Efficient Bootstrapping in Python for Estimating Multimedia Classifier 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

When developing a multimedia classification system in preparation for an evaluation such as TRECVID MED or MediaEval, one typically must estimate the error on the evaluation data set from the performance on the development data set. Statistical bootsrapping reveals the uncertainty of the estimate of a classifier's performance, yet often requires considerable computational resources, a problem compounded by today's sizable multimedia datasets. Large bootstrapping multimedia applications, particularly when written in common productivity languages such as Python or Matlab, can become essentially intractable.

In this paper, we propose a methodology to estimate classifier performance that relies on the ability to generate efficient, distributed bootstrapping applications from serial Python. The methodology arises from SEJITS (Selective Embedded Just-In-Time Specialization) [9], 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 [2] for a recently developed bootstrapping algorithm, we generate from native Python code a scalable, distributed 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 original 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. (add one sentence here on findings of experiments)

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 sets, test sets, and classifier. 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, speculation of classifier performance is a well understood problem [6]. Extensive investigation of bootstrapping and its variations [4, 8, 3, 12] has identified bootstrapping as a prime, ableit not perfect [7], 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)[10, 11], 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 [2]. This DSEL compiler makes use of Selective Embedded Just-In-Time Specialization (SEJITS) [9], an approach for converting DSELs in productivity level languages to high performance efficiency language (e.g. C++ or Cilk) code.

Here, we code 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– utilizing a bootstrapping 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 timing results for naive Python. Moreover, this methodology templates the mechanism through which further useful yet demanding, multimedia-analysis computations can be made attainable for non-performance programmers.

2. METHODOLOGY DESCRIPTION

The propoposed methodology enables multimedia domain experts to estimate multimedia classifier performance in Python, without worrying about making the code run efficiently. Multimedia domain experts need only adhere to the specified DSEL for BLB, for which they must define with a modest subset of Python the statistical function to be estimated on the data set, along with the reducer function (e.g. standard deviation) that measures the error on the estimate. More detailed information about the DSEL, or the subset of Python allowed can be found in [2], but as an overview, all basic flow control statements, lists, variable assignments, arithmetic operations, basic type conversions, and simple list and string operations (e.g. `len, range, split`) are available.

For this paper's experiments, we choose to have the statistical function compute the EER of a multimedia classifier on a set of feature vectors. The reducer function, which measures the error on the estimate, is the standard deviation. It would, however, be simple to modify the statistical function to estimate an alternate measure of classification performance, such as the misdetection rate at a certain false alarm rate. Similarly, the error estimate function could instead indicate uncertainty in terms of a confidence interval.

The already existing BLB DSEL compiler consumes the BLB Python application, and emits a scalable, distributed BLB application runnable on the Spark cluster computing system. Spark, similar to Map-Reduce, operates on clusters of commodity hardware, such as those purchasable with Amazon's EC2. Multimedia feature vectors, machine learning models, classification scores, or other input data is read from the Hadoop File System (HDFS) to support large input data sizes of up to hundreds of GBs.

(should talk about in here how we can also run pure python for debugging or if problem size isn't large enough to need to run in cluster ..

probably most bootstrapping applications where scores are inputted will be fine in just python, except for the absolutely most taxing, or where scores need to be computed in python, then quickly out of reach ... scalability)

3. DATA SET AND CLASSIFICATION SYSTEM DESCRIPTION

The data used in the following experiments is a subset of the NIST TRECVID MED 2012 corpus, which comprises 150,000 consumer-produced videos of around three minutes each. We select training sets of roughly 100 videos each for 20 different "events," or classes, that range from changing a car tire to making a sandwich. Classifying based on audio properties alone, we extract feature vectors with an ivector system (citation?), and build machine learning models with a binary support vector machine [1]. Our primary test set consists of 26,399 files, of which 24,920 files do not correspond to any event while the number of positives for each event varies between 15 and 234.

(will probably want to talk a little more about classification method/systems here)

4. EXPERIMENTS VARYING TRAINING SET SIZE

We vary the number of training positives for each event, decreasing by 10 each time from roughly 100 positives to 10 positives.

–Will soon show here graph of actual EER on training sets from each of the 10 models built for each of the 20 events

–Will soon show here graph of standard deviations of EER estimated on test set from training set of blb for each of the 20 events

–Will soon show here graph of actual EER on test set from each of the 10 models built for each of the 20 events

will locate point at which estimated standard deviation from BLB becomes so large, or performance becomes so poor, that cannot reduce training set further

this is minimal necessary training set size of consumer produced videos from this set for our classification performance requirements (to be defined).

data here can probably help to support: [5]

5. EXPERIMENTS VARYING TEST SET SIZE

We vary the number of test positives for each event in test set, arbitrarily halving the number of positives each time (number of positives range from 15 to 234). A high standard deviation from BLB on a test set indicates that the result on this test set may not be significant, and that it could sway largely on similar test sets. Contrarily, a low standard deviation from BLB indicates that the classification accuracy on similar test sets will likely be comparable, and even that the size of the test set may be reduced.

–Will soon show here graph of actual EER on each of the test sets (full, 1/2 positives, 1/4 positives, 1/8 positives) for each of the 20 events

–Will soon show here graph of standard deviations of EER estimated on each of the test sets for each of the 20 events

discussion of at what points test set is too big/small from results ..

6. COMPARISON TO NAIVE PYTHON

In this section we will briefly compare the performance of the generated code to that of naive Python, indicating why it makes sense to use this specializer by seeing just how much speedup can be gained

–table comparing times between distributed code and python for a few example runnings of blb

probably would be good to compare python times for when passing in scores, and a python time for a very small running where scores are calculated in estimator function

7. FUTURE WORK

explore other applications of scalable bootstrapping for multimedia analysis

8. CONCLUSION

Our proposed methodology makes accessible to multimedia classification domain experts coding in Python what was never accessible before: an efficient means to estimate multimedia classifier performance via bootstrapping. After specifying in Python the measure of the classifier performance, such as EER, or a misdetection rate for a given false alarm rate, the domain expert enjoys scalable performance without the stresses of implementing scalable performance. The ability to rapidly express efficient bootstrapping applications simplifies pinpointing ideal test and training set sizes with the goal of maximizing accuracy while minimizing comptuation time, as shown by our experiments. This methodology can be extended using the SEJITS approach to allow other taxing multimedia analysis computations to be approached from productivity level languages such as Python.

9. ACKNOWLEDGEMENTS

10. REFERENCES

[1] Multi-class support vector machine. `http://svmlight.joachims.org/svm_multiclass.html`.

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

[3] 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.

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

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

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

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

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