Efficient Bootstrapping in Python for Estimating Multimedia Classification Performance

Peter Birsinger
UC Berkeley
peterbir@eecs.berkeley.edu

Benjamin Elizalde
UC Berkeley
benmael@icsi

Gerald Friedland
UC Berkeley
fractor@icsi

Armando Fox
UC Berkeley
fox@cs.berkeley.edu

ABSTRACT

Determing the true accuracy of a multimedia classifier often requires estimating performance on unseen test sets based on the performance on a known data set. This can determine whether a training set amply informs a classifier of the data distribution, or whether high accuracy on a test set is more than a fluke. Bootstrapping, a statistical method to quantify uncertainty, uncovers the uncertainty of a classifier's performance, yet often demands considerable computational resources, a problem compounded by today's sizable multimedia datasets (citation?). Large bootstrapping multimedia applications can become essentially intractable in common productivity languages such as Python and Matlab, as shown later in the paper.

We propose a methodology to estimate classifier performance that relies on the ability to generate efficient, distributed bootstrapping applications from serial Python. This methodology arises from SEJITS (Selective Embedded Just-In-Time Specialization) [8], an approach to convert programs written in domain-specific languages (DSELs) to programs written in languages suitable for high performance or parallel computation. Utilizing an already made DSEL compiler [1] for a recently developed bootstrapping algorithm, we generate scalable, bootstrapping code suitable for the Spark [12] cluster computing system. This enables us to code in native Python adhering to the predefined DSEL a bootstrapping application that estimates the equal error rate (EER) of a multimedia classifier; the generated code executes in a cluster, garnering orders of magnitude speedup over the naive Python code. We then explore the effects of varying the number of positives in both training and test sets obtained from the TRECVID MED 2012 dataset, surveying the flavor of computations now made accessible to non-performance programmers.

Keywords

Multimedia Classification; bootstrapping; SEJITS

1. INTRODUCTION

Estimating a classifier's performance on unseen data sets based on its performance on a known data set can reveal useful information about the training and test sets. A large confidence interval on the accuracy based on the training data can indicate that the actual accuracy on a test set may deviate wildly from the accuracy on the training set whereas a low standard deviation on the accuracy based on the test data can indicate that the accuracy on other, similar test sets from the same distribution will likely be comparable.

Fortunately, classifier performance speculation is a well understood problem [5]. Extensive investigation of bootstrapping and its variations [3, 7, 2, 11] has identified bootstrapping as a prime, ableit not perfect [6], solution to this problem. We shift the focus in this paper, however, onto using bootstrapping to estimate the performance of specifically multimedia classifiers.

When dealing with multimedia data, particularly given the increasing size of modern datasets such as the TRECVID MED 2012 dataset, efficiency becomes paramount. We turn to a recently developed bootstrapping method, the Bag of Little Bootstraps (BLB)[9, 10], that is designed to run quickly in a distributed setting. An already existing DSEL compiler converts input Python BLB applications into scalable BLB applications able to run on the Spark cluster computing system [1]. This DSEL compiler makes use of Selective Embedded Just-In-Time Specialization (SEJITS) [8], an approach for converting DSELs in productivity level languages to high performance efficiency language (e.g. C++ or Cilk) code.

Here, we describe in this already existing Python DSEL a BLB application which estimates the standard deviation of the equal error rate (EER) of a multimedia classifier. We generate scalable code to run on a cluster to probe the effects of varying training and test set sizes, with data sets from the TRECVID MED 2012 datasest. We investigate the relationship between the training set sizes of these consumer-produced videos and the error estimation on test sets in order to reasonably estimate the necessary minimum size of training sets. We similarly vary the test size to attempt to answer how many consumer-produced videos from this set

are needed to provide a reasonable measure of significance to a classification result.

These experiments characterize our proposed methodology of relying on a DSEL compiler to generate efficient bootstrapping multimedia classification applications. This methodology now makes accessible an array of multimedia classifier estimation computations for those who never wish to leave the world of Python but previously were forced to, as confirmed by our section discussing the results of naive Python. Moreover, this methodology templates the mechanism through which further useful, multimedia-machine-learning-related algorithms can be brought to non-performance programmers.

2. METHODOLOGY DESCRIPTION

brief description of blb/bootstrapping

brief description of sejits

brief description of BLB cloud specializer

speedup values over naive python ?

why makes sense to use this ?

3. EXPERIMENTS VARYING TEST SET SIZE

important to determine significance of test result if fluke or not

some test sets can easily be reduced in size to reduce computation time and still have roughly same significance

4. EXPERIMENTS VARYING TRAINING SET SIZE

experiments with varying number of positives in training set for each event... then run blb on classifier ON training set, and then this standard deviation will give indication of how much can vary on test set ...

claims probably help to support: [4]

5. COMPARISON TO NAIVE PYTHON

6. FUTURE WORK

7. CONCLUSION

made accessible to multimedia classification domain experts efficient means to estimate multimedia classifier performance

testament to its efficiency all experiments and things we were able to look at example python to

other bootstrapping apps easily coded in python to estimate other multimedia computations

8. ACKNOWLEDGEMENTS

9. REFERENCES

[1] P. Birsinger, R. Xia, and A. Fox. Scalable bootstrapping for python. *ACM International Conference on Information and Knowledge Management*, 2013.

[2] M. Chernick, V. Murthy, and C. Nealy. Application of bootstrap and other resampling techniques: evaluation of classifier performance. *Pattern Recognition Letters*, 3(3):167–178, 1985.

[3] B. Efron. Bootstrap methods: another look at the jackknife. *The annals of Statistics*, pages 1–26, 1979.

[4] B. Elizalde, G. Friedland, H. Lei, and A. Divakaran. There is no data like less data: percepts for video concept detection on consumer-produced media. In *Proceedings of the 2012 ACM international workshop on Audio and multimedia methods for large-scale video analysis*, pages 27–32. ACM, 2012.

[5] K. Fukunaga and R. R. Hayes. Estimation of classifier performance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11(10):1087–1101, 1989.

[6] A. Isaksson, M. Wallman, H. Göransson, and M. G. Gustafsson. Cross-validation and bootstrapping are unreliable in small sample classification. *Pattern Recognition Letters*, 29(14):1960–1965, 2008.

[7] A. K. Jain, R. C. Dubes, and C.-C. Chen. Bootstrap techniques for error estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (5):628–633, 1987.

[8] S. A. Kamil. *Productive High Performance Parallel Programming with Auto-tuned Domain-Specific Embedded Languages*. PhD thesis, EECS Department, University of California, Berkeley, Jan 2013.

[9] A. Kleiner, A. Talwalkar, P. Sarkar, and M. Jordan. The big data bootstrap. *arXiv preprint arXiv:1206.6415*, 2012.

[10] A. Kleiner, A. Talwalkar, P. Sarkar, and M. I. Jordan. A scalable bootstrap for massive data. *arXiv preprint arXiv:1112.5016*, 2011.

[11] B. Sahiner, H.-P. Chan, and L. Hadjiiski. Classifier performance prediction for computer-aided diagnosis using a limited dataset. *Medical Physics*, 35:1559, 2008.

[12] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: cluster computing with working sets. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, pages 10–10, 2010.