-oriented system, however it is often used in an interactive setting where a user waits for her job to complete before proceeding with the next step in the data-analysis workflow. With the introduction of high-level languages such as Pig [12], Sawzall [14] and Hive [18], this trend had accelerated. Due to its batch oriented computation mode, traditional Hadoop provides poor support for interactive analysis. To overcome this limitation, Hadoop Online Prototype (HOP) [3] introduces a pipelined Hadoop variation in which a user is able to refine results interactively. In HOP however, the user is left with the responsibility of devising and implementing the accuracy estimation and improvement protocols.

Sampling techniques for Hadoop were studied previously in [6] where authors introduce an approach of providing Hadoop job with an incrementally larger sample size. The authors propose an *Input Provider* which provides the *JobClient* with the subset of the input splits. The assumption that each of the splits represents a random sample of the data, however, is not well justified. Furthermore the authors do not provide any functionality for error estimation.

Authors in [13] present a method of early returns for Hadoop jobs over big data. The authors, however, only focus on providing the early answer when the user error criteria is met. In our work we are focused on providing the error based on a given sample size. The information about the required sample size can then be used for future workflows. Furthermore EARL provides the learning curve which can often predict the required sample size for a given error bound without having to carry out the actual computation.

V. Conclusion

A key part of big data analytics is the need to collect, maintain and analyze enormous amounts of data efficiently. To address these needs, frameworks based on MapReduce are used for processing large data-sets using a cluster of machines. Current systems, however, are not able to provide accurate estimate of incremental results and are mostly suited for batch processing. This demo presents *EARL* which is a simple framework for estimating results and errors for mining algorithms. A surprising finding was that it is seldom necessary to use the whole dataset, and in most cases it is sufficient to use 1% of data to achieve similar results compared to the execution over the entire dataset. We made various optimizations to the resampling methods which makes the framework more attractive. The learning curve proved effective in planning the mining task by determining with confidence the point after which no noticeable improvement in error is seen. We are now working on extending the presented approximation approach to other classification algorithms and other data mining tasks, which suggests a future direction of our research.

References

[1] Earl release website: http://yellowstone.cs.ucla.edu/wis/.

[2] D.J. Newman A. Asuncion. UCI machine learning repository, 2007.

[3] Tyson Condie, Neil Conway, Peter Alvaro, Joseph M. Hellerstein, Khaled Elmeleegy, and Russell Sears. Mapreduce online. Technical Report UCB/EECS-2009-136, EECS Department, University of California, Berkeley, Oct 2009.

[4] B Efron. Bootstrap methods: Another look at the jackknife. *Annals of Statistics*, 7(1):1–26, 1979.

[5] Bradley Efron. Better bootstrap confidence intervals. *Journal of the American Statistical Association*, 82(397):171–185, 1987.

[6] Raman Grover and Michael Carey. Extending map-reduce for efficient predicate-based sampling. ICDE '12, 2012.

[7] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning, Second Edition: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, 2nd ed. 2009. corr. 3rd printing 5th printing. edition, February 2009.

[8] Herodotos Herodotou, Harold Lim, Gang Luo, Nedyalko Borisov, Liang Dong, Fatma Bilgen Cetin, and Shivnath Babu. Starfish: A self-tuning system for big data analytics. In *CIDR*, pages 261–272, 2011.

[9] Nikolay Laptev, Kai Zeng, and Carlo Zaniolo. Early accurate results for advanced analytics on mapreduce. *PVLDB*, 5(10):1028–1039, 2012.

[10] Sayan Mukherjee, Pablo Tamayo, Simon Rogers, Ryan M. Rifkin, Anna Engle, Colin Campbell, Todd R. Golub, and Jill P. Mesirov. Estimating dataset size requirements for classifying dna microarray data. *Journal of Computational Biology*, pages 119–142, 2003.

[11] Frank Olken and Doron Rotem. Random sampling from database files: A survey. In *SSDBM*, pages 92–111, 1990.

[12] Christopher Olston, Benjamin Reed, Utkarsh Srivastava, Ravi Kumar, and Andrew Tomkins. Pig latin: a not-so-foreign language for data processing. SIGMOD, pages 1099–1110, New York, NY, USA, 2008. ACM.

[13] Niketan Pansare, Vinayak R. Borkar, Chris Jermaine, and Tyson Condie. Online aggregation for large mapreduce jobs. *PVLDB*, 4(11):1135–1145, 2011.

[14] Rob Pike, Sean Dorward, Robert Griesemer, Sean Quinlan, and Google Inc. Interpreting the data: Parallel analysis with sawzall. In *Scientific Programming Journal, Special Issue on Grids and Worldwide Computing Programming Models and Infrastructure*, pages 227–298.

[15] David N. Reshef, Yakir A. Reshef, Hilary K. Finucane, Sharon R. Grossman, Gilean McVean, Peter J. Turnbaugh, Eric S. Lander, Michael Mitzenmacher, and Pardis C. Sabeti. Detecting novel associations in large data sets. *Science*, 334(6062):1518–1524, 2011.

[16] Jun Shao and D. Tu. *The jackknife and bootstrap*. Springer Verlag, 1995.

[17] Alun Thomas. Bootstrapping, jackknifing and cross validation. reusing your data. 2000.

[18] Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Suresh Anthony, Hao Liu, Pete Wyckoff, and Raghotham Murthy. Hive- a warehousing solution over a map-reduce framework. In *VLDB*, pages 1626–1629, 2009.

[19] V N Vapnik and A Y Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications*, 16(2):264–280, 1971.