BOOT-TS: A Scalable Bootstrap for Massive Time-Series Data

Nikolay Laptev[*], **Carlo Zaniolo**
UCLA
Los Angeles, CA
{nlaptev,zaniolo}@cs.ucla.edu

Tsai-Ching Lu
HRL
Malibu, CA
tlu@hrl.com

Abstract

We propose a scalable method of assessing the quality of machine learning algorithms over sampled time-series data. While bootstrap provides a simple and powerful means of estimating accuracy, its application to large time-series data still suffers from scalability issues. As an alternative we introduce BOOT-TS, a scalable extension of bootstrap for time-series which utilizes the recent advances in bootstrap and time-series theory to provide a practical implementation for assessing a time-series sample quality using Hadoop. For instance, our new procedure yields a robust and computationally efficient means of assessing the quality of our Twitter analytics workflow over large, real-world, time-series data.

1 Introduction

A large portion of today's data is in a time-series format (e.g., Twitter stream, system logs, blog posts), and time constraints as well as monetary constraints force the user to work on a sample of the data, for which the quality assessment is required. Furthermore the trend of dataset growth is accelerating, with 'big data' becoming increasingly prevalent. The original bootstrap approach [11, 2] and its time-series variants [1, 9] provide a simple way of assessing the quality of an estimate, however in the distributed environment the cost of transferring the data to independent processors as well as the cost of computing a single resample can be high. This is the source of incessant problems in the natural life-cycle of advanced analytics development and a major stumbling block that prevents interactive result exploration that many users desire [6, 7]. Thus we need a scalable and efficient way of assessing the quality of an estimator for large time-series data samples in a distributed environment.

In this paper we propose a simple, accurate and scalable technique for assessing the quality of results computed from samples of the real-world time-series data. This technique allows the average user or a small business to derive analytics on the data of massive sizes by using sampling. Therefore our method will go towards allowing everyone to gain access to and benefit from knowledge discovery.

Reliable estimation of the quality obtained from sampled data often leads to sample sizes that are orders of magnitude smaller than the original dataset thus providing major savings in terms of time and money [6]. The average user, when debugging her machine learning algorithm locally can work with a sample of the data while remaining confident in the expected accuracy of her final result. When the same user deploys her application on the cloud system such as EC2[1] money and time can also be saved by only working with a sample of the data and using only the needed resources[2].

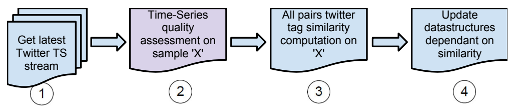
This paper addresses the scalability issue of error estimation for sampled time-series data. We build on previous work of bootstrap [6, 5, 4] and time-series theory [1, 9] to provide a scalable approach for assessing the quality of large time-series data samples. We take advantage of the trend in computational resources which is shifting towards a multicore and distributed architecture with cloud services providing thousands processing units[3]. These new processing units support fast

[*]This work was done while the author was at HRL.

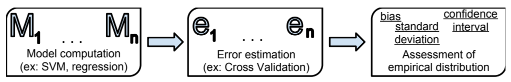
[1]http://aws.amazon.com/ec2/

[2]The pricing of cloud services is based on (a) storage and (b) processing unit usage.

[3]The total number of processing units in a cluster is: number of nodes $\times$ the number of CPUs per node.



(a) A 40,000 ft. view of Twitter hashtag association computation.



(b) Details on ②. Bias, s.d. and variance are among the many statistical error measures that can be used.

Figure 1: One of our analytics workflows over time-series data.

execution of 'big data' applications. Motivated by this trend, we introduce the BOOT-TS system that integrates recent research in both bootstrap and time-series theory. The core of our approach relies on the Bag of Little Bootstraps (BLB) [5] for bootstrap scalability, on the Stationary Bootstrap (SB) [9] for time-series resampling support and on Hadoop[4] for computational scalability. BOOT-TS works by first selecting in parallel a random block b of the original time-series data sample of size n. Then in parallel r resamples are generated by applying SB to each block b to obtain a resample of size n. In our implementation, r is not fixed, and each node can converge independently which was a limitation in [5]. The errors computed on the resamples are then averaged across all nodes to produce the final accuracy estimate. Thus, in BOOT-TS, the estimator is applied on only a single block b instead of on the entire time-series sample. In other words, the computational cost incurred only depends on the size of b and not on the size of the resample n.

The difficulty in supporting fast and accurate estimator assessment based on a sample has long been recognized as one of the major obstacles in interactive big data processing [5, 3]. The problem is more complicated when the data are a time-series because bootstrap must be carried out in a way that captures the dependence structure of the data generation process. As a concrete application of BOOT-TS we use one of our own Twitter analytics workflows which is composed of four stages: ① Get latest Twitter data of size $W \rightarrow$ ② Estimate error based on a sample n of $W \rightarrow$ ③ Compute the similarity between Twitter hashtags[5] from $|s| \rightarrow$ ④ Update internal tag similarity model. Note that we compute similarity across more than 2 Million Twitter tags which further motivates this research. Step ② is computed infrequently, only to estimate the sample size required to achieve a user defined accuracy. The goal of this workflow is to compute a similarity score of all hashtag pairs based on a state of the art algorithm called MIC [10]. While all stages of this workflow provide interesting challenges, this paper is specifically on step ②. Figure 1 further elucidates this workflow and emphasizes the focus of this paper.

To the best of our knowledge, this work is the first to provide scalable bootstrap with favorable statistical guarantees. Our preliminary experiments show a promising trend when applying BOOT-TS to the analytics workflow shown in Figure 1 and to other ML algorithms such as classification and regression.

2 Preliminaries

Let $\{X_i\}_{i=-\infty}^{\infty}$ be an $\mathbb{R}^d$-valued stationary process observed from some underlying distribution $P \in P$ with $\mathbb{P}_n = n^{-1}\sum_{i=1}^{n}\delta_{X_i}$ denoting the corresponding empirical distribution. Let $\mathbf{X_n} = \{X_1, ..., X_n\}$ denote the available observations (usually drawn iid from $\{X_i\}$). Based on $\mathbf{X_n}$ we compute an estimator $\hat{\theta}_n$ of the parameter of interest θ. $\hat{\theta}_n$ may for example estimate the accuracy of a classifier. Borrowing notation from [5] we define $Q_n\{P\} \in \mathbb{Q}$ as the true distribution of $\hat{\theta}_n$. Thus our final goal is to estimate $\xi(Q_n\{P\}, P)$. In our case ξ denotes the coefficient of variation but it can also denote bias, standard error, confidence region or a quantile. Using bootstrap with Monte Carlo approximation [2] one can repeatedly resample n points iid. from $\mathbb{P}_n$ from which an empirical distribution $\mathbb{Q}_n$ is formed and from which $\xi(Q_n\{P\}, P) \approx \xi(\mathbb{Q}_n)$ is approximated. For block bootstrapping, let the expected size of the block be l which is an integer satisfying $1 < l < n$. In this work we concentrate on the Stationary Block bootstrap (SB) [9], which functions as follows: we let $L_{ni} \equiv L_i, i \geq 1$ be conditionally iid random variables with parameter $p = l^{-1} \in (0, 1)$. Also let $\mathcal{I}_1, ..., \mathcal{I}_n$ be conditionally iid random variables with the discrete uniform distribution on $\{1, ..., n\}$. Then, the SB resample $X_1^*, ..., X_n^*$ is produced from the first n elements in the array $\mathbb{B}(\mathcal{I}, L_1), ..., \mathbb{B}(\mathcal{I}, L_\mathbb{K})$ where $\mathbb{K} \equiv inf\{\kappa \geq 1 : L_1 + ... + L_\kappa \geq n\}$ and where the block $\mathbb{B}(i, \kappa) = (X_i, ..., X_{i+\kappa-1}), i \geq 1, \kappa \geq 1$. The key difference of BOOT-TS from [9] and from [5] is that we focus on scalability when extending the bootstrap to time-series.

[4]hadoop.apache.org

[5]The # symbol, called a hashtag, is used to mark keywords or topics in a Tweet.

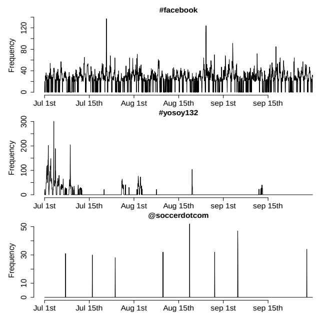
3 BOOT-TS: The Scalable Time-Series Resampling Algorithm

The Naïve approach: Predicting the quality of an estimate over time-series can be done in an embarrassingly parallel way using bootstrap variations for time-series [8, 1, 9]. We would send the time-series data to all nodes on which block sampling is applied. Since we only need to produce a certain number of resamples, the computation time simply scales down with the number of nodes. This naïve approach is inefficient because it sends the entire dataset to all machines, which can still be impractical for large datasets, but authors in [5] address this concern for iid data by splitting the sample into chunks and processing each chunk in parallel thus avoiding sending the whole dataset to all nodes. The limitations of the solution proposed in [5] is that (i) it is only applicable to iid data and (ii) it treats all block samples as equal. In practice however, some nodes may get a bad block sample, therefore more resamples will be required for convergence, whereas some nodes may converge quicker due to a good block sample. This means that setting a fixed r, as is done in [5], may lead to inefficiencies. To solve the above problems, we propose Algorithm 1, below.

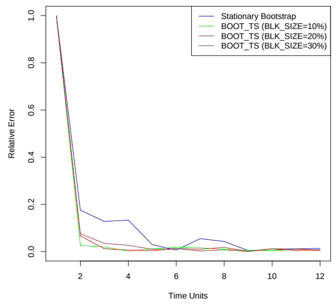
BOOT-TS Algorithm: Algorithm 1 presents the BOOT-TS algorithm. The algorithm works by first selecting a random block b from a set of possible blocks $\mathbb{B}(\mathcal{I}, L_1), ..., \mathbb{B}(\mathcal{I}, L_{\mathbb{K}})$ observed in the time-series sample. The stationary bootstrap is then applied to each block by appending the next point (wrapping around if necessary) in the series to the current block with probability $1-p$, and appending a random point from b with probability p. Note that although other time-series approaches are applicable, we focus on SB due to its popularity and its robustness with respect to block size [9]. Then the error $\xi(Q_n(\mathbb{P}^{(j)}_{n,b}))$ for each block b is estimated by applying the Monte Carlo approximation (standard bootstrap) via repeatedly resampling the blocks using the SB method and computing the estimates on each resample. The number of resamples r is locally determined by each node with $O(r) \approx 100$. Local convergence is important because two nodes, having subsamples of different quality, should not have the same r. Thus by locally determining r for each s, BOOT-TS turns out to be less sensitive to the quality and to the size of b than SB. More details on local convergence are given in Section 4. Note that if a local node converged after m resamples, BOOT-TS must append $r-m$ resample estimates to $\hat{\theta}^*_n$ which is done by drawing $r-m$ resample estimates iid from $\hat{\theta}^*_m$. Note that s is dependent on b and at convergence can be as low as 1-5 for $b = n^{0.9}$ or 10-20 for $b = n^{0.5}$ which is consistent with results for iid data found in [5]. The r resample estimates then form an empirical distribution which is used to approximate $\xi(Q_n(\mathbb{P}^{(j)}_{n,b})) \approx \xi(\mathbb{Q}_{n,j})$. The quality estimates are then averaged across all the nodes and final error is returned as $s^{-1}\sum_{j=1}^{s}\xi^*_{n,j}$.

Algorithm 1: BOOT-TS

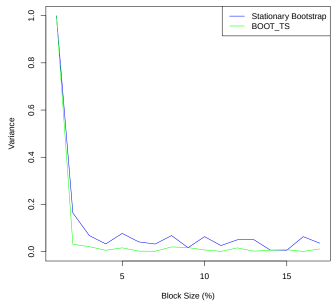
```
Input: Time-series data sample $X_1, \ldots, X_n$        $b$: Selected subsample
       $\hat{\theta}$: Estimator of interest                 $s$: Number of sampled blocks
       $\xi$: Estimator quality assessment           $r$: Maximum number of Monte Carlo iterations
Output: An estimate of $\xi(Q_n(P))$
for $j \leftarrow 1$ to $s$ do
    Randomly select a block $b$ from $\mathbb{B}(\mathcal{I}, L_1), ..., \mathbb{B}(\mathcal{I}, L_{\mathbb{K}})$
    for $k \leftarrow 1$ to $r$ do
        while Size of sample $\leq n$ do
            if $rgeom(1, p) == 0$ then
                currentIndex $\leftarrow$ A random index in $b$
                pick the next item as $b[currentIndex]$ and continue
            end
            pick the next item in our sample as $b[currentIndex$++]
        end
        $\mathbb{P}^*_{n,k} \leftarrow n^{-1}\sum_{a=1}^{b} n_a \delta_{X_{i_a}}$
        $\hat{\theta}^*_{n,k} \leftarrow \hat{\theta}(\mathbb{P}^*_{n,k})$
        check if $\hat{\theta}^*_{n,k}$ converged and if so, append $(r-k)$ estimates
        to $\hat{\theta}^*_{n,k}$ drawn iid from $\hat{\theta}^*_{n,k}$ and break
    end
    $\mathbb{Q}^*_{n,j} \leftarrow r^{-1}\sum_{k=1}^{r}\delta_{\hat{\theta}^*_{n,k}}$
    $\xi^*_{n,j} \leftarrow \xi(\mathbb{Q}^*_{n,j})$
end
return $s^{-1}\sum_{j=1}^{s}\xi^*_{n,j}$
```



(a) Sample time-series of several hashtags.



(b) Convergence rate of BOOT-TS is quicker than that of SB.



(c) BOOT-TS is less dependent on block size than SB.

Figure 2

4 Implementation and Experiments

Implementation: The performance of the current implementation of BOOT-TS relies on the two design decisions of adopting: (1) Model-error computation separation and (2) Convergence aware computation. The first decision proved invaluable for achieving a simple API design that is easy to understand and test. In model-error computation separation the error computation is performed in three stages: (a) Model computation (b) Model Validation and (c) Error derivation from empirical distribution of (b). It is important to realize that a lot of machine learning algorithms fall under this three-stage model, and the quality assessment of many classic machine learning algorithms (SVM, regression) can easily be expressed using it. Decision (2) is also important because in providing block samples, not every block will be of the same 'quality', error computation on some nodes may actually converge faster; therefore in our implementation r in Algorithm 1 is actually used as an upper-bound, and each node instead uses a threshold for local convergence checking. Note however that all nodes produce the same number of error estimates because we enlarge $|\hat{\theta}^*_n|$ to r via iid resampling if necessary. Our full paper will consist of more detailed experiments and justification of this dynamic setting of r.

Experimental Results: All experiments were performed on a 7-node cluster with 140 map-slots on Hadoop 1.0.4. Each node is a 16-core AMD Opteron Processor 6134 @ 2.3GhZ with 132GB RAM. The Total Twitter Dataset is of size 4.3TB.

In Figure 2a we show the type of time-series BOOT-TS was designed to deal with. Figure 2a shows only three time-series for three tags over a period of three months. Our experiments in production, however, were run on two million tags, with the goal of determining similarity measure across all $\frac{(n-1)(n)}{2}$ hashtag pairs. Note that some tags may have many random spikes (e.g., #riot), some tags may be fairly stable (e.g., #youtube) and some may show very little level of activity (e.g., #mycatsnameisboris). In Figure 2b we show the convergence rate of BOOT-TS and of Stationary Bootstrap. BOOT-TS succeeds in converging to a low relative error significantly faster than the Stationary Bootstrap. Although here we only show convergence results for similarity computation, we observed similar results for other machine learning tasks such as SVM and linear regression and we plan on sharing these results in our full paper. In Figure 2c we show the block size dependence on the estimated variance of the Stationary Bootstrap and of the BOOT-TS. Notice that the variance of the BOOT-TS is much more stable than that of the Stationary Bootstrap and therefore in practice the user should be much less concerned with picking the perfect block size.

5 Future Work

We have discussed BOOT-TS and shown that it is a promising approach to scalable time-series sample quality assessment. We have shown preliminary results of its superiority to the Stationary Bootstrap and found that using BOOT-TS we were able to discover some interesting patterns in our dataset, that would otherwise be impossible due to an impractically large dataset. Our work on BOOT-TS will continue in the direction of providing more rigorous statistical guarantees of BOOT-TS' quality assessment. We will also analyze not only the local convergence based on r but also based on n, or the size of the resample, in order to achieve an even faster convergence rates in a distributed system.

Acknowledgements: Supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center (DoI / NBC) contract number D12PC00285. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBE, or the U.S. Government.

References

[1] P. Bhlmann and H. R. Knsch. Block length selection in the bootstrap for time series. *Computational Statistics And Data Analysis*, 31(3):295 – 310, 1999.

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

[3] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.-H. Bae, J. Qiu, and G. Fox. Twister: a runtime for iterative mapreduce. HPDC, pages 810–818, 2010.

[4] A. Kleiner, A. Talwalkar, P. Sarkar, and M. Jordan. Bootstrapping big data. NIPS, 2011.

[5] A. Kleiner, A. Talwalkar, P. Sarkar, and M. I. Jordan. A scalable bootstrap for massive data. Technical report, UC Berkeley, 2012.

[6] N. Laptev, K. Zeng, and C. Zaniolo. Early accurate results for advanced analytics on mapreduce. *PVLDB*, 5(10):1028–1039, 2012.

[7] J. Lin and A. Kolcz. Large-scale machine learning at twitter. SIGMOD, pages 793–804, 2012.

[8] S. U. D. of Statistics, D. Politis, J. Romano, and N. S. F. (U.S.). *A Circular Block-resampling Procedure for Stationary Data*. 1991.

[9] D. N. Politis and J. P. Romano. The Stationary Bootstrap. *Journal of the American Statistical Association*, 89(428):1303+, Dec. 1994.

[10] D. N. Reshef, Y. A. Reshef, H. K. Finucane, S. R. Grossman, G. McVean, P. J. Turnbaugh, E. S. Lander, M. Mitzenmacher, and P. C. Sabeti. Detecting novel associations in large data sets. *Science*, 334(6062):1518–1524, 2011.

[11] J. Shao and D. Tu. *The jackknife and bootstrap*. Springer series in statistics. 1995.