Sampling-based Prediction of Algorithm Runtime

Quan Sun
Department of Computer Science
University of Waikato
Private Bag 3105
Waikato, New Zealand
qs12@cs.waikato.ac.nz

ABSTRACT
During the last decade, with the growth of computing power, machine learning algorithms have been widely adopted and used in various fields of computer science and real world applications. Currently, users of machine learning algorithms do not usually receive feedback on when a given algorithm will be finished with building a model for a particular data set. In this paper, we first investigate how to use sampling-based techniques to predict the running time of a machine learning algorithm training on a particular data set. Secondly, we empirically evaluate a set of sampling-based running time prediction methods. Experimental results show that, with some care in the sampling stage, by applying designed transformations on the running time observations and then using regression techniques as the base estimation model, it is possible to obtain useful average-case running time predictions for a given machine learning algorithm building a model on a particular data set. In this study, WEKA [7] machine learning algorithms are used for all experiments.

Categories and Subject Descriptors
I.2.6 [**Artificial Intelligence**]: Learning – *machine learning, parameter learning.*

General Terms
Algorithms, Measurement, Performance, Experimentation.

Keywords
Machine learning, sampling-based, runtime prediction, linear regression methods.

1.INTRODUCTION
There are two kinds of approaches that can be used to estimate the running time of an algorithm. The first is to use knowledge about the underlying algorithm to do a theoretical performance analysis in the Random Access Machine model [2], and then use this information to estimate the running time. This approach works when the target algorithm is simple. For more complex algorithms, such as machine learning algorithms, this approach can be a very difficult task. Another kind of approach is called 'empirical algorithm analysis'. Empirical algorithm analysis employs sampling-based techniques to construct a function that is an approximation to the true running time function of a given algorithm. In this paper, we focus on the latter approach. In Section 2, background about algorithm analysis and basic ideas of empirical algorithm analysis are given. Previously, in the domain of empirical algorithm analysis, researchers were interested mainly on obtaining closed form expressions for algorithms complexity. McGeoch [4] present several sampling-based techniques for curve bounding. These techniques are discussed in Section 3. The brief implementation details of six running time estimators examined in this study are given in Section 4. Experimental results are given in Section 5.

2.BACKGROUND
In mathematics, computer science and related disciplines, an algorithm is a step-by-step procedure for calculating a given task with a finite amount of resources, such as time, storage and computer memory usage. A given task may be completed by using different algorithms with a different set of resource requirements. One goal of algorithm study is to find methods that can be used to precisely calculate – or at least approximate – how much of a particular resource is required for a given algorithm on a particular task. Such study is referred to as algorithm analysis. In this paper, methods that can predict the running time of machine learning algorithms are investigated, and evaluated by experiments. This task formally can be thought of as a method of describing limiting behavior. For instance, let f and g be two functions of a natural number n. If f and g are asymptotically equivalent as $n \to \infty$, then:

$$\lim_{(n \to \infty)} \frac{f(n)}{g(n)} = 1.$$

Assume $f(n)$ is the running time function of an algorithm, where n is the input size. The goal is to find a method that can automatically construct an estimation model (asymptotic function) $g(n)$ based on sampling-based techniques, and then use this model to predict the running time t, where $t = f(n)$. Although the methods discussed in this paper can be applied to algorithms in general, only the running time estimation performance of these methods on machine learning algorithms is investigated.

Empirical Algorithm Analysis
The asymptotic behavior of an algorithm can be analyzed based on experimental observations. In this approach, firstly, the running times of an algorithm for different sizes of input are observed. Secondly, a curve fitting or curve bounding rule is applied for formulating the asymptotic trend or behavior. The uses of running time observations and experimental results can be different from problem to problem. Having these experimental data, one can draw conclusions about the asymptotic behavior of an algorithm. However, no inference about asymptotic behavior is reliable, since one can observe only a finite number of input sizes. Another problem is due to the random noise caused by the complexity of the machine model and experimental environment.

3.EMPIRICAL ASYMPTOTIC ANALYSIS

In this section, we consider methods for empirical asymptotic analysis. It is assumed that the observed running time t of an algorithm can be expressed by a unknown function $t(n)$, where n denotes the input size. An experiment produces observations, which is a set of $\{n,t\}$ pairs. Having these observations, the goal of empirical asymptotic analysis is to find methods that can construct a function $g(n)$ that is an asymptotic approximation to $t(n)$. In this section, existing empirical asymptotic analysis approaches are discussed. The concepts and approaches described in this section form the theoretical foundation of the running time estimators implemented and evaluated in this study.

Numerical Approaches

Guess Ratio Test

In the guess ratio test, it is assumed that the main term of the underlying algorithm's running time function can be formulated by $g(n)=n^c, c>0$, where n is the input size. Let $t(n)$ denote the observed running time. In [4], the guess ratio $r(n)$ is defined as $\frac{t(n)}{g(n)}$. If the ratio grows as the input size increases, then $g(n)$ underestimates the running time; if the ratio converges to 0 as the input size increases, then the $g(n)$ is an overestimate. In the case that the ratio converges to some constant b greater than 0, then $g(n)$ is a good estimate for the growth rate of $t(n)$.

Guess Difference Test

The guess difference test [4] works similarly to the guess ratio test in the sense of iterating over guess functions. Rather than evaluating the guess ratio curves, the guess difference test evaluates the difference defined as $g(n)-t(n)$. The test begins with guessing a function having the form $g(n)=an^b$ where a and b are positive rationals. In theory, if the difference curve increases monotonically with n, then the guess function $g(n)$ is not $O(t(n))$; if the difference curve monotonically decreases in the range from n_1 to n_k, then monotonically increases after n_{k+1}, the guess difference test concludes that the guess function $g(n)$ has the "Down-Up" property. Then the test needs to search for other difference curves that have the "Down-Up" property by adjusting the coefficient a until a new "Down-Up" curve is found. When that happens, the guess function is assumed to overestimate the exponent b of $t(n)$. In this case, the guess difference test needs to try another exponent, namely b' where $0<b'<b$, and applies the same "Down-Up" curve searching procedure again.

Sampling-based Approaches

Simple Linear Regression

Simple linear regression is a method that studies the relation between a response variable y and a single explanatory variable x. It assumes that for each value of x, the observed values of the response variable y are normally distributed about a mean that depends on x. The statistical model for simple linear regression states that the observed response y_i when the explanatory variable takes the value x_i is $y_i=b_0+a_1x_i+e_i$, $y_i=b_0+a_1x_i$ is the mean response when $x=x_i$, and e_i are the deviations that are assumed to be normally distributed with mean 0 and standard deviation s; e_i is also referred to as the random error. This is used as a basis for the techniques that follow.

Power Test

As in the guess difference test, the power test method [3], [4] also assumes $t(n)$ can be formulated by $g(n)=an^b$ where a and b are positive rationals. To find the proper a and b, the power test applies a logarithmic transformation on each $\{n,t\}$ pair in the observations. Secondly, it examines the $\{n',t'\}$ pairs, where $n'=\log n$, $t'=\log t$, to see whether they can be fitted by a simple linear regression line.

Ladder Transformations

The power family of transformations $T(y)=y^k$ or $T(x)=x^k$ provides a set of transformations for "straightening" a single bend in the relationship between two variables, and is referred to as a family of "one-bend" transformations [4], [6]. These transformations can be used on either x or y. If the transformations are ordered according to the exponent k, a sequence of power transformations is given. In [5], this is called the ladder of transformations. For example:

$$k=-1,-\frac{1}{2},0,\frac{1}{2},1,2,$$

where the power transformation $k=0$ is to be interpreted as the logarithmic transformation. In applying the idea of ladder transformations to asymptotic analysis, the procedure is to try several transformations of n for the $\{n,t\}$ pairs in the observations, where n corresponds to the explanatory variable x, t corresponds to the response variable y in the simple linear regression model, and then choosing that transformation $T_i(n)$ which make the points most nearly collinear.

4.RUNNING TIME ESTIMATORS

Based on the theoretical foundation discussed in Section 3, six sampling-based running time estimators, namely A1, A2, A3, A4, A5 and A6, were implemented for this study. To simplify the description, we introduce some data abstractions. An *Observation* object is defined as a data structure consisting of an $\{n,T\}$ pair, where n is the input size, and $T\langle t_1,\ldots,t_m\rangle$, which is a vector of running times observed from m runs, for input size n. An *Observations* object is defined as a data structure that is a collection of *Observation* objects. Table 4.1 shows an example *Observations* object (without actual observed values). An *Estimator* takes an *Observations* object as input to build an estimation model that can predict the running time of a given input size. The 'training observations' corresponds to the *Observations* object that is used to build an estimation model. The 'testing observations' corresponds to the *Observations* object that is used to evaluate the prediction performance of a running time estimator.

Estimators

A1 – Power Rule with Simple Linear Regression

The A1 estimator (based on the power test) first applies a log-log transformation on each *Observation* in the *Observations* object. Secondly, A1 builds a simple linear regression model based on the transformed *Observations* object. To predict the running time t for a given input size n, A1 uses the simple linear regression model to predict response variable t' of explanatory variable n', where

$n' = \log(n)$. Finally, A1 gives the predicted running time as $t = \exp(t')$.

A2 – Box-Cox Method with Simple Linear Regression

The A2 estimator (based on the Box-Cox method) works by searching for the best simple linear regression model after applying different transformations on the explanatory variable *n* – the input size. The Box-Cox transformations are controlled by evaluating the exponent λ of n, which is defined as:

$$T(n) = n^{\lambda}, \quad n^{\lambda} = \begin{cases} \dfrac{n^{\lambda} - 1}{\lambda n^{\lambda - 1}} & \text{if } \lambda \neq 0 \\ \log(n) & \text{if } \lambda = 0 \end{cases}$$

In this work, A2 applies transformations on *n* from the range $\lambda = 0 \text{ to } 2.5$, where λ is incremented by 0.1 in each step. There are a total of 26 transformations. Thus, A2 builds 26 simple linear regression models corresponding to the 26 transformations. The 'best' simple linear regression model over these transformations is defined as the one that results in the lowest squared error. To predict the running time *t* for a given input size n, A2 uses the 'best' simple linear regression model to predict the response variable *t* based on the explanatory variable n', where $n' = T(n^{\lambda_{best}})$.

A3/A4 – Ladder Transformations with Simple Linear Regression

Estimator A3 applies the most commonly used power transformations for the input size *n.* These transformations are controlled by the exponent *k* of $g(n)$, which is defined as $T(n) = n^k$, where *k* is set to 0.5, 1 and 2. Like A2 (Box-Cox estimator), A3 uses the 'best' simple linear regression model over these transformations. Rather than using the most common ladder transformations, A4 uses a set of ladder transformations that are designed for predicting an algorithm's running time. Running time functions can be ordered by growth rate, such as $\log n, \log^2 n, \sqrt{n}, n, n\log n, n^{1.1}, n^{1.2}, n^{1.3}, \ldots, n^{1.9}, n^{2.0}, n^{2.1}, n^{2.2}, n^{2.3}, \ldots, n^{2.9}, n^{3.0}$.

A5/A6 – Advanced Ladder Transformations and Multiple Regression without/with Feature Selection

A5 uses the same set of ladder transformations as A4. The difference is that A5 uses multiple regression as the base estimation model, which assumes the response variable depends on not one but multiple explanatory variables. The only difference between A6 and A5 is that A6 uses feature selection before building the base multiple regression model.

5.EXPERIMENTAL RESULTS

This section shows the experimental results obtained using the six estimators predicting the running time of machine learning algorithms.

Experimental Environment

The hardware and system specifications of the computer used for running all the experiments are:

Hardware

Processor: 32-bit Intel Pentium 4 3.00GHz
Memory: 2GB

Software

Operating system: Ubuntu Linux 8.04 (Kernel Linux 2.6.24-19-generic)
WEKA version: 3.5.7

The average CPU usage of other system processes while running an experiment was less than 5%.

Variance Reduction

Random noise leads to variability between runs of an algorithm taking the same input instance and may impede the prediction performance of sampling-based estimators. When generating running time observations, rather than simply using the running time of a single run to represent the running time of an algorithm M training on a subset of data set *S*, the multiple runs approach is employed to get a stable running time representative *r* of *M* training on a subset *s*. There are many possible choices for *r*. In this study, the sample mean and the upper/lower limit of the estimated population mean with 95% confidence are employed. Not only the variation of running time contributes to the random noise, but also the method that is used to generate the subset of a data set *S*. In machine learning, for classification problems, a training set *S* has its own class distribution. When sampling a subset of size *k* from *S*, a simple method randomly selects *k* instances from *S*. If this method is employed, *s* does not necessarily follow the same class distribution as *S*. In this study, a more sophisticated method is used, which generates a subset *s* that has the same class distribution as the full training set *S*. This method is also known as *stratified sampling*.

Experimental Results

Due to the space limitations, we only list the main results of this study.

Table 1. The best running time sampling treatment for each estimator

Estimator	Sampling treatment for running time observations	Percentage of wins
A1	Upper limit of estimated population mean with 95% confidence	51%
A2	Upper limit of estimated population mean with 95% confidence	64%
A3	Upper limit of estimated population mean with 95% confidence	60%
A4	Lower limit of estimated population mean with 95% confidence	45%
A5	Lower limit of estimated population mean with 95% confidence	39%
A6	Upper limit of estimated population mean with 95% confidence	40%

Comparing Prediction Performance of a Single Estimator Using Different Sampling Treatments

Table 1 lists the best running time sampling treatment for each estimator. The result was obtained by running experiment over 23 WEKA machine learning algorithms in 3 different sampling treatments and 5 sampling setups for training observations.

Comparing Prediction Performance Between Estimators

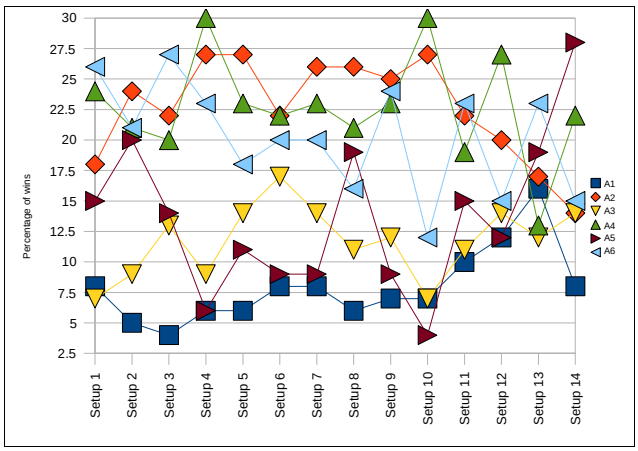

Figure 1. Prediction performance curves of the six estimators while the size of Observations increases

Figure 1 shows the experimental results of the prediction performance of the six estimators over 23 WEKA machine learning algorithms in 14 different sampling setups. Both A2 (based on Box-Cox transformations) and A4 (based on advanced ladder transformations) win for about 23% of the total 3703 tests. Estimator A6 wins about 22% of the 3703 tests. Note that S*etup 1* to S*etup 14* are ordered by the number of training *Observations.* It can be seen that when the size of the training *Observations* is small, for example, from S*etup 1* to S*etup 6*, estimators A2, A4 and A6 outperform others. However, when the size of the training *Observations* increases, the prediction performance of all six estimators gets closer. Overall, A2 and A4 outperform the other estimators.

6.CONCLUSIONS AND FUTURE WORK

Conclusions

Sampling-based running time prediction for machine learning algorithms, by its very nature, is a function approximation problem. From the theoretical perspective, mathematical analysis procedures, such as the asymptotic analysis approaches discussed in Section 3, form the fundamental ideas for sampling-based running time prediction methods. From the practical point of view, applying variance reduction techniques and statistical treatments to the running time observations is necessary. The results for the experiments presented in Section 5 show that estimators using the 95% confidence upper/lower limit of the estimated population mean as the training data perform better than simply using the sample mean. All the estimators implemented in this study use a search procedure to find appropriate transformations and then construct regression models based on transformed observations. The experimental results show that estimator A2 and estimator A4 outperform the other estimators, especially when the number of training observations is not large. The results also show that as the number of training observations increases, the prediction performance of all six estimators gets closer.

Future work

There are some questions left for future research, these include:

Random noise modeling

In this study, the random noise is modeled as the random error in linear regression models, which is assumed to be normally distributed. However, whether the random noise in empirical algorithm analysis follows a normal distribution is still an open question. Therefore, one direction for future research is to investigate whether random noise can be modeled more accurately.

Overfitting problem in multiple regression models

Estimator A5 and A6 use multiple regression models as the base estimation model. The model usually gives a polynomial function that can be used to predict a future response variable. However, the polynomial function may not be a monotonic function, thus it might give a negative response prediction. One avenue for future research is to apply multiple regression models that can return a monotonic polynomial function.

Sampling setup

For practical use, ideally a running time estimator should finish its prediction computation in a few seconds on a moderate computer like the one used for this study. In order to achieve this goal, the size of the sampling setup needs to be as small as possible. One direction for future research is to investigate how to compute an optimal sampling setup for a machine learning algorithm and its input instance.

Prediction performance evaluation

In this study, counting the number of wins based on the smallest absolute error is employed to evaluate estimators. One future research direction is to use more sophisticated evaluation methods to examine the prediction performance between estimators.

7.REFERENCES

[1] Box, G. P. and Cox, D. R. An analysis of transformations. Journal of the Royal Statistical Society (1964), 26, 211–246.

[2] Elgot, C. C. and Robinson, A. Random-access stored-program machines, an approach to programming languages. J. ACM (1964), 11, 365-399.

[3] Goodrich, M. T. and Tamassia, R. Algorithm design: Foundations, analysis, and Internet examples. Paris, John Wiley & Sons, Inc, 2002.

[4] McGeoch, C. Using finite experiments to study asymptotic performance. Experimental algorithmics (2002), 93-126.

[5] Mosteller, F., & Tukey, J. W. Data analysis and regression: A second course in statistics. Reading, Mass, Addison-Wesley, 1977.

[6] Tukey, J. W. Exploratory data analysis. Reading, Addison-Wesley, 1977.

[7] Witten, I. H. and Frank, E. Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, San Francisco, 2 edition, 2005.