Nonlinear Prediction in Credit Forecasting and Cloud Computing Deployment Optimization

by

Nicholas W.D. Jarrett

Department of Statistical Science
Duke University

Date: ______________________________

Approved:

____
Sayan Mukherjee, Supervisor

____
David Banks

____
Katherine Heller

____
Jun Yang

Dissertation submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in the Department of Statistical Science
in the Graduate School of Duke University
2015

Abstract

Nonlinear Prediction in Credit Forecasting and Cloud Computing Deployment Optimization

by

Nicholas W.D. Jarrett

Department of Statistical Science
Duke University

Date: ____________________________

Approved:

____

Sayan Mukherjee, Supervisor

____

David Banks

____

Katherine Heller

____

Jun Yang

An abstract of a dissertation submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy in the Department of Statistical Science
in the Graduate School of Duke University
2015

Abstract

This thesis presents data analysis and methodology for two prediction problems. The first problem is forecasting midlife credit ratings from personality information collected during early adulthood. The second problem is analysis of matrix multiplication in cloud computing.

The goal of the credit forecasting problem is to determine if there is a link between personality assessments of young adults with their propensity to develop credit in middle age. The data we use is from a long term longitudinal study of over 40 years. We do find an association between credit risk and personality in this cohort. Such a link has obvious implications for lenders but also can be used to improve social utility via more efficient resource allocation

We analyze matrix multiplication in the cloud and model I/O and local computation for individual tasks. We establish conditions for which the distribution of job completion times can be explicitly obtained. We further generalize these results to cases where analytic derivations are intractable. We develop models that emulate the multiplication procedure, allowing job times for different deployment parameter settings to be emulated after only witnessing a subset of tasks, or subsets of tasks for nearby deployment parameter settings. The modeling framework developed sheds new light on the problem of determining expected job completion time for sparse matrix multiplication.

To Pyotr Kropotkin.

Contents

List of Tables

List of Figures

List of Abbreviations and Symbols

Symbols

R^2	Pearson's product-moment correlation coefficient
$\mathbb{1}$	Indicator function
diag	An operator to construct a diagonal matrix from a vector, or block diagonal matrix from a collection of matrices
df	Degrees of freedom, as in the case of T distributions.

Abbreviations

FLOP	Floating Point Operation
JBLAS	Java Basic Linear Algebra Subprograms
MLGP	Multinomial Logit Gaussian Process
MPQ	Multidimensional Personality Questionnaire
RMM	Replication Based Matrix Multiplication
SVM	Support Vector Machine
VC	Vapnik Chervonenkis

Acknowledgements

Thank you.

1

Introduction

In this work we present data analysis and methodology for two prediction problems. The first problem is forecasting midlife credit ratings from personality information collected during early adulthood. The second problem is analysis of matrix multiplication in cloud computing.

1.1 Credit Forecasting

The goal of the credit forecasting problem is to determine if there is a link between personality assessments of young adults with their propensity to develop credit in middle age. The data we use is from a long term longitudinal study, following a birth cohort, of over 40 years. We do find an association between credit risk and personality in this cohort, and there are several features of the data which suggest that this link may generalize to individuals outside the study group.

Inclusion criteria into the study was primarily defined by being born in New Zealand's fourth largest city, Dunein, during the year enrollment was active. By virtue of this design decision, the study participants represent a truly diverse group spanning a wide spectrum of different walks of life. Furthermore, the Dunedin study has enjoyed an extremely high retention rate, which is important as individuals generally self-select their disappearance from prospective longitudinal studies, leading bias in subsequently discovered patterns and trends (Krueger et al., 2000). Lastly, personality is assessed using the Tellegen's Multidimensional Personality Questionnaire (MPQ). Personality traits defined based on the MPQ have been shown to be partially heritable and stable from early adulthood (Caspi and Silva, 1995; McGue et al., 1993; Tellegen et al., 1988).

All of these factors combine to lend plausibility to the ability to extrapolate the link between personality and credit development at midlife to individuals not in the test group. This has obvious implications for lenders but also can be used to improve social utility via more efficient resource allocation.

The link between personality and credit development is explored through the use of several learning algorithms including elastic net penalized multinomial logistic regression, naïve Bayes, support vector machines, and multinomial logistic Gaussian process regression.

Our approaches are influenced by the extant body of epidemiological personology research, and we incorporate into our algorithm design features which capitalize upon the structures captured by methods historically used in this area. For example, factor analysis is a very useful tool in this area as it can be used to validate relatively natural personological theories, by grouping questions into categories like, 'anger' or 'jealousy', and then grouping these latent dimensions into more general emotional states like 'negative emotionality.' The effect of correlated predictors are naturally drawn together by such an interpretation, and this behavior of factor analysis motivated our use of the elastic net penalty for multinomial logistic regression, since it also has the property that the regression coefficients for correlated predictors are shrunk towards each other.

1.2 Matrix Multiplication

Matrix multiplication is ubiquitous in statistical analysis, for which cloud resources are increasingly being used. Many cloud applications either directly involve multiplications of large, real matrices, or iteratively perform matrix multiplications, to approximate the effects of non-linear procedures while taking advantage of the gains in computation time from parallelization. This is especially the case for statistical analysis of massive datasets, as direct application of nonlinear algorithms may yield unsatisfactory completion times, or may simply have computational requirements (e.g. on working memory) which exceed what is available or what is practically obtainable.

We analyze matrix multiplication in the cloud and model I/O and local computation for individual tasks. This work is performed within the context of the Cumulon execution model, as it is more efficient through avoiding the unnecessary workloads traditionally incurred in MapReduce approaches relating to replication and shuffling procedures (Huang et al., 2013).

We develop models that emulate the multiplication procedure, allowing job times for different deployment parameter settings to be emulated after only witnessing a subset of tasks, or subsets of tasks for nearby deployment parameter settings. The models incorporate a great deal of expert knowledge, allowing for the construction of a parsimonious semi-parametric representation additively featuring minimally dependent submodels. This increases the efficiency at which patterns can be discovered, and allows for information sharing across deployment parameter settings.

Conditions are established under which the distribution of job completion times can be explicitly obtained, and under which we can reproduce expressions estimating job completion times from previous non-probabilistic approaches. These results are further generalized to situations where analytic derivations are intractable. These results are mainly targeted towards a single prohibitively large multiplication problem, however iterative procedures can be also be optimized stepwise given these results. Given the distribution of job completion time, it is possible to obtain expected job completion time, which can be optimized to identify optimal deployment parameter settings.

This work also represents advancement towards a solution to estimation of job completion time for sparse matrix multiplications. This problem is plagued by random variables with excessive skew and large variances, to such a degree that non-probabilistic methods of estimating job completion time may break down. The insights into the distributions of subcomponents of matrix multiplications developed here can be utilized to help parse down these uncertainties and obtain reliable estimates of job completion times.

2

Personality and Credit Forecasting

2.1 Introduction

In this work we employ various statistical and machine learning methods to assess the utility of using personality information to predict future credit score for individuals with limited credit history. Based on the behavior of these algorithms, we identify the nature of the functional relationships between personality and long term development of credit score, leading to implications on optimal resource allocation for credit lines, especially those extended to individuals with limited credit histories.

This research follows in the wake of a rapidly growing body of epidemiological personology research which attempts to pair developmental perspectives on individual differences and population based sampling frames to obtain insights about the possible social outcomes related to personality (Krueger et al., 2000). Through this lens, personality has been linked to a wide range of social outcomes encompassing mental disorders and health-risk behaviors, as well as other less directly health related social outcomes, but often in the context of specific theories (Krueger et al., 1998). In contrast to this, with the sophisticated statistical and machine learning algorithms which have been developed over the years, it is possible to employ non-parametric methods to investigate the utility in atheoretical approaches. To explain the context of epidemiological personology approach, it is relevant to independently consider epidemiology and personology.

Historically, epidemiological approaches are common in the study of infectious diseases, where a rigorous controlled experiment may not be feasible, yet through studying the patterns of outbreaks, it is possible to identify risk factors ultimately leading to improved quality of life for society. A famous example illustrating the merits of epidemiological research is given by the management of the cholera outbreak in London in the 1854. By analyzing the pattern of the outbreak, and collecting additional information in the form of testimonials from those involved, it was deduced that the evidence pointed to the Broad Street water pump as the origin of the outbreak. Statistical analysis enabled connections to be drawn from water quality, including sewage contaminants, and the spatial incidence of cholera cases, and even a simple dot map illustrated that most cases occurred in close spatial proximity to the problematic water pump. By questioning the families at outlying locations relative to the region of highest concentration, it was revealed that they often had connections to the Broad Street pump (Newsom, 2006).

Upon inactivation of the pump, the rate of new infections rapidly dropped, though the exact magnitude of the effect is unknown since the disease was already in decline by the time the policy change was made. However, it is still remarkable to note that this was done without de facto accepting the germ theory of disease, which was not widely believed at the time. It was later discovered that the well corresponding to this pump was situated very near a cesspit which began to leak into the water well at some point. This example illustrates how statistics can be utilized to explore temporal development of a response, and associate it with covariates to indicate further investigation for causality without specifying a rigorous model for how those covariates may cause the response to be developed over time.

Regarding the definition of personology, Henry Murray stated, “The branch of psychology which principally concerns itself with the study of human lives and the factors that influence their course, which investigates the individual differences and types of personality, may be termed ‘personology’ instead of ‘the psychology of personality,’ a clumsy and tautological expression,” (Murray, 1938). In this regard, we build upon the existing body of research from personology which informs the prediction algorithms, through supplying covariate information, to explore the patterns between differences in the trajectory of an individuals life course based on a snap-shot of their personality, specifically in terms of credit development.

Epidemiological personology combines these two ideas to generally explain the style in which analyses are performed in this area since hypotheses are often approached as they would be in epidemiology with large, hopefully population representative samples, and where the subsequent analyses of these data involve personological approaches, wherein personality information is utilized to understand the response which is under consideration for the target collection of hypotheses under scrutiny in the study. In this regard, it is possible imagine a spatial landscape of personalities which serves as the analog to the traditionally seen geographical distribution of outbreaks common to historical epidemiological works. Further, while epidemiological research typically identifies likely culprits which may impact the development of medical maladies, and subsequently recommends more optimal resource allocation in the search for solutions to end outbreaks, in the case of credit forecasting, greater understanding of the associations between personality and credit development, may lead to more optimal extension of credit lines to those with limited credit histories, by identifying regions with higher, or lower relative risk. Alternatively, the predictions could be used to serve as an early warning to an individual about the potential long term consequences of poor credit management, increasing their well-being as well as social utility on the whole.

2.2 Data Exploration, Illuminating the Path to Credit Forecasting

The data used comes from the Dunedin study, which is prospective and longitudinal, following a complete cohort of children born during a one year period beginning on April 1$^{\text{st}}$, 1972. The Dunedin study is aptly named, as the cohort originates from Dunedin, New Zealand, and is the country's fourth largest city. The study began with about 1000 participants 52% of which were male, with 91% of eligible births over the year. Participants are periodically reassessed, though the frequency is not completely regular. However, each assessment is performed within 60 days of the participants birthday, so to a granularity of plus or minus a couple months, data represent differences at identical biological ages. The prospective nature of the Dunedin study guards against the potential bias in estimating the responses to a personality battery many months or years after the fact. Over time, people's perspectives on past events change, and it is in general, difficult to adjust for this effect. Another important attribute of the Dunedin study is that despite the dispersion of study participants across New Zealand, and across the globe, the Dunedin study has maintained a very high retention rate which is helpful for the generalizability of the patterns discovered therein (Krueger et al., 2000).

Personality information is extracted using the Multidimensional Personality Questionnaire (MPQ) originating from Tellegen's work. Factor analysis is a popular technique in this area of research, and traditionally the question items are grouped into factors, and then 'super-factors' corresponding to hierarchies of personality traits, with the 'Big Three' or 'Big Five' representing the highest level of organization. These high level dimensions represent ideas such as, 'Negative Emotionality' or 'Neuroticism', and are considered to be comprised of more specific 'negative' dimensions such as 'guilt', 'anger', et cetera. These questions have been calibrated for use in nonclinical populations, and perhaps even more interesting is that studies have shown that personality traits ascertained via use of the MPQ are partially heritable, and stable from adolescence into adulthood. The stability of personality traits based on the MPQ is important to our work since credit scores were first obtained at age 38, and we are primarily interested in the assessment of credit forecasts based on personality information expressed before individuals have had significant time to establish their credit. Since it is known that there exist personality differences between the genders, the MPQ items are augmented with gender to form the set of covariates entertained as being potentially relevant to credit development.

The goal of this research is to demonstrate a link between personality and credit development, with more specific focus on the event that an individual develops a credit score which may not be sufficient for approval, by asserting that even over a long, twenty year horizon, it is possible to extract meaningful predictions from personality information.

To investigate this plausibility of such a link, we obtain and interpret the output of several statistical and machine learning methods, incorporating into our analyses structures which resemble those capitalized upon by factor analysis algorithms which have been shown to reveal interesting links between personality and the aforementioned categories of social outcomes from section 2.1, as well as borrowing general ideas from epidemiology. Two examples of ways by which we can achieve these effects are through combining information across correlated predictors via penalized regression likelihoods, and by estimating the conditional distributions of covariates given a particular response level.

As part of the initial exploratory data analysis, into the correlations and spatial patterns between variables, pairwise hexagonally binned scatter plots were constructed to begin identifying any obvious structures. For example, the pairwise hexagonally binned scatters between VEDA credit score at age 38, and the achievement, aggression, and alienation personality factors at age 18 is given in figure (2.1) below.

There are many interesting structures visible in the pairwise associations, especially in the low to medium credit ranges, however there is a tendency that every level of a dependent variable is expressed for high credit scores. It may be possible that the the dependent variables observed for high credit may assume their levels with some higher dimensional structure. This could potentially be captured by including interactions or more generally sharing information spatially across the range of personalities, however further analysis showed that the high credit observations don't occur together with a strong spatial pattern.

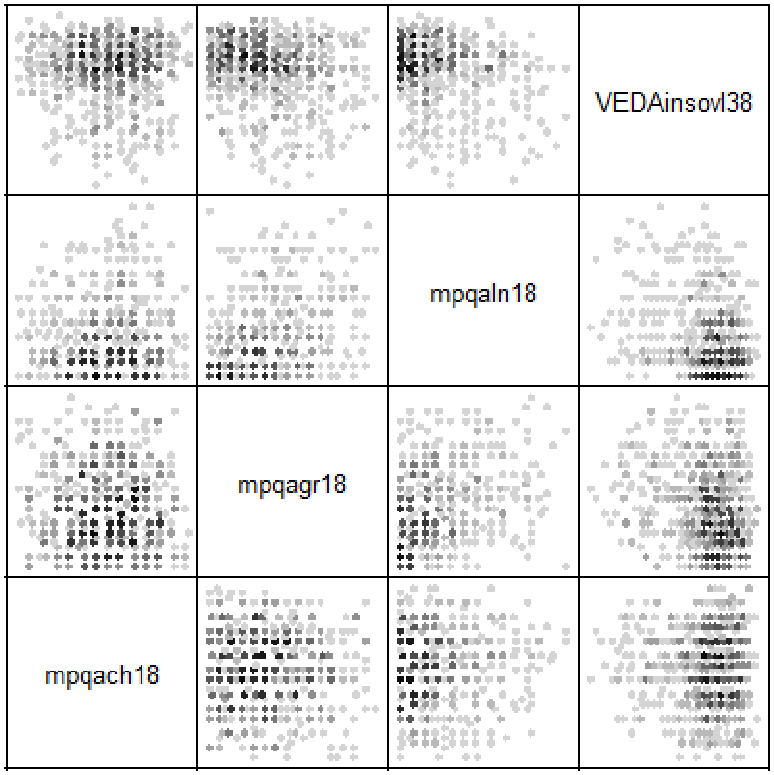


Figure 2.1: Hexagonally binned scatter plot of simulated response against simulated predictor variables, where the data simulation algorithm is based on on first order approximations to the pairwise joint distributions.

2.2.1 Partitioning Credit Scores

It is not clear that there is enough information with personality alone to distinguish between fine-scale changes in credit over this twenty year time horizon, and a prediction algorithm's performance could potentially suffer by down-weighting spatial information according to these differences. Regression on the raw credit scores, often leads to poor, potentially negative, predictive R^2 on a hold-out test set, even when accounting for possible predictor irrelevance to the response. This is partially due to the aforementioned high credit observations available at nearly every level of each dependent variable, which can lead to biased coefficients in many models. Furthermore, credit decisions are often subject to threshold cutoffs which can determine whether or not a lendee is approved, or rejected for a credit application. Given this, we parametrically define thresholds to partition credit into three groups allowing for a range of possible accept/reject cutoffs which we label as $\{0, 1, 2\}$, for low, medium, and high credit. This work proceeds forward with this categorical prediction problem, with our primary focus on being able to discriminate $\{0\}$ labels, from the labels $\{1, 2\}$ for each partition.

2.3 Methods

The fundamental procedures common to all methods are defined in greater detail below. In specific, all methods share the partition on raw credit scores, outer cross-validation subset patterns, and covariates entertained as being (potentially) relevant to predicting future credit development. Given these common procedures, the following subsections detail the elastic net penalized multinomial logistic regression, naïve Bayes, support vector machines, and multinomial logistic Gaussian process regression models' calibration and subsequent prediction procedures. Predictions are compared against the cross-validation test sets and assessed primarily via receiver operating characteristics (ROC); subsequent implications of algorithm performance on the nature of plausible functional relationships which may link personality with credit development will be discussed in each subsection.

Let Y denote the vector of observed credit scores. In accordance with section 2.2.1, construct a parametrically defined partition to establish categorical membership. This is achieved through the equation

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = t \begin{bmatrix} 820 \\ 850 \end{bmatrix} + (1-t) \begin{bmatrix} 420 \\ 750 \end{bmatrix} \tag{2.1}$$

Where τ_1 and τ_2 respectively denote the thresholds from low to medium, and medium to high credit.

This choice of partitions roughly covers the range of the data, without having excessive class imbalance, and avoiding the situation where any specific class label is a rare event, allowing us to assess predictive performance, as the fraction of responses labeled as 'low credit' in each case varies from nearly 0 to nearly 1. Then define categorical indicators of credit score, $Z \in \{0, 1, 2\}$, as

$$Z_i = \mathbb{1}(Y_i \leqslant \tau_1) + \mathbb{1}(Y_i \leqslant \tau_2) \tag{2.2}$$

Models were fit for $t \in \{0, 1/8, 2/8, \ldots, 1\}$ to illustrate the effect of changing the threshold on predictive performance.

The design matrix for all models consists of the set of all MPQ items, and gender. In all cases, 8-fold cross validation was used to assess predictive performance, with the cross validation partition randomly resampled 50 times to explore the effect of fixing the partition. This is intended to show that the results are insensitive to the randomness induced from specifying the cross-validation subsets.

Naturally, some models contain parameters which are not completely specified, for example kernel parameters in spatial methods. In these cases, such parameters were chosen by further running an inner loop of cross-validation within the training set of the outer cross-validation procedure.

2.3.1 Logistic Regression

For three class multinomial logistic regression, it is common to specify one of the categorical response's levels as a reference, often called a pivot, and fitting two logits to determine if the label should be changed from the reference value. Alternatively, each class can be given it's own logit function, but the resulting unconstrained parametrization is not identifiable.

The un-regularized likelihood is given by

$$\mathsf{P}(Z_i = z_i \mid x_i) = \frac{e^{\beta_{0z_i} + \sum_{j=1}^{p} x_{i,j} \beta j z_i}}{\sum_{k=0}^{2} e^{\beta_{0z_k} + \sum_{j=1}^{p} x_{i,j} \beta j z_k}} \tag{2.3}$$

It is well known that inclusion of predictors which aren't truly related to the response function can negatively impact predictive performance on a hold out test set. Many approaches to the variable selection problem suppose that predictors are either entirely in, or entirely out of the model. For example, consider the Bayesian spike-and-slab, or Frequentist forward/backward selection procedures (Ghosh and Clyde, 2011). It is also possible to obtain a sparse solution by adding a regularization penalty to the regression likelihood, as is the case in the popular Lasso algorithm, which adds an ℓ_1 penalty to the regression likelihood (Tibshirani, 1996).

For our logistic regression, we added an elastic-net penalty (Friedman et al., 2010) to the data log-likelihood, which interpolates between the Lasso algorithm's ℓ_1 penalty, which results in a sparse solution with many regression coefficients estimated to be exactly zero, and ridge regression's ℓ_2 penalty, which shrinks the regression coefficients towards zero, but maintains a dense solution. In addition to shrinking coefficients towards zero slightly, ridge regression also brings the coefficients of correlated predictors closer to each other, allowing the model to borrow strength from multiple correlated predictors. This behavior induced by the elastic net penalty for the estimated coefficients of correlated covariates resonates particularly well with the factor analysis roots of this research area, as factors are nearly interpretable as latent linear combinations of the observed covariates which attempt explain the observed covariance structure, and thus are intrinsically linked to correlation, where highly correlated predictors are grouped into the same 'factor', and possibly subsequent 'super-factors'.

As stated, the unconstrained likelihood is not identifiable in equation (2.3), however lack of identifiability in the likelihood simply means that the same data likelihood can be achieved by multiple parameter settings, which isn't a direct cause for concern as our primary goal is prediction, however there is only one combination of parameter values will both optimize the likelihood component while minimizing the elastic net penalty.

The regularized log-likelihood is given by

$$\tilde{\ell}(\beta) = \frac{1}{N}\sum_{i=1}^{N}\log(\mathsf{P}(Z_i = z_i \mid x_i)) - \lambda\sum_{k=0}^{2}\left[(1-\alpha)/2\left\|\beta_k\right\|_{\ell_2}^2 + \alpha\left\|\beta_k\right\|_{\ell_1}\right] \tag{2.4}$$

So that at $\alpha = 0$, the elastic net solution resembles ridge regression, with many non-zero coefficients and coefficients of correlated predictors shrunk towards each other, and at $\alpha = 1$ it resembles the Lasso, with the vast majority of coefficients set to exactly zero. As λ increases, more shrinkage and selection is induced in the solution set. Empirically, optimal α tend to be closer to Lasso than ridge regression, suggesting that while it is important to borrow strength from some correlated predictors, there may be others which should be excluded to optimize predictive accuracy.

Estimating Model Parameters

The coefficients are estimated via coordinate descent algorithm which cycles between the parameter updates for each of the 3 levels of the response variable, at each iteration optimizing the next update given the current set of parameters, which essentially determine a running residual for the optimization. More specifically, each set of logit parameters are updated given the rest using a second order Taylor approximation to the penalized likelihood, resulting in a conditional Newton-Raphson update step,

$$\tilde{\ell}(\tilde{\beta} \mid \tilde{\ell}(\beta)) \approx \tilde{\ell}(\beta) + (\tilde{\beta} - \beta)^T \dot{\tilde{\ell}}(\beta) + (\tilde{\beta} - \beta)^T \ddot{\tilde{\ell}}(\beta)(\tilde{\beta} - \beta)/2 \tag{2.5}$$

Since the log-likelihood of the new proposed logit, $\tilde{\beta}$, is quadratic, the standard normal equations can be used to obtain the expression for its conditional update

$$\tilde{\beta}_j = \beta_j - \dot{\tilde{\ell}}(\beta)|_j \ddot{\tilde{\ell}}(\beta)|_j^{-1} \tag{2.6}$$

where $\tilde{\beta}$ differs from β for the logit function indexed by response level j, and the derivatives, $\dot{\tilde{\ell}}$ and $\ddot{\tilde{\ell}}$, are restricted to the component contribution of the logit function indexed by response level j as well. This simplification is possible due to the abundance of zero differences in the Taylor expansion which eliminate any derivative terms including contributions from the fixed logit functions in the conditional update. Let $z_{i,j} = 1$ when $z_i = j$, then

$$\dot{\tilde{\ell}}(\beta)|_j = \frac{1}{N} \sum_{i=1}^{N} z_{i,j} x_i^T - x_i^T \frac{\exp\{x_i \beta_j\}}{\sum_{k=0}^{2} \exp\{x_i \beta_k\}} - \lambda \left[(1 - \alpha) I \beta_j + \alpha I \operatorname{sgn}(\beta_j)\right] \tag{2.7}$$

$$\dot{\tilde{\ell}}(\beta)|_j = \frac{1}{N} \sum_{i=1}^{N} x_i^T \exp\{x_i \beta_j\} x_i^T \frac{-1 + \sum_{k=0}^{2} \exp\{x_i \beta_k\}}{\sum_{k=0}^{2} \exp\{x_i \beta_k\}} - \lambda(1 - \alpha) I \tag{2.8}$$

This process is iterated until the conditional update has converged for all logit parameters, and predictors are added into the model in subsequent iterations as λ

decreases whenever their weighted inner product with the current working residual meets or exceeds $\lambda\alpha$, where the weights are given by $w_{i,j} = \mathsf{P}(Z_i = j \mid x_i, \beta)(1 - \mathsf{P}(Z_i = j \mid x_i, \beta))$ and the working response is defined as $x_i\beta_j + (z_{i,j} - \mathsf{P}(Z_i = j \mid x_i, \beta))/w_{i,j}$. This completes the procedure for obtaining the regularization path.

Generating Predictions

Even given the elastic net penalty, we still believed that the model might over fit on the training data. Therefore, instead of picking the point on the regularization path corresponding to coefficients which minimized the training error outright, we used coefficients corresponding to the largest value of λ such that we remained within one standard error of optimal training set accuracy. Given this set of parameters, predictions are assigned according to index of the maximum element of the sum in equation (2.3), that is, we predict the categorical label according to which ever is the most likely conditional response given the parameter estimates and design points for the held-out response under consideration.

2.3.2 Naïve Bayes

Naïve Bayes is a generative classification algorithm which attempts to model the joint distribution of credit categories and design information $\mathsf{P}(Z, X)$. Given this joint density, Bayes rule is then used to obtain $\mathsf{P}(Z \mid X)$, and predictions can be generated by the rule $\max_z \mathsf{P}(Z = z \mid X)$. Modeling the joint density is attractive from a practical perspective as there may be several personality profiles, which lead to the development of different levels of credit over time. The discriminative analog of naïve Bayes is logistic regression, which attempts to model $\mathsf{P}(Z \mid X)$ directly, rather than obtaining it through using Bayes rule on the joint density. For our implementation, $\mathsf{P}(X \mid Z = z)$ is assumed to have elliptical contours, and the predictors are a priori independent, which simplifies computation.

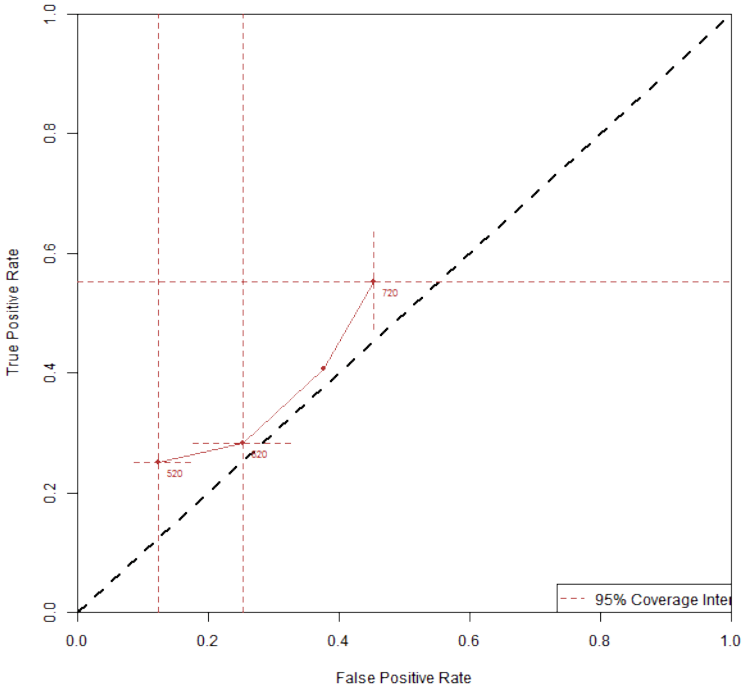


FIGURE 2.2: Threshold averaged ROC curve for the elastic net multinomial logistic regression algorithm. Even accounting for potential predictor irrelevance, and borrowing strength between correlated predictors, there is a large amount of variance in the true positive and false positive rates of the logistic regression classifier, with the algorithm potentially either completely capturing or completely missing predictions for the lower probability group at a given threshold. This distribution is skewed in the corresponding favorable direction, leading to the observed average performances. The threshold averaged ROC curve is only shown for thresholds in the range of partition elements for which predictions are not entirely dominated by the majority group. These behaviors could be partially caused by misspecification in the latent regression function, which we will explore using multinomial Gaussian logit processes, as well as negative correlation for cross-validation predictions in regression algorithms.

The asymptotic error of an unregularized logistic regression classifier, is smaller than that of its naïve Bayes counterpart, however naïve Bayes may converge more quickly to it's higher asymptotic error. It is possible to mathematically view the regularization term added to the logistic regression as directly modifying the likelihood instead of as an optimization aid. In this case, the convergence and error rate heuristics for model comparison do not strictly apply (Jordan, 2002). Given the challenging nature of the credit forecasting problem, the possibly accelerated convergence rate may be helpful in obtaining higher quality predictions.

The probability that a test point, z^* is assigned to class j given x^* is given by

$$\mathsf{P}(z^* = j \mid x^*) = \frac{\mathsf{N}(X^* \mid \mu^{(j)}, \Sigma^{(j)}) \sum \mathbb{1}(z_i = j)/N}{\sum_{k=0}^{2} \mathsf{N}(X^* \mid \mu^{(k)}, \Sigma^{(k)}) \sum \mathbb{1}(z_i = k)/N} \tag{2.9}$$

where $\mu^{(j)}$ is the vector of predictor column means and $\Sigma^{(j)}$ is a diagonal matrix of column variances restricted to design points with response level j.

With enough training points, the estimated distribution for an irrelevant predictor will be identical at each response level, which when combined with the independence assumption, causes its contribution to cancel out of the numerator and denominator of equation (2.9), leading to a kind of asymptotic variable selection based on dissimilarity of the predictor distribution across the response levels. For any finite amount of training data, the inclusion of irrelevant predictors will on average negatively impact the quality of predictions, however, without incorporating additional information beyond what is strictly contained within these data, it is unclear that predictor relevance can be established with enough certainty to outweigh the additional complexity incurred in model search. This is further supported by examining Markov chain trajectories based on spike-and-slab variant algorithms.

2.3.3 Support Vector Machines

Instead of presuming that predictors have the same functional effect everywhere, kernel methods can be used to more generally explore the impact of spatial patterns in the space of personalities with credit level. Kernel methods generally achieve this by associating the observed points in space from the training set with their responses, and assuming smoothness in the unknown response function, or unknown latent response functions, so that nearby points in the test set are labeled similarly

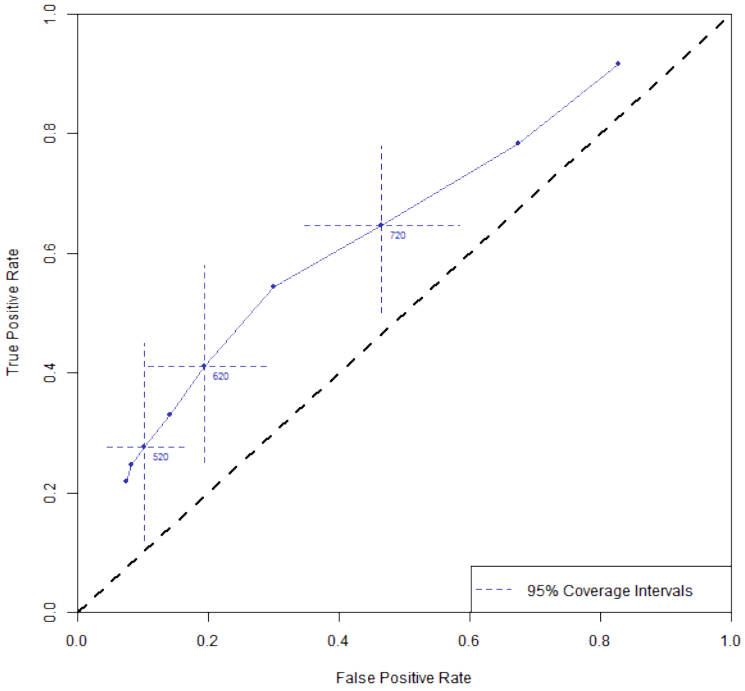


FIGURE 2.3: Threshold averaged ROC curve for the naïve Bayes algorithm. The naïve Bayes classifier tends to outperform most other classifiers at both very high, and very low thresholds. Both of these cases naturally feature less examples of either extreme group, potentially highlighting the superior rate of convergence to it's asymptotic error. Furthermore, it has significantly tighter coverage intervals than logistic regression, which are entirely contained above the line $y = x$, which represents the accuracy of random guessing in ROC space.

to their nearby training counterparts, where the kernel function determines how 'nearby' two points are to each other.

Support Vector Machines (SVMs) are non-probabilistic classification algorithms which assign responses to spatial regions in high dimensional space according to ordered relations based on maximum margin separating hyperplanes (Vapnik, 2000). The mapping to the high dimensional space is achieved by an a priori fixed kernel function, and the maximal margin requirement is to ensure that the resulting classifier has optimal generalization error based on VC theory. As such, it is important to explore different functional choices for the kernel, as well as to tune the parameters of any chosen kernel family to the prediction problem at hand. Soft-margin SVM further allows a trainable cost parameter, C, which specifies a trade-off between a hyperplane which correctly classifies all of the training data points, and one which has much greater margin between groups, but allows some misclassification with penalty proportional to the distance from the separating hyper-plane with coefficient C. The corresponding optimization problem to identify this hyperplane is given by

$$\min_{w \in \mathbf{R}^d} ||w||^2 + C \sum_{i=1}^{N} \max(0, 1 - \tilde{z}_i f(x_i)) \tag{2.10}$$

Where the $\tilde{z}_i$ terms appearing in this equation have been remapped to the corresponding one against the rest classification problem. The penalty term on the right hand side scales misclassifcations by rate C by the hinge loss. The hinge loss which appears here is inherently problematic to probabilistic approaches, and there does not exist a Gaussian process model with a likelihood which renders the two algorithms equivalent, as there exist likelihoods to bridge the gap between various other models. That being said, the results can be quite close, while theoretically distinct.

We explored the use of polynomial, radial, and sigmoid kernels. The functional form of each kernel and misclassification cost parameter were estimated as the parameter setting which optimized predictiver accuracy over an internal loop of cross validation. This solution was approached via the use of an adaptive grid to accelerate discovery of the optimal settings featuring successive refinements to regions corresponding to high predictive accuracy on the internal cross validation sets. The functional form of these kernels are given by

$$(\gamma x_1 \cdot x_2 + c_0)^d \tag{2.11}$$

$$\exp(-\gamma |x_1 \cdot x_2|^2) \tag{2.12}$$

$$\tanh(\gamma x_1 \cdot x_2 + c_0) \tag{2.13}$$

Polynomial kernels often resulted in a larger number of support vectors and poorer predictive performance on test sets, after performing model tuning over the parameters γ, c_0, d using an internal loop of cross-validation. Radial and sigmoid kernels performed the best with sigmoid kernels slightly outperforming radial kernels for some cross-validation splits, and for the highest credit score partitions.

It is well known that the sigmoid kernel may not be positive semidefinite depending on its parameter settings, however in practice γ was estimated to be small but positive, while c_0 was estimated to be small and occassionally positive (Lin and Lin, 2003). In this case, for negative c_0, the sigmoid kernel is conditionally positive semidefinite, and when c_0 is positive but small, it can be viewed as being 'close' to being positive semidefinite.

The sigmoid kernel with $\gamma > 0, c_0 < 0, |c_0| < \epsilon$ case has been stated to be similar to the radial kernel, which naturally confirms their similar performances observed in this case (Lin and Lin, 2003). Though there is no Gaussian process which is 'equivalent' to an SVM, the fact that the kernel is never estimated to be very far from positive definite implies that it is plausible to entertain a stationary Gaussian process based approach as being potentially useful for prediction.

2.3.4 Multinomial Logit Gaussian Process Regression

These SVM implementations do not directly incorporate the idea that the high credit responses are farther from a low credit response than a medium credit response. One way to adapt this concept regarding the ordinal nature of the credit partitions into

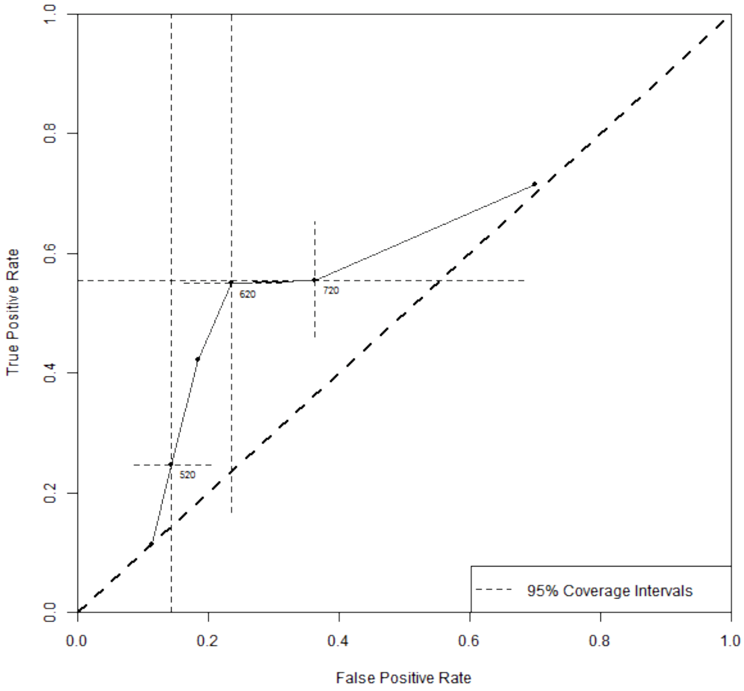


FIGURE 2.4: Threshold averaged ROC curve for SVM with Radial kernel. Behavior is overall similar to that of the sigmoid kernel, with the exception of a more quickly decaying tail as the partition divisions are significantly raised above their midpoint.

an SVM is through the requirement that the hyperplanes constructed in the high dimensional space must be parallel while simultaneously enforcing that each region must be labeled in order, so that points farther apart in the high dimensional feature space are predicted to belong to categories which are likewise farther apart. However the data has finite range, and so there is a wide variety of hyperplanes which separate the set of high dimensional feature points into three sets, where the outer two are separated by at least some positive distance of the ‘middle’ set everywhere.

In light of this, and in combination with the realization that trained kernels for SVM are never very far from being conditionally positive definite, we incorporate the ordinal information into non-parametric estimation of the response categories through the use of latent unobserved Gaussian process which inform logit functions for each response category. Even in the case where the latent Gaussian process distributed functions are a priori independent, the posterior distribution should discover some amount of the inter-function dependence, should it exist. It has therefore been claimed that the a priori independent function implementation is sufficient for applications for which prediction is the primary goal.

For each datum element, i, let $z_i \in \{0, 1, 2\}$ represent the indicator of membership to the corresponding credit partition element according to equation (2.1), and let f_i be a vector of possibly intercorrelated spatially varying functions over the domain of covariates. The likelihood is specified through

$$\mathsf{P}(z_i = j \mid f_i) = \frac{\exp e_j^T f_i}{\sum_{k=0}^{2} \exp e_k^T f_i} \tag{2.14}$$

$$\langle e_j^T f(\cdot), e_k^T f(\cdot) \rangle = K_c(j,k) K_x(\cdot,\cdot) \tag{2.15}$$

Wherein K_c encodes correlation structure between functional responses representing different outcomes, and K_x encodes correlation structure between spatial covariates, and e_j, e_k denote the corresponding elements of the canonical basis of a Euclidean space with the usual ordering. While this formulation is useful as it allows a significant amount of flexibility in the latent functions upon through which logit functions are informed to ultimately determine the categorical membership of the i^{th} datum, it leads to an exact analysis which is intractable.

Variational Parameter Estimation

More specifically, the exact posterior distributions for the multinomial logit Gaussian process as defined above are intractable. To proceed forward, we employ the variational multivariate normal approximation, $q \sim \mathsf{N}(m, V)$, to the posterior distribution developed by Chai (2012). Let Y denote the $3N$ dimensional column vector which is formed by stacking $y_i = 0 + e_{z_i+1}$, where for example $e_1 = [1, 0, 0]^T$. Let

$q_i \sim \mathsf{N}(m_i, V_i)$ denote the variational approximation for the i^{th} datum marginalizing over all other training points.

Optimization of the variational posterior involves iterating between conditional updates for each of b, S, V, where the variational optimization parameters $b = [b_1, \ldots, b_N]^T$ and $S = \text{diag}[S_1, \ldots, S_N]$ are updated using the variational lower bound to the data log likelihood, $h(y \mid q, b, S) = \sum_{i=1}^{N} h_i(y_i \mid q_i, b_i, S_i) \leqslant \ell(Y \mid q)$ where $h_i(y_i \mid q_i, b_i, S_i) \leqslant \ell(y_i \mid q_i)$. The parameter m is jointly updated with b using the self-consistent equation $m = K(Y - b)$, where $K = K_x \otimes K_c$ with K_x being the correlation matrix between the training points based on the chosen spatial kernel, K_c being the inter-function correlation matrix, and $\otimes$ representing the Kronecker product. The parameter V is optimized with respect to another variational lower bound to the likelihood, $\log(Z_h)$, which encompasses the datum specific lower bounds $h_i(y_i \mid q_i, b_i, S_i)$. The fixed point updates for each of these parameters may fail to improve their respective variational lower bounds, so a combination of Newton-Raphson, false position, and binary search are used to guarantee that each update increases the variational lower bound. The details of the learning procedure are given more thoroughly in the following subsections.

Updating b and S Let

$$h_i(y_i \mid q_i, b_i, S_i) = \frac{3}{2} + \frac{1}{2}\log|S_i V_i| - \frac{1}{2}\operatorname{tr}(S_i V_i) - m_i^T y_i - \log\sum_{k=0}^{2} g^k(q_i, b_i, S_i) \tag{2.16}$$

where

$$g^k(q_i, b_i, S_i) = \exp\left[m_i^T e_k + \frac{1}{2}(b - e_k)^T S_i^{-1}(b - e_k)\right] \tag{2.17}$$

and V_i is the i^{th} 3×3 diagonal block of V. Note that there are no cross-terms, so the element-wise gradient of h with respect to the vector b is given by stacking the gradient of h with respect to the i^{th} subvector of b, b_i.Then equating the gradient of h with respect to b_i with zero yields a fixed point update of

$$b_i^* = \bar{g}_i \tag{2.18}$$

where

$$\bar{g}_i = \frac{1}{\sum_{k=0}^{2} g^k(q_i, b_i, S_i)} [g^0(q_i, b_i, S_i), g^1(q_i, b_i, S_i), g^2(q_i, b_i, S_i)]^T \tag{2.19}$$

When this update fails, Newton-Raphson is employed to attempt to find the optimal value of $b*$ to optimize h, or equivalently h_i. The Newton-Raphson update is given by

$$b_i^* = b_i - \eta[I + (\mathrm{diag}(\bar{g}_i) - \bar{g}\bar{g}^T)S_i]^{-1}(b_i - \bar{g}_i) \tag{2.20}$$

Further, when the Newton-Raphson update fails to improve the variational lower bound, false position is used to find an optimal adjustment to the step-size, η, of the Newton-Raphson update in regions of high curvature to guarantee improvement in the variational lower bound.

As with the update for b, when updating S, we need only individually find the optimal S_i, as the cross-terms in the scalar-by-matrix derivative cancel out. Equating the gradient of h with respect to S_i with zero yields the fixed point update

$$S_i^* = S_i^{\text{fx}} = L_i^{-1^T} P_i \tilde{\Lambda}_i P_i^T L_i^{-1} \tag{2.21}$$

where L_i is the lower Cholesky factor of V_i, $P_i \Lambda_i P_i^T$ is the matrix of eigenvectors corresponding to $L_i^T A_i L_i$, and $\tilde{\Lambda}_i = (\Lambda_i + I/4)^{1/2} + I/2$. Since S_i should remain positive semidefinite after the update, instead of falling back on Newton-Raphson, when the fixed point update fails to improve the variational lower bound, false position (and grid search in regions of high curvature) is used to find an optimal interpolation between the original value of S_i and it's fixed point update, S_i^{fx}, directly. In these cases $S_i^* = S_i + \eta(S_i^{\text{fx}} - S_i)$ remains positive semidefinite since it is a convex combination of two positive semidefinite matrices.j need to define A

It is known that in regions of high curvature, the false position algorithm may be initially slow to converge. In this situation we fall back on grid search, which may initially converge faster to the solution. The objective function is convex in S_i, so simple evaluation of the objective at the midpoint of the current refinement is often enough to establish the next digit in the binary representation of $\eta \in [0, 1]$. When there is not enough information using convexity and knowledge of the variational lower bound at the end points and mid point, the derivative is used to inform the next binary digit in the representation.

Updating V Let

$$Z_h = \frac{3N}{2} + \frac{1}{2}\log|K^{-1}| - \frac{1}{2}m^T K^{-1} m + \frac{1}{2}\log|V| - \frac{1}{2}\operatorname{tr}(K^{-1}V) + \sum_{i=1}^{N} h_i(y_i \mid q_i, b_i, S_i) \tag{2.22}$$

Observe that b and S appear only in the last term, so the optimization in the previous section equivalently applies for the variational objective Z_h. Following the advice of Chai (2012), we take the fixed point update for V to be

$$V^* = V^{\text{fx}} = (K^{-1} + W)^{-1} \tag{2.23}$$

where $W = \operatorname{diag}(W_1, \ldots, W_N)$ and $W_i = S_i + V_i^{-1}$. As with before, the fixed point update may fail to improve the variational lower bound, and Newton-Raphson may fail to preserve positive semidefiniteness, so false position, and in the presence of high curvature grid search, are used to identify an optimal update which necessarily improves the variational lower bound Z_h.

Parameter Initialization Each b_i is initialized as a random draw from the 3 dimensional Dirichlet distribution, and W, V, m are defined as

$$W_i = \frac{\gamma}{3}I - \frac{\gamma}{9}11^T \tag{2.24}$$

$$V = (K^{-1} + W)^{-1} \tag{2.25}$$

$$m = VWa \tag{2.26}$$

with $a = \gamma(y + \frac{y-1}{2})$ and $S_i = V_i^{-1} + W_i$.

Prediction

Though the model is trained on the credit labels in range $\{0, 1, 2\}$, it is of primary interest to us to distinguish between the lowest group, and either of the upper two. Given the value of m, discovered through the variational learning procedure described in the previous subsection, define

$$m^* = K_{x^*,x}K^{-1}m \tag{2.27}$$

Predictions are obtained through population matching with the frequency of observed 0 labels in the training data and the most likely test responses to be labeled zero according to normalized probabilities $p(z^* = 0 \mid x^*, Y, X)$ where in general,

$$p(z^* = j \mid x^*, Y, X) = \frac{\exp(m^* e_{j+1})}{\sum_{k=0}^{2} \exp(m^* e_{k+1})} \tag{2.28}$$

2.4 Results and Discussion

In general, results show that there is indeed a link between differences in personality and propensity to develop different levels of credit score, even over such a long time horizion. The nature of the functional relationship that connects them appears to vary across the space of personalities, and may more fundamentally change structure at the extremes. Future research which tracks prospective credit development in the

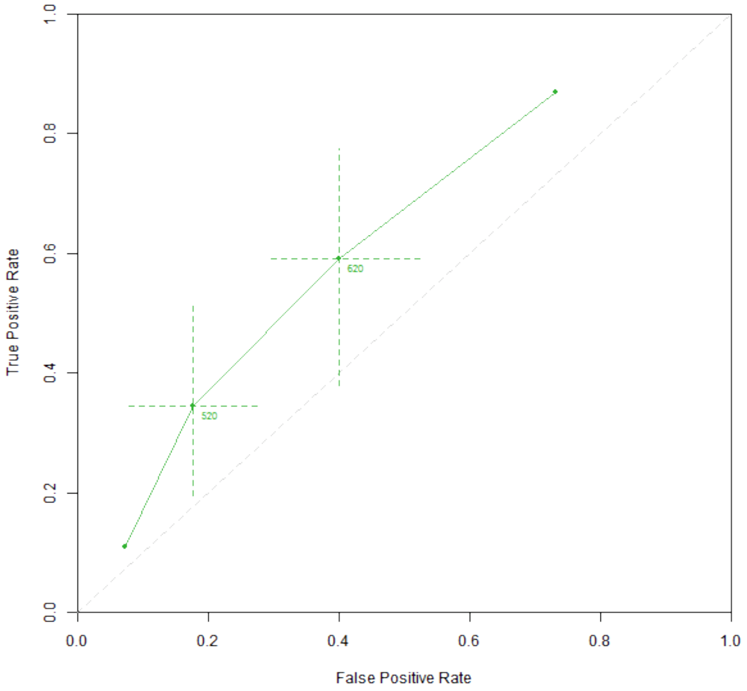


Figure 2.5: Threshold averaged ROC curve for MLGP.

short term after a personality battery could help identify the strength of this link for the purposes of more optimally extending credit to those with limited credit histories. This initial research is encouraging towards that prospect, however this data does not speak directly to that hypothesis.

With regards to interpreation of the results, it is likely that false positives are more deleterious for lenders, in that they represent people who were thought to be creditworthy but actually were not. With this in mind, it is reassuring that the false positive 95% coverage intervals are generally much tighter than the true positive 95% coverage bands.

It is possible to incorporate this directly into the optimization procedures undertaken during model fitting with an appropriately defined cost function, however, it is in general difficult to drive this directly from credit score. It is well known that ‘higher credit score is better’, but it is difficult, without knowledge of a specific lending agency’s pricing algorithms to translate credit score into expected return which could then be used as a proxy for both relative cost, and the cost of misclassifications.

It is however possible to obtain any (true positive,false positive) combination which is a convex combination of the rates achieved by the classifiers at any given threshold obtained here by defining a new classifier which is defined as a mixture distribution of the available classifiers. Furthermore, the mixture weights can then be optimized with the intent to minimize expected loss given specification of a cost function for misclassifications after the models’ parameter estimation procedures are independently completed as well.

The elastic net penalized multinomial logistic regression, and support vector machine, approaches demonstrated some additional negative correlation in predictions, which can be seen in their wide coverage bands for true positives. This could partially be an artifact of the cross-validation procedure, since for the regression setting, the removal of a data point response pair into the test set which is positively linked with a collection of covariates, will in general cause the estimate of the coefficients corresponding to those covariates based on the training set to decrease. This effect can in principle be quite large, and reducing the value of K can help to guard against it, however reducing K also reduces the volume of training datasets. Since this has a negative effect on the quality of predictions, the results herein can be thought of as a lower bound to potential future studies, which can target larger audiences to achieve reasonable learning with smaller K.

As indicated by figure (2.6), there is no one method which dominates everywhere. Often the optimal method further depends on expectations about the population distribution as well as the penalty for misclassifications. For example at the 520 level, either MLGP or naïve Bayes are the best observed classifiers, but without knowing the more about misclassification cost, its not possible to assert that one would necessarily lead to better expected performance than the other. Similarly these methods perform better at the extremes, perhaps owing to their propensity to predict based on the joint distribution of personality and credit, with this specified in the case of MLGP through the frequency matching between the training and testing subsets.

It is important to note that both MLGP and naïve Bayes had relatively tight coverage intervals, wheras elastic net penalized logistic regression and SVM often had predictions which varied wildly in accuracy. In general, this would significantly count against the utility of the algorithm, however SVM performed quite well at the 620 credit cut off, this is especially important as 620 is a potentially likely candidate cut off for certain loan opportunities.

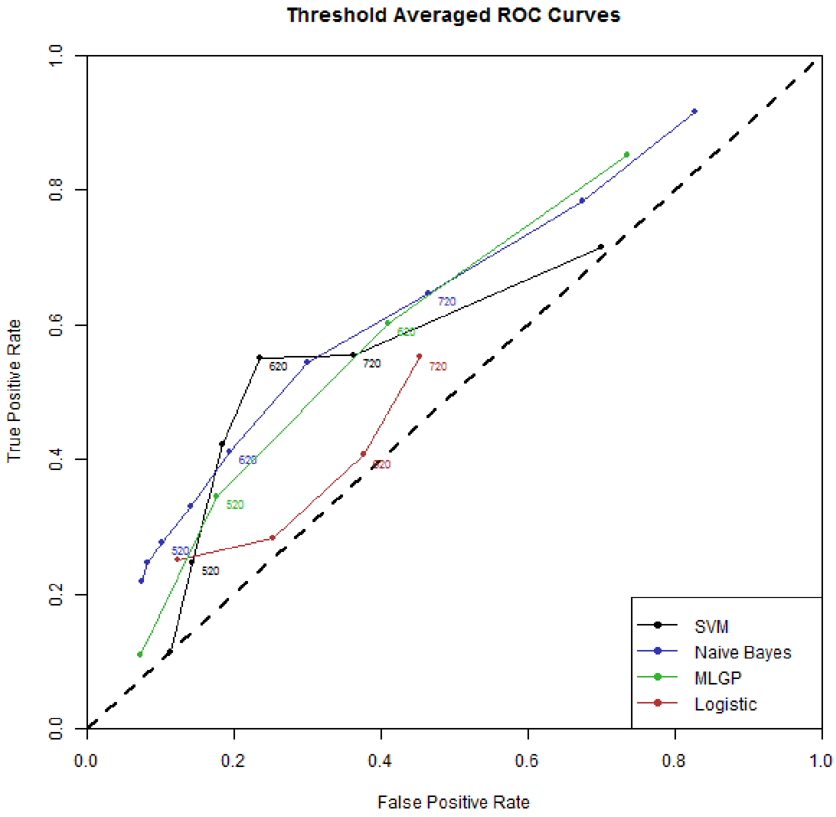


FIGURE 2.6: Overlay of the threshold averaged ROC curves for elastic net logistic regression, naïve Bayes, SVM, and MLGP, demonstrating the predictive link between personality and future credit development as well as the fact that optimal performance is achieved by interpolations of different classifiers depending on the value of t chosen for the credit partitions cutoffs, τ_1 and τ_2 from equation (2.1).

3

Completion Time

3.1 Introduction

In recent years, datasets have been growing in size at an astounding rate, and simultaneously, algorithms are becoming increasingly sophisticated in an attempt to identify signals in these increasingly large, noisy datasets. In response, it has been increasingly popular to adapt algorithms to run simultaneously on multiple machines to help alleviate this computational bottleneck. However, in practice many researchers don't have a large network of computers laying around at their disposal.

Public cloud service providers have risen to fill this niche. In public clouds, users can gain temporary access to computational resources in exchange for a time-based fee. This avoids the need to acquire and maintain the network, which is most useful only at the data-mining phase of a project. Generally speaking, users can pay a higher rate to guarantee they hold a machine for as long as they require it, or risk possibly losing a *spot instance* machine before their algorithm completes in exchange for a lower cost. From the perspective of cloud providers, spare capacity often represents loss of potential revenue. To encourage utilization of resources which remain free after purchases at the standard rate, spare capacity is divided into spot instances which are (possibly virtual) machines, following a dynamic market price. The specific bidding and pricing mechanisms can vary across cloud providers, however optimization from users' perspectives essentially interpolates between risk–reward tolerances, with low bids being more likely to fail in the near future (Zhang et al., 2011). Optimization using spot instances to reduce costs and accelerate computation has been explored in Cumulon's spot price strategies (Huang et al., 2014).

Due its parallelizability, matrix multiplication has become ubiquitous in statistical analysis of big data, for which cloud resources are increasingly being used. Many cloud applications either directly involve multiplications of large, real matrices, or iteratively perform matrix multiplications.

In this paper, we develop a statistical emulator for the completion time of matrix multiplication jobs. Algorithms which feature independent matrix multiplications can be directly optimized using this framework, while iterative procedures can be piecewise optimized in steps. For iterative algorithms, it is typical to perform exploratory data analysis to help understand what kind of convergence rates which are reasonable to expect. Using the information gained in these trial runs, it is often the case that the end user can use the step-wise optimal solutions to estimate how the run time behavior of the combined iterative procedure.

In this work, we explore the uncertainty and optimization of the completion time of matrix multiplication within the Cumulon framework because it is often more efficient for the matrix multiplications found in statistical applications than the MapReduce framework as can be seen in Figure (3.2). This is because MapReduce tasks generally need to be given disjoint subsets of the input, which often results in unnecessarily replication of many subsets of the data. A simple example of MapReduce in SystemML with $f_l = 1$ is provided below in figure (3.1). Cumulon avoids this by storing matrices in disjoint, contiguous *tiles* which can have either one or more concurrent readers, or a single writer. Furthermore, Cumulon is built on top of Hadoop and supports both sparse and dense matrix multiplications, which are both of primary interest to statistical applications.

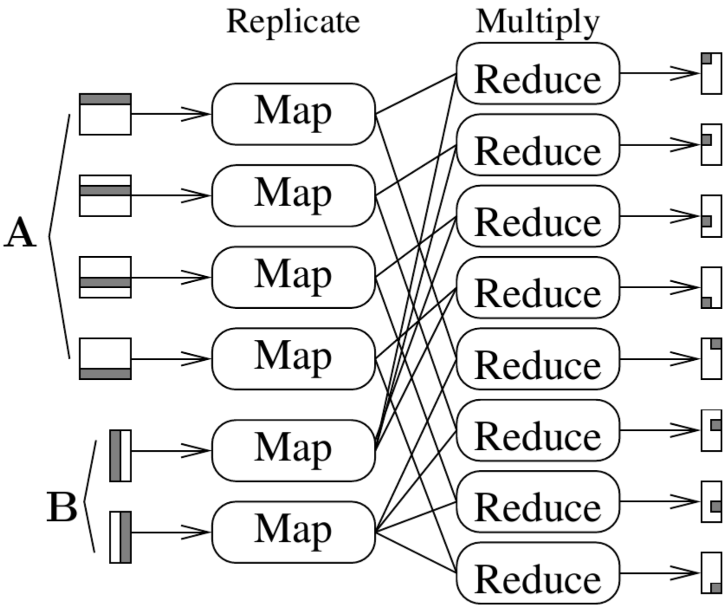


FIGURE 3.1: Diagram from Huang et al. (2013), illustrating the behavior of the MapReduce, RMM strategy within SystemML. In this case $f_l = 1$, so that the columns of A and the rows of B have no splits, allowing MapReduce to avoid a second call for the additional sum operations required to obtain blocks of the output; however the restriction that $f_l = 1$ negatively impacts performance as more balanced submatrix sizes tend to yield increased performance efficiency for sufficiently large input matrices. (Huang et al., 2013)

3.2 Deconstructing Completion Time

Each matrix multiplication *job*, $C = AB$, executes in parallel, independent *tasks*. The input matrices $A \in \mathbf{R}^{m \times l}$ and $B \in \mathbf{R}^{l \times n}$ are conceptually divided into *submatrices* according to *split factors* which form the triplet $[f_m, f_l, f_n]$, indicating how many approximately equally sized partition elements are allocated out of A and B for the first, conforming, and last dimensions of the input matrices A and B. Each task

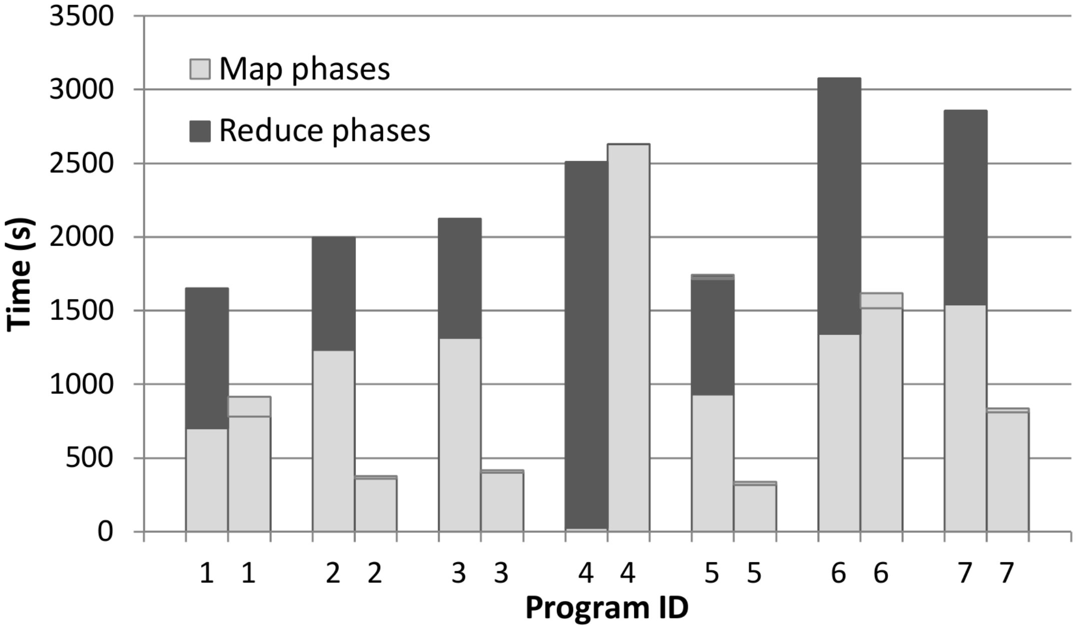


FIGURE 3.2: Diagram illustrating the relative performances between MapReduce within SystemML and the Cumulon strategy from Huang et al. (2013). For each program ID, the left bar shows MapReduce performance, and the right bar shows Cumulon performance.

reads a submatrix pair and executes identical code to generate disjoint subsets of the output C. A *scheduler* allocates the $f_m \cdot f_l \cdot f_n$ tasks to *slots* which service one task at a time, where each of the N machines in the network is configured into S slots, so that there are NS slots available in total. Reads and writes are performed at the *tile* level, with size $\mathsf{B} \times \mathsf{B}$. A common setting for B is 2048, which leads to 32 MB tiles, assuming the matrix consists of double precision floating point numbers. For simplicity, we consider homogeneous networks of machines, where each node is roughly identical with respect to how well it performs the steps of the matrix multiplication procedure. Cumulon supports cluster switching, so networks can be heterogeneous across time, and can be estimated by independently benchmarking each machine type with relevant workloads.

There are many factors impacting completion time, but in practice, workloads are reasonably large, since the reason they have been brought to the cloud is that they were impractically large to otherwise resolve. This typically leads the cost of actually running the Hadoop jobs to dominate the other components of completion time, barring excessive cluster switching et cetera. The multiplication physical operator parameters include the split factors as well as other parameters which indicate how the input matrices are read into memory. Generally, these parameters correspond to a trade-offs between CPU utilization and memory usage and efficiency. Optimizing completion time for matrix multiplication therefore requires optimization over the CPU/memory trade-off, as well as optimization over network composition, provisioning and configuration settings. Manually optimizing these quantities is a daunting task, which motivates the need for automatic methods for identifying efficient strategies.

Instead of building one large model for job completion time, submodels are built at the task level for I/O time and computation time exploiting conditional independence relationships between respective quantities to obtain a more parsimonious model which will be more amenable to task and job completion time optimization. Task time is taken to be the sum of I/O time, computation time, and initialization time. Once task level models are obtained, they can be convolved to obtain the distribution of job completion times.

3.3 Task Input Time

Let γ denote the Hadoop Distributed File System replication factor. Then given a collection of observed input times, let n_{I} denote the number of inputs for a task are which respectively have volumes $\{\mathsf{V}_{\mathsf{I}_1}, \ldots, \mathsf{V}_{\mathsf{I}_{n_\mathsf{I}}}\}$ and inverse transfer rates $\{\mathcal{D}_{\mathsf{I}_1}, \ldots, \mathcal{D}_{\mathsf{I}_{n_\mathsf{I}}}\}$. The total input time for a task is then modeled as

$$\mathsf{I}_{\mathsf{time}} = \sum_{i=1}^{n_\mathsf{I}} \mathsf{V}_{\mathsf{I}_i} \nu_{\mathsf{I}_i}^{-1} + \epsilon_\mathsf{I} \qquad (3.1)$$

Note that $\{\mathsf{V}_{\mathsf{I}_1}, \ldots, \mathsf{V}_{\mathsf{I}_{n_\mathsf{I}}}\}$ can be analytically derived from input split size, input sparsity and relevant operator parameter settings. The $\nu_{\mathsf{I}i}^{-1}$ parameters follow a Gaussian mixture model distribution where component membership is determined based on whether or not the read was local or remote.

Recall that matrices are divided into submatrices, which are distributed to tasks as needed to complete portions of the output. These submatrices are made up of collections of smaller memory blocks, so the decision to load both submatrices entirely into memory, or to stream one of them in by blocks impacts the likelihood equations for the inverse transfer rates for reads as streaming incurs an additional overhead expense per block. Therefore using additional block level regression parameters for streamed matrices will improve the accuracy of predictions.

Both of these input schemes were explored in Huang et al. (2013). Generally speaking, it is more straightforward to read both submatrices entirely into memory. This approach yields good CPU utilization, but also requires more memory, imposing a stricter size constraint on submatrices to avoid thrashing. Thrashing is a condition where virtual memory is swapped quickly to and from physical memory, this can preoccupy system resources to such an extent that it can significantly hinder CPU utilization Zhang et al. (2010). On the other hand, if one submatrix is streamed into memory at a specified granularity as needed, memory requirements reduce and the range of feasible submatrix sizes becomes more flexible at the cost of lower CPU utilization.

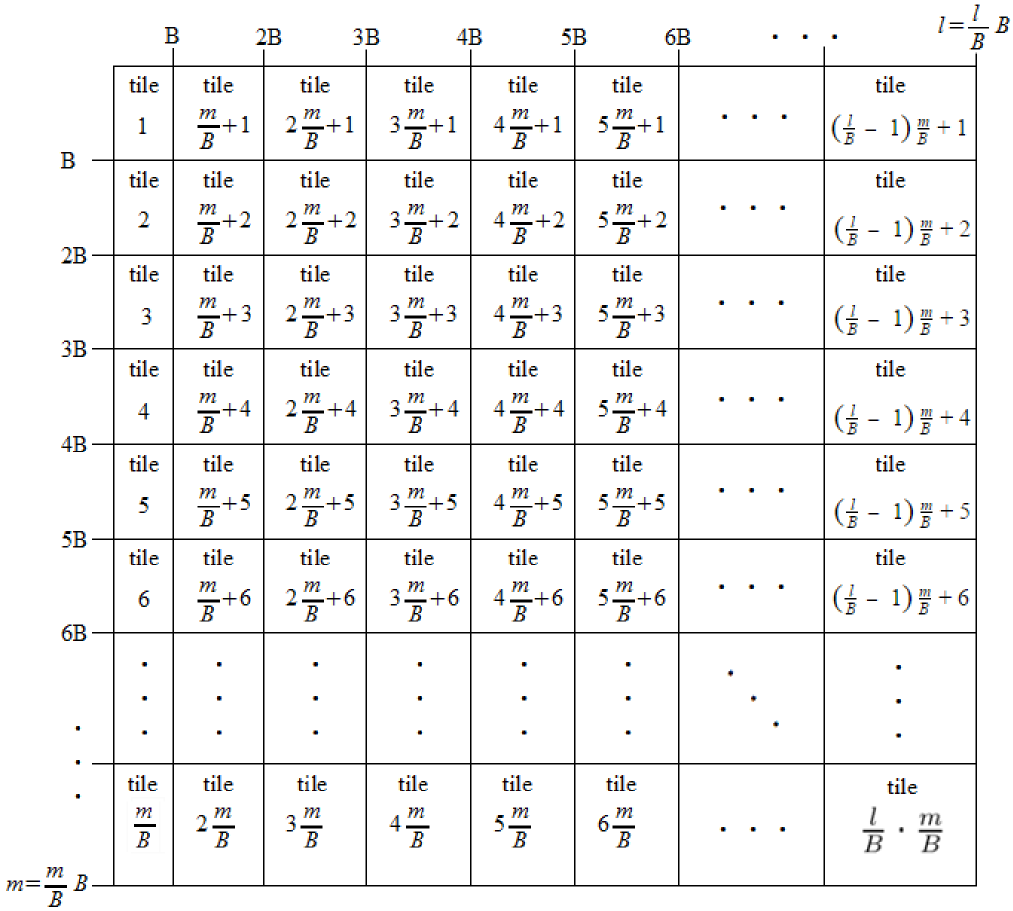
The likelihood equations for the inverse transfer rates for reads are specified as

$$\nu_{\mathsf{I}i}^{-1} \mid \beta_\mathsf{I}, \sigma_\mathsf{I}^2, \ell_{\mathsf{I}i} = 1 \sim \mathsf{N}([1, \ell_{\mathsf{B}_i}]\beta_{\mathsf{I}_1}, \sigma_{\mathsf{I}_1}^2) \tag{3.2}$$

$$\nu_{\mathsf{I}i}^{-1} \mid \beta_\mathsf{I}, \sigma_\mathsf{I}^2, \ell_{\mathsf{I}i} = 0 \sim \mathsf{N}([1, \ell_{\mathsf{B}_i}, \mathcal{N}_{\mathsf{I}_i}]\beta_{\mathsf{I}_0}, \sigma_{\mathsf{I}_0}^2) \tag{3.3}$$

$$\ell_{\mathsf{I}i} \sim \mathsf{Bernoulli}(\gamma/N) \tag{3.4}$$

Where $\ell_{\mathsf{I}i}$ is a binary indicator of the event that the i^{th} read is local, ℓ_{B_i} is an indi-



	B	2B	3B	4B	5B	6B	· · ·	$l=\frac{l}{B}B$
B	tile 1	tile $\frac{m}{B}+1$	tile $2\frac{m}{B}+1$	tile $3\frac{m}{B}+1$	tile $4\frac{m}{B}+1$	tile $5\frac{m}{B}+1$	· · ·	tile $(\frac{l}{B}-1)\frac{m}{B}+1$
2B	tile 2	tile $\frac{m}{B}+2$	tile $2\frac{m}{B}+2$	tile $3\frac{m}{B}+2$	tile $4\frac{m}{B}+2$	tile $5\frac{m}{B}+2$	· · ·	tile $(\frac{l}{B}-1)\frac{m}{B}+2$
3B	tile 3	tile $\frac{m}{B}+3$	tile $2\frac{m}{B}+3$	tile $3\frac{m}{B}+3$	tile $4\frac{m}{B}+3$	tile $5\frac{m}{B}+3$	· · ·	tile $(\frac{l}{B}-1)\frac{m}{B}+3$
4B	tile 4	tile $\frac{m}{B}+4$	tile $2\frac{m}{B}+4$	tile $3\frac{m}{B}+4$	tile $4\frac{m}{B}+4$	tile $5\frac{m}{B}+4$	· · ·	tile $(\frac{l}{B}-1)\frac{m}{B}+4$
5B	tile 5	tile $\frac{m}{B}+5$	tile $2\frac{m}{B}+5$	tile $3\frac{m}{B}+5$	tile $4\frac{m}{B}+5$	tile $5\frac{m}{B}+5$	· · ·	tile $(\frac{l}{B}-1)\frac{m}{B}+5$
6B	tile 6	tile $\frac{m}{B}+6$	tile $2\frac{m}{B}+6$	tile $3\frac{m}{B}+6$	tile $4\frac{m}{B}+6$	tile $5\frac{m}{B}+6$	· · ·	tile $(\frac{l}{B}-1)\frac{m}{B}+6$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮
$m=\frac{m}{B}B$	tile $\frac{m}{B}$	tile $2\frac{m}{B}$	tile $3\frac{m}{B}$	tile $4\frac{m}{B}$	tile $5\frac{m}{B}$	tile $6\frac{m}{B}$	· · ·	tile $\frac{l}{B}\cdot\frac{m}{B}$

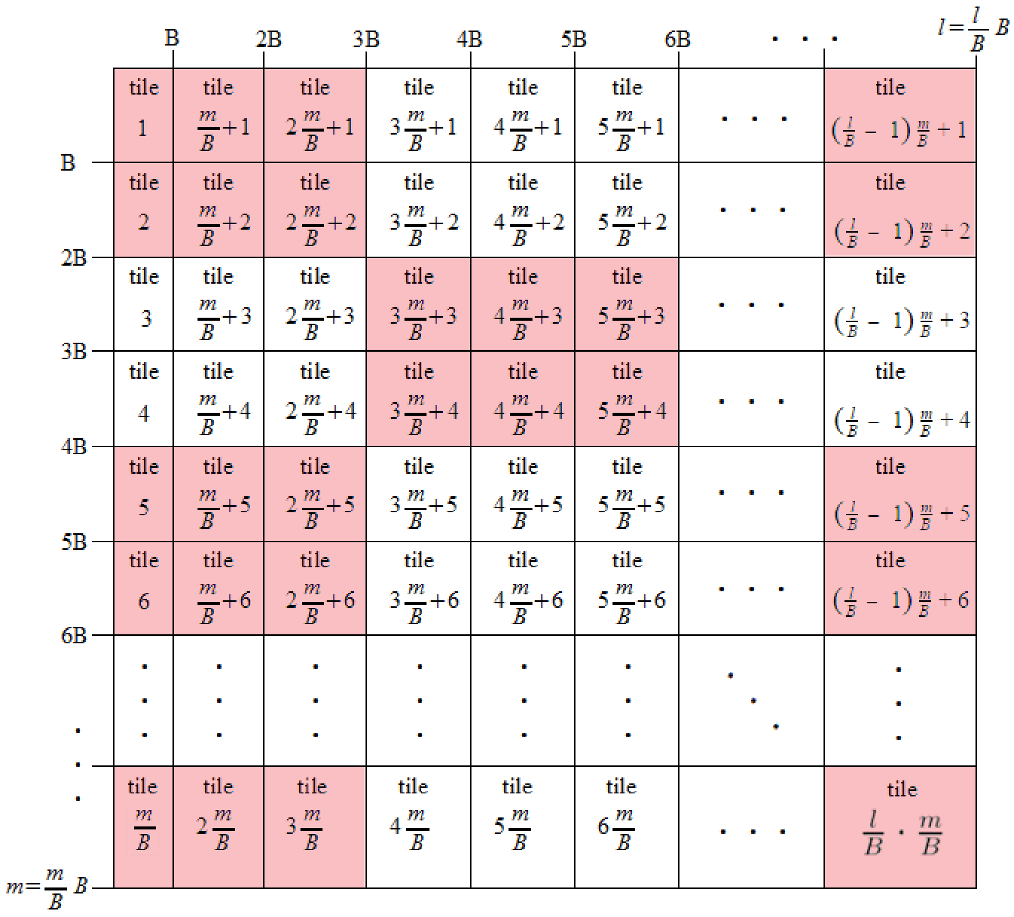
FIGURE 3.3: This figure illustrates the first level of regular tiling overlay on the input matrix $A \in \mathbf{R}^{m\times l}$, which is determined by the fundamental unit of storage, the block of size $\mathsf{B} \times \mathsf{B}$. Data is stored in column major order, leading to the observed pattern of indices.

cator for the event that the i^{th} read originated from a streamed submatrix, and $\mathcal{N}_{\mathsf{l}_i}$ denotes the size of the i^{th} cluster used for benchmarking the inverse transfer rates for reads. Note that additional regression parameters for matrix B are included here to reflect the fact that its submatrices are streamed into memory for the multiplication algorithm that this emulator is based on. This model can easily collapse back down to the case where both submatrices are loaded entirely into memory by setting these specialized coefficients to zero.

We use a non-informative prior with the residual variances a priori following limiting $\mathsf{Gamma}(0,0)$ distributions, and flat priors on the regression coefficients. If the indicators for the events that each requested tile is local relative to its task's execution location are explicitly collected, then equation (3.4) need not be considered for developing the local or remote read posterior emulators based on the training data, and would only play a role in the frequency by which a future read was emulated via sampling from either one of these conditional emulators. However, since it is often impractical to explicitly collect the indicator for the event that a requested tile is local relative to its task's execution location, distributional assumptions on the relative location of a tile are appropriate in order to proceed forward with obtaining functional forms of the local and remote posterior emulators in practice.

The independent Bernoulli product distribution for the vector of read locality indicators is a simplifying assumption to this analysis, which is akin to a first order approximation. To explain, in general, the storage granularity for a Cumulon job need not be identical to the tile size. Commonly multiple adjacent tiles are jointly allocated to storage across the network. This decision can be viewed as both simplifying the data allocation procedure, and more importantly, as a method of marginally improving the efficiency of data requests, as some overhead is incurred for each network request. The impact of this data storage scheme on the distribution of the indicators for read locality is most easily observed through graphical representations of the input matrix, as shown in figures (3.4) and (3.5).

While it is possible to propose a model where all of the read indicators learned for every location in the input matrices, and one such model that would be able to cope with the complicated orbits inherent in the data allocation and query procedures, this is incredibly far from being parsimonious to work with. Running the entire job once would only give one sample point for each of the distributions, implying that, even accounting for some information sharing across distributions, it might be

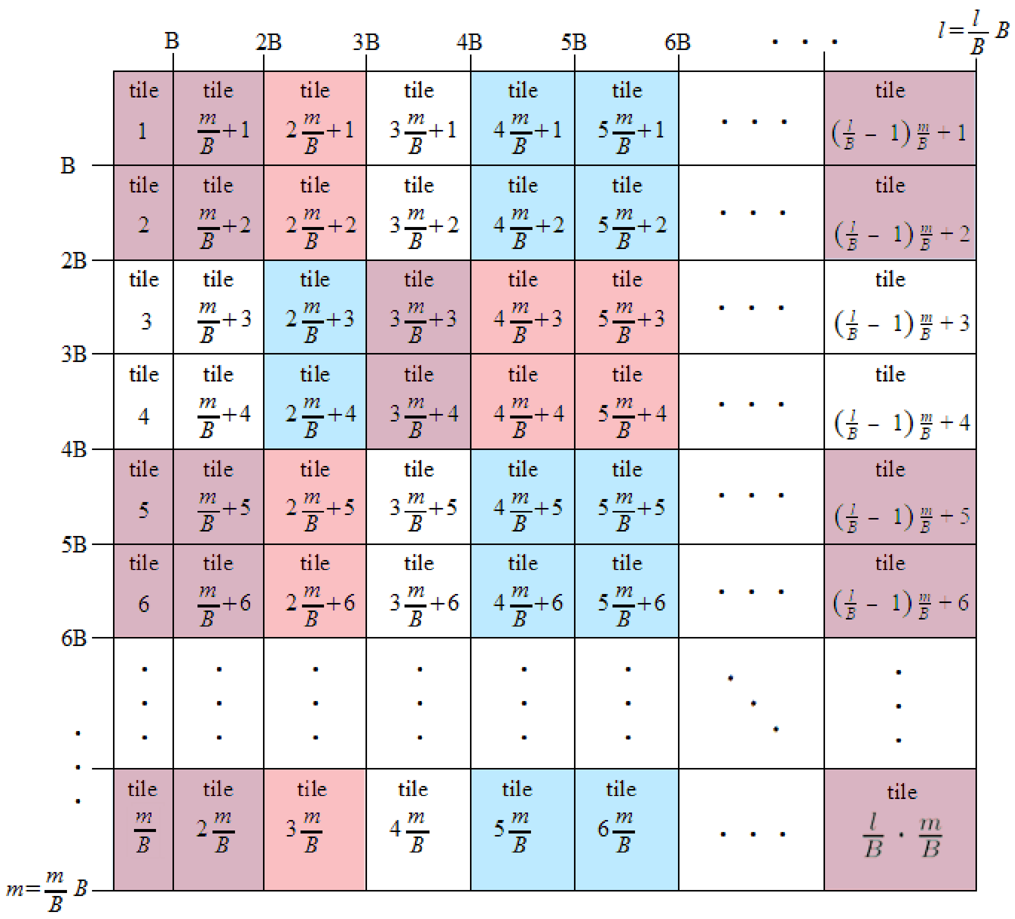


	B	2B	3B	4B	5B	6B	. . .	$l=\frac{l}{B}B$
B	tile 1	tile $\frac{m}{B}+1$	tile $2\frac{m}{B}+1$	tile $3\frac{m}{B}+1$	tile $4\frac{m}{B}+1$	tile $5\frac{m}{B}+1$	. . .	tile $(\frac{l}{B}-1)\frac{m}{B}+1$
2B	tile 2	tile $\frac{m}{B}+2$	tile $2\frac{m}{B}+2$	tile $3\frac{m}{B}+2$	tile $4\frac{m}{B}+2$	tile $5\frac{m}{B}+2$	. . .	tile $(\frac{l}{B}-1)\frac{m}{B}+2$
3B	tile 3	tile $\frac{m}{B}+3$	tile $2\frac{m}{B}+3$	tile $3\frac{m}{B}+3$	tile $4\frac{m}{B}+3$	tile $5\frac{m}{B}+3$	. . .	tile $(\frac{l}{B}-1)\frac{m}{B}+3$
4B	tile 4	tile $\frac{m}{B}+4$	tile $2\frac{m}{B}+4$	tile $3\frac{m}{B}+4$	tile $4\frac{m}{B}+4$	tile $5\frac{m}{B}+4$	. . .	tile $(\frac{l}{B}-1)\frac{m}{B}+4$
5B	tile 5	tile $\frac{m}{B}+5$	tile $2\frac{m}{B}+5$	tile $3\frac{m}{B}+5$	tile $4\frac{m}{B}+5$	tile $5\frac{m}{B}+5$	. . .	tile $(\frac{l}{B}-1)\frac{m}{B}+5$
6B	tile 6	tile $\frac{m}{B}+6$	tile $2\frac{m}{B}+6$	tile $3\frac{m}{B}+6$	tile $4\frac{m}{B}+6$	tile $5\frac{m}{B}+6$	. . .	tile $(\frac{l}{B}-1)\frac{m}{B}+6$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮
$m=\frac{m}{B}B$	tile $\frac{m}{B}$	tile $2\frac{m}{B}$	tile $3\frac{m}{B}$	tile $4\frac{m}{B}$	tile $5\frac{m}{B}$	tile $6\frac{m}{B}$	. . .	tile $\frac{l}{B}\cdot\frac{m}{B}$

FIGURE 3.4: This figure illustrates the interaction of the submatrix tiling induced by fixed split factors with that of the block storage unit. Given that $A \in \mathbf{R}^{m\times l}$, and split factors for A are fixed over the lifetime of the job to f_m, f_l, the submatrix tiling shares edges with the first tiling at vertically locations given by multiples of $\frac{m}{f_m} = \frac{m}{\mathsf{B}f_m}\mathsf{B}$ and horizontally locations given by multiples of $\frac{l}{f_l} = \frac{l}{\mathsf{B}f_l}\mathsf{B}$. Data is stored in column major order, leading to the observed pattern of indices.

necessary to run an impressive amount of tests in order to reasonably explore the joint distribution. This is far from the spirit of this work, where we would like to assess how expensive reads will be for a deployment plan, without actually enacting the labor of executing that plan in its entirety.

With this in mind, a point of consideration is that the volumes of dependent tiles in a task can vary significantly as the submatrix windows slide along the input



tile 1	tile $\frac{m}{B}+1$	tile $2\frac{m}{B}+1$	tile $3\frac{m}{B}+1$	tile $4\frac{m}{B}+1$	tile $5\frac{m}{B}+1$	. . .	tile $(\frac{l}{B} - 1)\frac{m}{B}+1$
tile 2	tile $\frac{m}{B}+2$	tile $2\frac{m}{B}+2$	tile $3\frac{m}{B}+2$	tile $4\frac{m}{B}+2$	tile $5\frac{m}{B}+2$	. . .	tile $(\frac{l}{B} - 1)\frac{m}{B}+2$
tile 3	tile $\frac{m}{B}+3$	tile $2\frac{m}{B}+3$	tile $3\frac{m}{B}+3$	tile $4\frac{m}{B}+3$	tile $5\frac{m}{B}+3$	. . .	tile $(\frac{l}{B} - 1)\frac{m}{B}+3$
tile 4	tile $\frac{m}{B}+4$	tile $2\frac{m}{B}+4$	tile $3\frac{m}{B}+4$	tile $4\frac{m}{B}+4$	tile $5\frac{m}{B}+4$	. . .	tile $(\frac{l}{B} - 1)\frac{m}{B}+4$
tile 5	tile $\frac{m}{B}+5$	tile $2\frac{m}{B}+5$	tile $3\frac{m}{B}+5$	tile $4\frac{m}{B}+5$	tile $5\frac{m}{B}+5$	. . .	tile $(\frac{l}{B} - 1)\frac{m}{B}+5$
tile 6	tile $\frac{m}{B}+6$	tile $2\frac{m}{B}+6$	tile $3\frac{m}{B}+6$	tile $4\frac{m}{B}+6$	tile $5\frac{m}{B}+6$	. . .	tile $(\frac{l}{B} - 1)\frac{m}{B}+6$
⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮
tile $\frac{m}{B}$	tile $2\frac{m}{B}$	tile $3\frac{m}{B}$	tile $4\frac{m}{B}$	tile $5\frac{m}{B}$	tile $6\frac{m}{B}$	. . .	tile $\frac{l}{B}\cdot\frac{m}{B}$

FIGURE 3.5: This figure illustrates the interaction of the submatrix tiling induced by fixed split factors with that of the tiling generated from joint allocation of data to storage locations, and shows these overlaid over the tiling generated from the fundamental block unit of storage. In terms of block units, the combined pattern generated by the submatrix tiling of sizes, $\frac{m}{\mathsf{B}f_m} \times \frac{l}{\mathsf{B}f_l}$, in this case 2×3, and joint storage allocation tiling of sizes, γ_1', γ_2', in this case 2×2, is repeated every area of size $\left(\frac{m}{\mathsf{B}f_m}\gamma_1'\right) / \left(\text{factors}\left(\frac{m}{\mathsf{B}f_m}, \gamma_1'\right)\right) \times \left(\frac{l}{\mathsf{B}f_l}\gamma_2'\right) / \left(\text{factors}\left(\frac{l}{\mathsf{B}f_l}, \gamma_2'\right)\right)$, in this case 2×6, beginning at the top left.

matrices, however since the storage granularity and submatrix dimensions are typically fixed over a matrix multiplication job, the pattern by which tile groups are disseminated to tasks for execution is repeated over areas of the input matrices corresponding to the prime factors shared by submatrix sizes and the tile group storage sizes.

More specifically, the set of unique tile bundles are arranged in an additional regular square tiling which takes the form of a mask over the first two tilings, sharing boundaries at every multiple of the storage dimensions divided by all common prime factors between the relevant dimension of storage granularity and submatrix. A consequence of this is that the frequency of each submatrix type over the whole input matrix, converges to it's frequency within the mask tiles as the input matrices grow in size.

It is possible to further take the data allocation procedure into account, by allowing dependence in the joint distribution of read locality for completely encompassed tile groups. This reflects the idea that while relatively uniform distribution of data over the network can optimize read times by decreasing the distance to the nearest copy of a desired tile, it also implies slight negative correlation in the indicators for read locality over disjoint tile groups. The joint distribution for split tile groups may be influenced by their location within the mask tile, which can be correlated with the event that the tasks handling the split tile group have overlapping execution times, especially for jobs with small split factors.

The storage procedure divides the input matrices into regular square tilings, and the allocation of submatrices to tasks further overlays another square tiling with identical orientation over the input matrix. Provided the submatrices aren't dominated in size by the jointly allocated tile groups, data from each tile group will either be entirely handled by one task, or will be divided between either two or four tasks, which may or may not have overlapping execution times. The probability that a split tile group is local relative to any one of the tasks is γ/N when execution locations are not local to each other, but even in this case the joint distribution can be complicated, depending on the temporal relationship between the execution times of the tasks to which the split tile group is allocated. For overlapping tasks, it is natural to consider the joint distribution as hypergeometric, where the number of successes in the population is given by the number of execution locations which are local relative to any of the γ storage locations for the tile group under consideration, as could be the case with networks of virtual machines or of machines with multiple logical threads, however as tasks complete, 'successes' are re-released into the population, leading the joint distribution over time to be given by a mixture of hypergeometric distributions, where the mixture weights are defined by the scheduler behavior. That is to say, we additionally expect a slight negative correlation for the relative read locality indicators for multiple tasks handling the components of split tile groups in addition to the negative correlation between tile groups within a task.

Given this, we can provide an upper bound on the ℓ_2 error between the 'true' distribution and the independent Bernoulli approximation for split tile groups, given N and γ, by considering that the worst-case for the Bernoulli approximation occurs when a tile block is disseminated to four tasks with pairwise overlapping execution times. This error rapidly decreases to zero as the cluster size grows, representing the fact that while there is a finite population of local execution locations to the tile group under consideration, it is very unlikely to sample enough of them to be in a region of the support where the distributions are significantly different, even for modest cluster sizes. Furthermore, the expected number of local reads are equivalent under both distributions, where the binomial probability of success is calibrated to the initial probability of success in the hypergeometric setting, thereby justifying the first order approximation interpretation. The convergence of these distributions is illustrated in the figure (3.6) below.

Another interesting feature of the Bernoulli product distribution, is that it leads to a non-linear average predicted inverse transfer rate, which closely resembles the observed average inverse transfer rates. The rate of convergence to the remote inverse transfer rate is initially fast, but decays quickly, leading to a steep yet smooth elbow

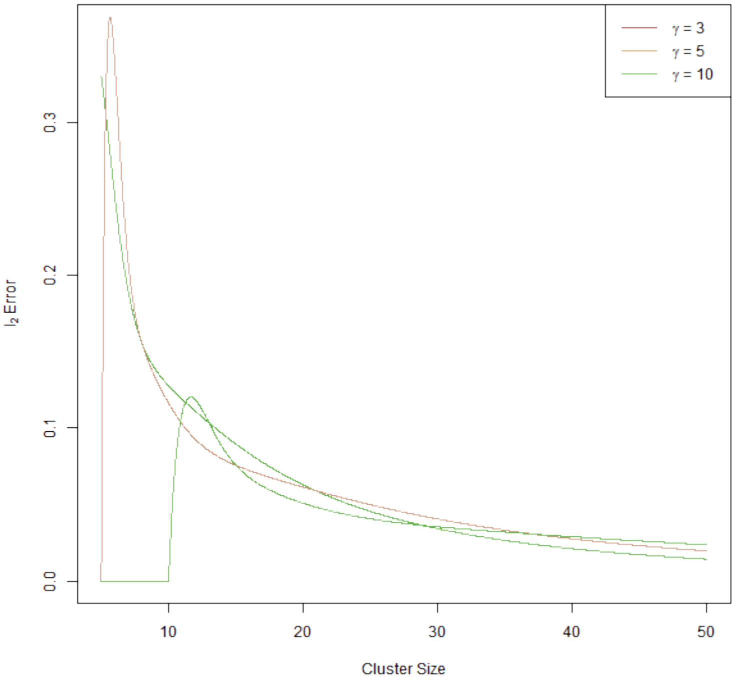


Figure 3.6: Upper bound on the error between Hypergeometric sampling of read locality for split tile groups and the first order binomial approximation by ℓ_2 norm given cluster size, N, and the Hadoop file system replication factor, γ.

in the average predicted inverse transfer rates for cluster sizes just above the $N = \gamma$ threshold. It is notable to observe that due to the nuances of queries in the cloud, it is not always the case that a local copy of a tile is fetched if one exists. This leads to occasional deviations from this general rule at small cluster sizes. Evidence of this behavior is shown in figure (3.7) from Huang et al. (2013) below.

Since expectation is a linear operator, the conditional expectations of the mixture distribution components from the likelihood can be interpolated according to their relative probabilities to obtain an expression for the average inverse read times which are a priori built into the modeling. Also note that for conciseness, most parameters are omitted from the conditional distributions. The effect of adding an additional

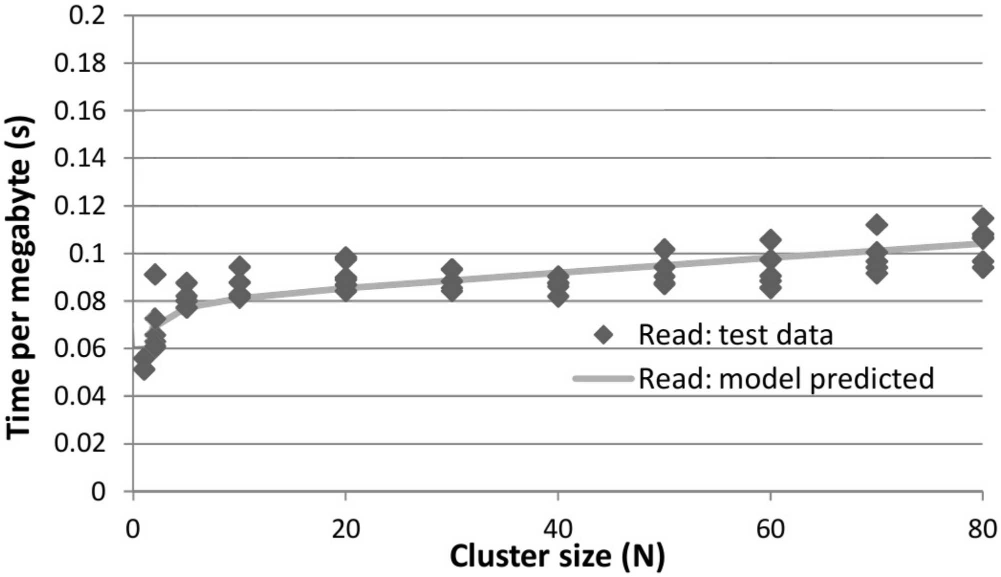


FIGURE 3.7: Graphical summary of the behavior of inverse transfer rates for reads as cluster size is increased in Cumulon with γ fixed at 3 from Huang et al. (2013). For interpretation of this graph in the context of the proposed mixture distribution for read times, it is relevant to note that the times in this graph are at the task I/O level, and not the tile I/O level. This implies that the mixture distribution manifests itself in the skewness of the observations at a given level of N, which is indeed visually apparent. Observe that the law by which conditional averages increase approximately follows equation (3.5)

machine into the cluster is approximated by

$$\begin{aligned}
\mathbb{E}(\nu_{\mathsf{l}i}^{-1} \mid N+1) - \mathbb{E}(\nu_{\mathsf{l}i}^{-1} \mid N) =& \mathbb{E}(\nu_{\mathsf{l}i}^{-1} \mid \ell_{\mathsf{l}i} = 1)\frac{\gamma}{N} + \mathbb{E}(\nu_{\mathsf{l}i}^{-1} \mid \ell_{\mathsf{l}i} = 0)\left(1 - \frac{\gamma}{N}\right) \\
&- \mathbb{E}(\nu_{\mathsf{l}i}^{-1} \mid \ell_{\mathsf{l}i} = 1)\frac{\gamma}{N+1} \\
&- \mathbb{E}(\nu_{\mathsf{l}i}^{-1} \mid \ell_{\mathsf{l}i} = 0)\left(1 - \frac{\gamma}{N+1}\right) \\
=& \mathbb{E}(\nu_{\mathsf{l}i}^{-1} \mid \ell_{\mathsf{l}i} = 1)\frac{-\gamma}{N(N+1)} + \mathbb{E}(\nu_{\mathsf{l}i}^{-1} \mid \ell_{\mathsf{l}i} = 0)\left(1 - \frac{\gamma}{N}\right) \\
&- \mathbb{E}(\nu_{\mathsf{l}i}^{-1} \mid \ell_{\mathsf{l}i} = 0)\left(1 - \frac{\gamma}{N+1}\right)
\end{aligned}$$

Keeping the remote inverse transfer rate fixed, to focus on the impact in the curvature of average predictions from the mixture distribution alone, we can observe that this simplifies to

$$\frac{\gamma}{N(N+1)}\left(\mathbb{E}(\nu_{\mathsf{l}i}^{-1} \mid -, \ell_{\mathsf{l}i}=0) - \mathbb{E}(\nu_{\mathsf{l}i}^{-1} \mid -, \ell_{\mathsf{l}i}=1)\right) \tag{3.5}$$

This effect is of course compounded by the linear association between remote read time and cluster size, but this illustrates how the mixture distribution utilized in the input time model allows expected inverse transfer rates to converge quickly to the remote rate at small cluster sizes.

Local Input Time Posterior Emulator

Once a collection of reads observed during benchmarking are determined to be local, whether by explicit collection, or through application of distributional assumptions about the frequency and pattern of their occurrence, we can define the (conditional) local input time posterior emulator based on these benchmarking data. While this distribution is conditional in the sense that it depends on the subset of reads labeled as local, when the read locality indicators are explicitly collected, this can be thought of as being unconditional, since data is no longer random once observed. In this situation, the emulator specified herein can be referred to as simply the 'local input posterior time emulator'. In either case, this emulator can be used to generate local read times which closely mimic those observed by actually running matrix multiplication jobs, and can therefore be used in simulation studies or the derivation of (approximations to) the exact distributional consequences at the task and job levels.

Let $\mathcal{D}_{\mathsf{l}_1}$ denote the vector of observed local inverse transfer rates and suppose that we wish to emulate inverse transfer rates, $\widetilde{\mathcal{D}}_{\mathsf{l}_1}$, with corresponding design points given by $\widetilde{X}_{\mathsf{l}_1} = \begin{bmatrix} \widetilde{\mathbf{1}}_{\mathsf{l}_1} & \widetilde{\ell}_{\mathsf{B}_1} \end{bmatrix}$. Due to the limiting conjugacy of our prior specifications, we can integrate out the model parameters to obtain the posterior emulator conditional only on the observed data. Specifically, we integrate the product of the density given in equation (3.2) with $\mathsf{p}(\beta_{\mathsf{I}_1} \mid \sigma^2_{\mathsf{I}_1}, \mathcal{D}_{\mathsf{I}_1})\mathsf{p}(\sigma^2_{\mathsf{I}_1} \mid \mathcal{D}_{\mathsf{I}_1})$ with respect to $\sigma^2_{\mathsf{I}_1}, \beta_{\mathsf{I}_1}$ and with $\widetilde{\mathcal{D}}_{\mathsf{I}_1}$ substituted for $\mathcal{D}_{\mathsf{I}_1}$ in the likelihood. This yields the following relation between $\widetilde{\mathcal{D}}_{\mathsf{I}_1}$ and $\mathcal{D}_{\mathsf{I}_1}$.

$$\widetilde{\mathcal{D}}_{\mathsf{I}_1} \mid \mathcal{D}_{\mathsf{I}_1} \sim \mathsf{T}_{n_{\mathsf{I}_1}-2}\left(\widetilde{X}_{\mathsf{I}_1}\hat{\beta}_{\mathsf{I}_1}, \hat{\sigma}^2_{\mathsf{I}_1}\left(I + \widetilde{X}_{\mathsf{I}_1}\begin{bmatrix} n_{\mathsf{I}_1} & \sum \ell_{\mathsf{B}_1} \\ \sum \ell_{\mathsf{B}_1} & \sum \ell_{\mathsf{B}_1} \end{bmatrix}^{-1} \widetilde{X}^T_{\mathsf{I}_1}\right)\right) \tag{3.6}$$

Where $n_{\mathsf{I}_1} = \sum(\ell_{\mathsf{I}i})$, $\mathbf{1}_{\mathsf{I}_1}$ denotes an $n_{\mathsf{I}_1} \times 1$ vector of ones, ℓ_{B_1} is shorthand for the vector of indicators the event that a given local read was streamed, and the design matrix for the clusters we wish to estimate inverse rates for is given by $\widetilde{X}_{\mathsf{I}_1}$.

Observe that samples from the posterior emulator are dependent upon one another, especially when n_{I_1} is small. This dependence is induced by the fact that there is remaining uncertainty about the underlying model parameters for the data generation mechanism. As we obtain more data, i.e. as we reduce our uncertainty about the parameters, samples of $\widetilde{\mathcal{D}}_{\mathsf{I}_1}$ become less dependent upon each other, and in the limit as n_{I_1} becomes large, $\widetilde{\mathcal{D}}_{\mathsf{I}_1}$ are independent and normally distributed with parameters equal to the observed benchmarking data mean and variance.

Remote Input Time Posterior Emulator

Similar with the local read case, it is convenient to let $\mathcal{D}_{\mathsf{I}_0}$ denote a vector of the observed of remote reads.

As with before, the additional model parameters can be integrated out of the sampler to obtain the posterior emulator conditional only on the observed data. (3.3) with $\mathsf{p}(\beta_{\mathsf{I}_0} \mid \sigma^2_{\mathsf{I}_0}, \mathcal{D}_{\mathsf{I}_0})\mathsf{p}(\sigma^2_{\mathsf{I}_0} \mid \mathcal{D}_{\mathsf{I}_0})$ with respect to $\sigma^2_{\mathsf{I}_0}, \beta_{\mathsf{I}_0}$ and with $\widetilde{\mathcal{D}}_{\mathsf{I}_0}$ substituted for $\mathcal{D}_{\mathsf{I}_0}$ in the likelihood. This yields the following relation between $\widetilde{\mathcal{D}}_{\mathsf{I}_0}$ and $\mathcal{D}_{\mathsf{I}_0}$.

$$\widetilde{\mathcal{D}}_{\mathsf{I}_0} \mid \mathcal{D}_{\mathsf{I}_0} \sim \mathsf{T}_{n_{\mathsf{I}_0}-3}\left(\widetilde{X}_{\mathsf{I}_0}\hat{\beta}_{\mathsf{I}_0}, \hat{\sigma}^2_{\mathsf{I}_0}\left(I + \widetilde{X}_{\mathsf{I}_0}(X^T_{\mathsf{I}_0}X_{\mathsf{I}_0})^{-1}\widetilde{X}^T_{\mathsf{I}_0}\right); n_{\mathsf{I}_0} - 2\right) \tag{3.7}$$

$$X^T_{\mathsf{I}_0}X_{\mathsf{I}_0} = \begin{bmatrix} n_{\mathsf{I}_0} & \sum \ell_{\mathsf{B}_0} & \sum \mathcal{N}_{\mathsf{I}} \\ \sum \ell_{\mathsf{B}_0} & \sum \ell_{\mathsf{B}_0} & \sum \ell^T_{\mathsf{B}_0}\mathcal{N}_{\mathsf{I}} \\ \sum \mathcal{N}_{\mathsf{I}} & \sum \ell^T_{\mathsf{B}_0}\mathcal{N}_{\mathsf{I}} & \sum \mathcal{N}^2_{\mathsf{I}} \end{bmatrix}$$

Where $\hat{\beta}_{\mathsf{I}_0}$ is the ordinary least squares solution for the regression coefficients based on the benchmarking data, $\hat{\sigma}^2_{\mathsf{I}_0}$ is the residual sum of squares divided by the degrees of freedom, $\mathcal{N}_{\mathsf{I}}$ is a vector of cluster sizes during benchmarking, ℓ_{B_0} is shorthand for the vector of indicators the event that a given remote read was streamed, and the design matrix for the clusters we wish to estimate inverse rates for is given by $\widetilde{X}_{\mathsf{I}_0}$.

Observe again that samples from the posterior emulator are dependent upon one another, especially when n_{I_0} is small. Similar with the results for the local read model, as we obtain more benchmarking data, i.e. as we reduce our uncertainty about the parameters, samples of $\widetilde{\mathcal{D}}_{\mathsf{I}_0}$ become less and less informative of each other. This result holds in practice where given a set of benchmarks, we are curious about investigating performances, and theoretically as long as the determinant of the inner matrix grows faster than its entries as n_{I_0} tends to infinity.

3.3.1 Input Time Posterior Emulator

If the vector of read origin locations is collected while benchmarking jobs are run, then the simulations according to equations (3.6) and (3.7) can be combined according to equation (3.1) to obtain the posterior emulator of input time conditional on the observed local and remote inverse transfer rates.

In the case where the read locations are not collected during benchmarking, a Gibbs sampler can be constructed based on the model specifications above to infer the read locations conditional on the observed inverse transfer rates. We can then either condition on particular discovered configuration(s) of the indicators, or sample input times over the Markov chain induced by the Gibbs sampler to obtain posterior samples of input times. It is notable that this is akin to a first order approximation, since there is commonly dependence in the storage locations of nearby tiles. Dependence between storage locations of nearby tiles can cause the mixture weights between local and remote reads to fluctuate across tasks, altering the skew of the read time distributions.

The Gibbs update for the read locations is given by sampling each $\ell_{\mathsf{l}i}$ conditional on the full vector of inverse transfer rates and $\ell_{\mathsf{l}(-i)}$, all remaining read location indicators.

$$\begin{aligned}
&\mathsf{P}(\ell_{\mathsf{l}i} = 1 \mid \nu_{\mathsf{l}}^{-1}, \ell_{\mathsf{l}(-i)}) \\
&\qquad = \frac{\mathsf{P}(\nu_{\mathsf{l}i}^{-1} \mid \nu_{\mathsf{l}(-i)}^{-1}, \ell_{\mathsf{l}(-i)}, \ell_{\mathsf{l}i} = 1)\mathsf{P}(\nu_{\mathsf{l}(-i)}^{-1} \mid \ell_{\mathsf{l}(-i)})\mathsf{P}(\ell_{\mathsf{l}(-i)}, \ell_{\mathsf{l}i} = 1)}{\sum_{j=0}^{1}\left[\mathsf{P}(\nu_{\mathsf{l}i}^{-1} \mid \nu_{\mathsf{l}(-i)}^{-1}, \ell_{\mathsf{l}(-i)}, \ell_{\mathsf{l}i} = j)\mathsf{P}(\nu_{\mathsf{l}(-i)}^{-1} \mid \ell_{\mathsf{l}(-i)})\mathsf{P}(\ell_{\mathsf{l}(-i)}, \ell_{\mathsf{l}i} = j)\right]} \\
&\qquad = \frac{\mathsf{P}(\nu_{\mathsf{l}i}^{-1} \mid \nu_{\mathsf{l}(-i)}^{-1}, \ell_{\mathsf{l}(-i)}, \ell_{\mathsf{l}i} = 1)\mathsf{P}(\ell_{\mathsf{l}i} = 1)}{\sum_{j=0}^{1}\left[\mathsf{P}(\nu_{\mathsf{l}i}^{-1} \mid \nu_{\mathsf{l}(-i)}^{-1}, \ell_{\mathsf{l}(-i)}, \ell_{\mathsf{l}i} = j)\mathsf{P}(\ell_{\mathsf{l}i} = j)\right]}
\end{aligned} \tag{3.8}$$

Due to our limiting conjugate specifications, these probabilities can be computed analytically. The density for the leading terms are given by equations (3.6) and (3.7), while the second terms are given by equation (3.4). For each sample of ℓ_{l} we can generate posterior inverse transfer rates for reads, with the amount of ‘acceptable’ oversampling in practice related to the size and connectedness of our cluster as well as the amount of dependence between the storage locations of nearby tiles.

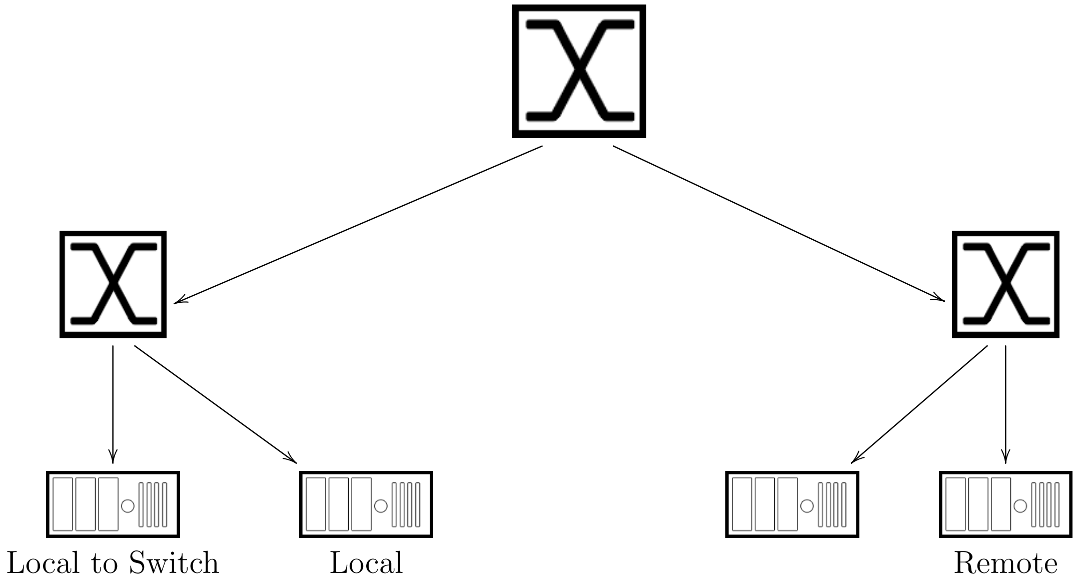
3.4 Task Output Time

Just as there is a surprising amount of complex mathematical systems interacting with each other during the read phase, leading to a very complicated joint distribution based on the first principles of the architectures and algorithms involved, the joint distribution for write times is similarly complicated. We again take a look into the fundamental principles at play, and how we can approximate it's distribution to obtain accurate predictions.

The Hadoop distributed file system replication factor, γ, dictates how many copies of a tile will be stored across the network. A simplistic, yet roughly accurate, view of write times is to consider them as defined by the the maximum of the write times to each of these γ locations. However, the algorithm for writes employed by Cumulon utilizes information from 'fast' writes to accelerate the process, leading standard approaches based on order statistics to have limited utility in this setting.

More specifically, writes in Cumulon are executed through the use of pipelines to increase efficiency. Explicitly incorporating modeling of the write pipeline into the output time model entails modeling the graphical structure of the computational network as well as the algorithm for disseminating data across the network for more optimal access by subsequent tasks or jobs. Furthermore, it is not uncommon to have virtual machines in applications, which can further complicate this modeling effort, as traversing some edges will be much less time consuming than others.

For example, with $\gamma = 3$, the writes may be given by one local write, one write which is local to the same switch, and one write which is remote across the network. Writes over the network furthermore take advantage of efficient data pipelines created by Cumulon, utilizing the relatively fast local and local-to-switch write to aid in the completion of the relatively slow remote write. Given this explanation of the write procedure, it is natural to suppose that the write time will generally be dictated by the remote write time, but while writes slow down for locations which are farther from the existing copies of a tile, the efficiency by which that distance can be traversed increases as near by tiles are completed.



Example of the selected locations for the output of a task in a cluster with $N = 4$, and $\gamma = 3$. It is natural to choose one of the writes to be local to the task which generated the solution tile(s), as local writes are in general fastest, and by choosing the second write to be near by, information from both can be optimally combined into a pipeline to execute the write which is remote across the network.

It is not clear that the network structures observed during benchmarking will, in general, be identical to the network structure for the deployment plan that is ultimately decided upon. One strategy by which computation time can be decreased, is to bid on any available spot-instance machines at a significantly lower cost to attempt to obtain more machine hours for the same cost. However, if the spot price rises above the bid price, the spot-instance machines will be lost, along with any work which hasn't been written back to either a persisting cluster of core machines or persisting file storage system. Given the transient nature of these spot-instance machines, it is especially doubtful when employing this strategy, that the network will be identical to those observed during benchmarking.

Therefore, given the significant increase in complexity, possible negative impacts on the generalizability of predictions from the fitted model, and ultimately increased demand for benchmarking data in order to obtain said predictions, it would seem prudent to develop a model for output time which approximates the observed output times during benchmarking, marginalizing out any specific information about the benchmarking network structures.

As was the case for input volumes, the output volumes can be analytically derived given relevant Cumulon settings. Therefore we proceed by developing a model for the inverse transfer rates for the remote writes.

3.4.1 Output Time Model

Given a collection of observed output times, let n_{O} denote the number of outputs for a task are which respectively have volumes $\{\mathsf{V}_{\mathsf{O}_1}, \ldots, \mathsf{V}_{\mathsf{O}_{n_{\mathsf{O}}}}\}$. As mentioned above, these volumes can be analytically derived from input split size, input sparsity and relevant operator parameter settings.

In practice, the effect of increasing cluster size can vary substantially for small clusters versus large clusters, with write times increasing steeply for each additional machine in small clusters, but only very slightly for large clusters. The steep rise in write times for very small clusters makes sense intuitively. For example, when $N < \gamma$, the addition of an extra machine constitutes an extra write chained onto the end of the pipeline created by Cumulon. That being said, this trend empirically persists for clusters above the $N = \gamma$ threshold, provided that N is not much greater than γ. This could be partially due to the fact that write traffic occupies a significantly larger proportion of available transfer channels of the network when N is only slightly larger than γ, leading to increased wait times as compared with the general setting where $N \gg \gamma$. A plot of inverse transfer times for writes in Cumulon is included below in figure (3.10).

In the case of reads, the probability a block originated from local storage is reasonably well approximated on a first order basis by γ/N, and the incorporation of

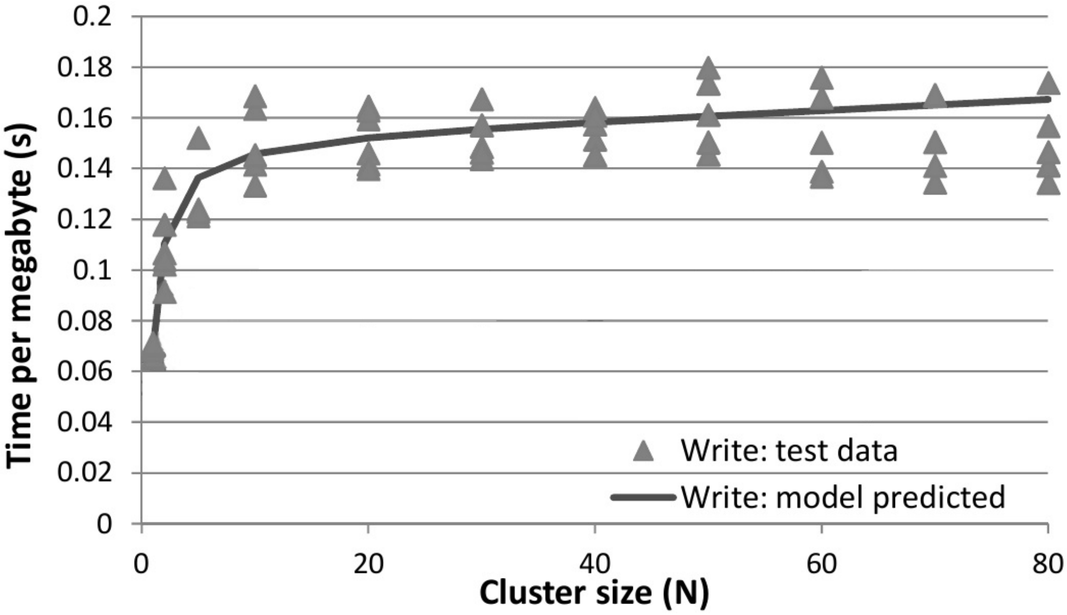


FIGURE 3.8: Graphical summary of the behavior of inverse transfer rates for writes as cluster size is increased in Cumulon from Huang et al. (2013). Observe that responses for cluster sizes $N \geqslant 10$ and $N \leqslant \gamma = 3$ both respectively have strong linear relationships between the effect of increasing cluster size, yet the regression component membership for the $N = 5$ benchmark is slightly ambiguous. Furthermore, note that the deviations from the mean inverse transfer rate shown here at the task level are much closer to normality than the those at the tile level.

the mixture distribution between remote and local read times also served to capture the steep, smooth elbow observed just above the $N = \gamma$ threshold. Writes similarly exhibit the same elbow-like feature in this region, however while the analog of this phenomenon for writes is likely related to the network structure problem discussed above, that is, occasionally a super efficient pipeline can be created for the write chain, and the probability of this event decreases as cluster size increases. Due to differences in network configurations however, such a model would naturally suffer from some inherent misspecification.

Alternatively, given that the effect of cluster size on write times is non-linear in general, another approach is to approximate the regression functions by piecewise linear regression splines. While the exact, optimal position of the knot is not a

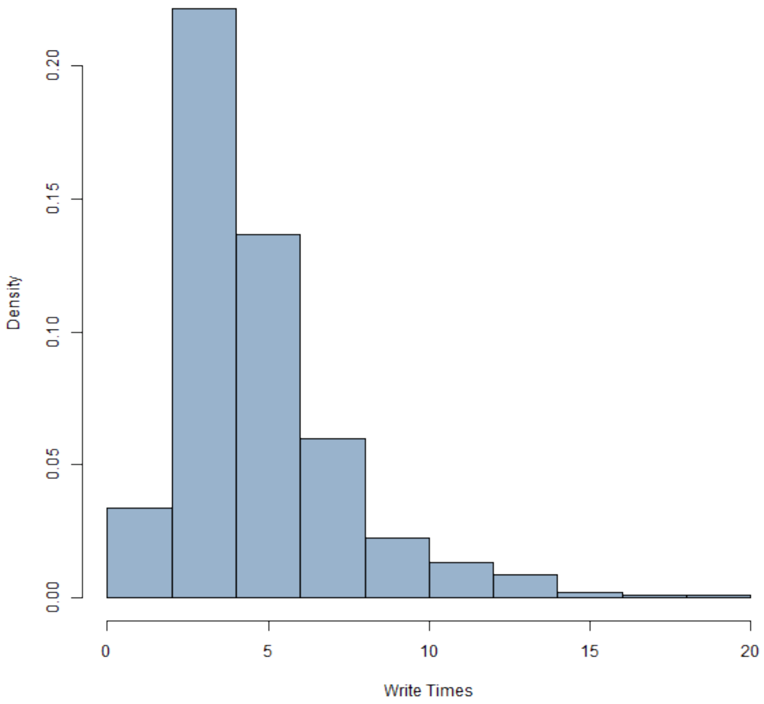


FIGURE 3.9: While the writes originating from a specific task are not uncommonly characterized by having reasonably well separated super-efficient writes, with a few very slow outliers, both the probability and relative efficiency of the 'slow' writes can vary depending on execution location and job ID. This relationship is potentially due to the differences in network structure at both from the perspective of individual nodes, and across jobs as a whole. The result is that it is possible that the general distribution of writes marginalized over execution locations is heavily skewed but does not display clear multimodality, even though many individual tasks do exhibit this trait. This behavior is not entirely surprising, as mixtures of normal distributions are commonly used to to approximate skewed, unimodal distributions, for example the χ^2 distribution which has proven useful in various financial applications. Furthermore, very few components are required in order to construct a high accuracy approximation thereof, meaning the variability across the cloud network need not be extremely complicated to generate this behavior.

priori clear, it is reasonable to presume based on the aforementioned properties of Cumulon's write algorithm, that it lies between $[\gamma + 1, 2\gamma]$. Furthermore, since all benchmarking deployments and ultimately the final deployment plan must at any given time run on a cluster with an integer number of machines, we can restrict the knot to lie in $[\gamma+1, 2\gamma] \bigcap \mathbb{N}$, as any fractional assignment would yield identical training and testing distributions to those corresponding to an integer solution in this interval. As such, the mathematical restriction to integer knot locations, does not practically limit the utility of this approach in any way, and simultaneously allows the posterior to be fully explored.

We suppose that the inverse remote transfer rates $\nu_{\mathsf{O}i}^{-1}$ follow a piecewise linear spline regression, where the location of the knot, k, is a priori uniformly distributed on the bounded set of plausibly optimal integer locations as defined by our knowledge of the Cumulon write procedure. Then

$$\nu_{\mathsf{O}i}^{-1} \mid \beta_{\mathsf{O}}, \sigma_{\mathsf{O}}^2, k \sim [1, \mathcal{N}_{\mathsf{O}_i}]\beta_{\mathsf{O}_0} + \mathbb{1}(\mathcal{N}_{\mathsf{O}_i} > k)\mathcal{N}_{\mathsf{O}_i}\beta_{\mathsf{O}_1} + \mathsf{N}(0, \sigma_{\mathsf{O}}^2) \tag{3.9}$$

$$k \sim \mathsf{Uniform}\{\gamma+1, \ldots, 2\gamma\} \tag{3.10}$$

where $\mathcal{N}_{\mathsf{O}}$ denotes the vector of cluster sizes corresponding to benchmarking data.

With the exception of the case where $N = 1$, there is not significant evidence to suggest that the variance of inverse transfer rates for small clusters systematically vary substantially from those of larger clusters, at least for the range of cluster sizes under consideration here. However, this should be viewed as a practical consideration for our intended applications, as in general, as cluster size increases given a fixed replication factor, γ, the variance of inverse transfer rates for writes should theoretically increase, barring for example, the possibility that the write algorithm at some point imposes a maximum limit on the distance across the network for all 'remote' writes.

It is important to note that due to the underlying pipeline mechanism which can occasionally be radically more efficient (or inefficient) from time to time, that the individual tile writes will likely follow a mixture distribution. However as mentioned, it is difficult to calibrate what the probability of such a highly efficient pipeline might be and to do so in a way that will generalize well to future deployments. As a consequence, the normality assumption inherent in (3.9) will often be more well satisfied when training on the task level writes, as opposed to the tile level writes. This is because it is unlikely that enough of the writes will be radically faster or slower than normal to significantly push the sum away from the general trend. The accuracy of the normal assumption depends on the central limit theorem, with the number of tiles generated in the output of a task being key to the convergence, as well as the degree of non-normality in the underlying tile level write distribution.

Non-informative priors are given by a priori independent $\mathsf{Gamma}(0,0)$ distributions for the residual variance, and flat priors for the regression coefficients. It is convenient to let $\mathcal{D}_{\mathsf{O}}$ denote a vector of the observed of inverse transfer rates for the write operations.

Output Time Posterior Emulator

The posterior emulator based on the likelihood equation (3.9) and prior specifications given by non-informative limiting conjugate choices in conjunction with the a priori independent prior on the knot location from equation (3.10) is represented by a mixture distribution with γ components corresponding to each of the possible knot locations. However, these additional mixture components in practice do not contribute a great deal to the quality of predicted write times, both near the transition between regressions, and at the extremes.

Given this, rather than directly sampling from the posterior predictive distribution to obtain the output time posterior emulator, we work with the conditional posterior emulator, given the most likely knot position. The conditional emulator can then be used to explore the behavior of tasks and jobs across various deployment parameter settings, potentially including combinations not tested during benchmarking. The restriction to the most likely knot simplifies both the procedure of drawing samples from the posterior emulator, as well as derivations involving convolutions of the output time distribution, since each additional mixture component significantly increases the number of terms in higher order convolutions.

The posterior distribution of k given $\mathcal{D}_{\mathsf{O}}$ can be analytically computed due to the limiting conjugate specifications for the regression parameters.

$$
\begin{aligned}
\mathsf{P}(k = k^* \mid \mathcal{D}_{\mathsf{O}}) &= \frac{\mathsf{P}(\mathcal{D}_{\mathsf{O}} \mid k^*)\mathsf{P}(k^*)}{\mathsf{P}(\mathcal{D}_{\mathsf{O}})} \\
&= \frac{\int \mathsf{P}(\mathcal{D}_{\mathsf{O}} \mid \beta_{\mathsf{O}}, \sigma^2_{\mathsf{O}}, k^*)\mathsf{P}(\beta_{\mathsf{O}}, \sigma^2_{\mathsf{O}})\mathsf{P}(k^*) d\beta_{\mathsf{O}} d\sigma^2_{\mathsf{O}}}{\sum_{k=\gamma+1}^{2\gamma} \int \mathsf{P}(\mathcal{D}_{\mathsf{O}} \mid \beta_{\mathsf{O}}, \sigma^2_{\mathsf{O}}, k)\mathsf{P}(\beta_{\mathsf{O}}, \sigma^2_{\mathsf{O}})\mathsf{P}(k) d\beta_{\mathsf{O}} d\sigma^2_{\mathsf{O}}} \\
&= \frac{\frac{1}{\gamma}\int \mathsf{P}(\mathcal{D}_{\mathsf{O}} \mid \beta_{\mathsf{O}}, \sigma^2_{\mathsf{O}}, k^*)\mathsf{P}(\beta_{\mathsf{O}}, \sigma^2_{\mathsf{O}}) d\beta_{\mathsf{O}} d\sigma^2_{\mathsf{O}}}{\frac{1}{\gamma}\sum_{k=\gamma+1}^{2\gamma} \int \mathsf{P}(\mathcal{D}_{\mathsf{O}} \mid \beta_{\mathsf{O}}, \sigma^2_{\mathsf{O}}, k)\mathsf{P}(\beta_{\mathsf{O}}, \sigma^2_{\mathsf{O}}) d\beta_{\mathsf{O}} d\sigma^2_{\mathsf{O}}} \qquad (3.11)
\end{aligned}
$$

Observe that the denominator has only γ terms, so the posterior of any desired quantities can be computed conveniently without the need to resort to Monte Carlo methods, as is typical in model selection problems. Inverse transfer rates for writes are conditionally independent given model parameters, leading to the final simplification in the last line above.

The expression for the density of the collapsed models which appear in equation (3.11) with β_{O} and σ^2_{O} integrated out, that is $\mathsf{P}(\mathcal{D}_{\mathsf{O}} \mid k)$ is given by

$$
\begin{aligned}
\mathsf{P}(\mathcal{D}_{\mathsf{O}} \mid k) &= \int\int \mathsf{P}(\mathcal{D}_{\mathsf{O}} \mid \beta_{\mathsf{O}}, \sigma^2_{\mathsf{O}}, k)\mathsf{P}(\beta_{\mathsf{O}}, \sigma^2_{\mathsf{O}}) d\beta_{\mathsf{O}} d\sigma^2_{\mathsf{O}} \\
&= \int \left(\frac{1}{2\pi}\right)^{\frac{n_{\mathsf{O}}-3}{2}} \left(\frac{1}{\sigma^2_{\mathsf{O}}}\right)^{\frac{n_{\mathsf{O}}-3}{2}} \left|X^{(k)T}_{\mathsf{O}} X^{(k)}_{\mathsf{O}}\right|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}\mathcal{D}^T_{\mathsf{O}}(I - P_{\mathsf{O}_k})\mathcal{D}_{\mathsf{O}}\frac{1}{\sigma^2_{\mathsf{O}}}\right\} d\sigma^2_{\mathsf{O}} \\
&= \Gamma\left(\frac{n_{\mathsf{O}}-3}{2}\right)\left(\frac{1}{\pi}\right)^{\frac{n_{\mathsf{O}}-3}{2}} \left|X^{(k)T}_{\mathsf{O}} X^{(k)}_{\mathsf{O}}\right|^{-\frac{1}{2}} \left(\mathcal{D}^T_{\mathsf{O}}(I - P_{\mathsf{O}_k})\mathcal{D}_{\mathsf{O}}\right)^{-\frac{n_{\mathsf{O}}-3}{2}} \qquad (3.12)
\end{aligned}
$$

where P_{O_k} is the projection onto the column space of the design matrix,

$X_{\mathsf{O}}^{(k)} = \left[1, \mathcal{N}_{\mathsf{O}}, \mathcal{N}_{\mathsf{O}}^{(k)}\right]$, associated with the benchmarking data for the regression defined in equation (3.9) with knot position k. The first integral is completed through recognizing that the kernel of the product of the prior and likelihood in β_{O} in the is Multivariate Normal with parameters $\mathcal{D}_{\mathsf{O}}^T X_{\mathsf{O}}^{(k)} \left(X_{\mathsf{O}}^{(k)T} X_{\mathsf{O}}^{(k)}\right)^{-1}$ and $\sigma_{\mathsf{O}}^2 \left(X_{\mathsf{O}}^{(k)T} X_{\mathsf{O}}^{(k)}\right)^{-1}$, and the second integral is completed through observing that the kernel of the resulting expression is Inverse-Gamma distributed in σ_{O}^2 with parameters $(n_{\mathsf{O}} - 3)/2$ and $\mathcal{D}_{\mathsf{O}}^T (I - P_{\mathsf{O}_k}) \mathcal{D}_{\mathsf{O}}/2$. The calculation above also serves to show that the conditional collapsed posterior emulator follows a T distribution.

Recall that $\mathcal{N}_{\mathsf{O}}$ denotes the vector of cluster sizes corresponding to benchmarking data, and let $\mathcal{N}_{\mathsf{O}}^{(k)}$ denote vector of the element wise products between $\mathcal{N}_{\mathsf{O}}$ and $\mathbb{1}(\mathcal{N}_{\mathsf{O}_i} > k)$. Let $\widetilde{\mathcal{D}}_{\mathsf{O}}$ denote a vector of inverse transfer rates we would like to predict with corresponding to design points $\widetilde{X}_{\mathsf{O}}^{(k)} = [\widetilde{\mathbf{1}}_{\mathsf{O}}, \widetilde{\mathcal{N}}_{\mathsf{O}}, \widetilde{\mathcal{N}}_{\mathsf{O}}^{(k)}]$, as the analog to the observed inverse transfer rates, $\mathcal{D}_{\mathsf{O}}$, with corresponding design points given by $X_{\mathsf{O}}^{(k)} = [\mathbf{1}_{\mathsf{O}}, \mathcal{N}_{\mathsf{O}}, \mathcal{N}_{\mathsf{O}}^{(k)}]$. Then the conditional collapsed posterior emulator for write times is given by

$$\widetilde{\mathcal{D}}_{\mathsf{O}} \mid \mathcal{D}_{\mathsf{O}}, k^* \sim \mathsf{T}_{n_{\mathsf{O}}-3}\left(\widetilde{X}_{\mathsf{O}}^{(k)} \hat{\beta}_{\mathsf{O}}, \hat{\sigma}_{\mathsf{O}}^2 \widetilde{X}_{\mathsf{O}}^{(k)} \left(X_{\mathsf{O}}^{(k)T} X_{\mathsf{O}}^{(k)}\right)^{-1} \widetilde{X}_{\mathsf{O}}^{(k)T}\right) \tag{3.13}$$

$$k^* = \arg\max_k \mathsf{P}(k \mid \mathcal{D}_{\mathsf{O}}) \tag{3.14}$$

Where $\hat{\beta}_{\mathsf{O}_0}$ is the ordinary least squares solution for the regression coefficients based on the benchmarking data with the knot fixed at level k^*, and $\hat{\sigma}_{\mathsf{O}}^2$ is the residual sum of squares divided by the degrees of freedom. As with before, draws from this distribution can be well approximated by using independent normals with the parameters, with the approximation getting sharper as more benchmarking data is collected.

3.5 Computation Time

It is possible to analytically derive the number of floating point operations (FLOPs) for each task, and such an approach is indeed commonplace in addressing the computational complexity of algorithms. However, FLOP counts are biased by differences in both software and hardware, leading their generalizability to suffer significantly. For example, implementations from the JBLAS library will likely significantly outperform user created code with the same analytic number of FLOPs in terms of computation time (Huang et al., 2013). Given this we proceed forward using other informative quantities of computational load to build a predictive model for each task's time spent in the computation phase.

In general, computation tends to be more efficient for split factors which result in more balanced submatrices, however unlike the read and write efficiencies for which we were able to specify parsimonious parametric models based on volume insensitive quantities, it is in general difficult to identify any parsimonious parametric relationship between changes in split factors or equivalently submatrix dimensions for a given job and the rate at which computations are completed. Furthermore, unlike the inverse transfer rates for I/O, it is not clear that the inverse rate at which computations are completed is similarly volume insensitive. However for a given job, due to the fixed split-factor method for disseminating submatrices to tasks, computation volumes and computation efficiencies are inextricably linked, leading the volume scaling issue to be of secondary concern in generalizing predictions based on benchmarking data to future jobs with possibly input matrices of differing size or structure.

Given this, we proceed forward with non-parametric estimation of the computational efficiency directly through the use of Gaussian Process regression in the flavor of Conti and OHagan (2010). The choice to work with computational efficiency rather than computation time, is partially influenced by the potential for dissimilarity in the error distribution of varying amounts of convolved computations. Put simply, the variance of computation time should increase with volume, which could potentially be problematic for the more tractable, non-heteroscedastic Gaussian Processes which may not sufficiently capture this effect. Consider computation times to be generated by an unknown function $C(\cdot)$, and further suppose that

$$C(\cdot) \mid B, \Sigma, r \sim \mathsf{GP}(m(\cdot), a(\cdot, \cdot)\Sigma) \tag{3.15}$$

where $C(\cdot)$ is at least theoretically a deterministic simulator.

Though the common perception of program executions would indicate that computational efficiency is not deterministic in nature, this could simply be due to the high degree of complexity in the mathematical system. A simple example of this disconnect between a deterministic physical phenomenon which appears as random due to a practical lack of information about the underlying conditions which inform the physics of it's mathematical system can be seen in a simple coin toss Diaconis et al. (2007). It is possible to absorb this apparent randomness induced by incomplete information into the likelihood for a Gaussian process emulator of a complex, deterministic system. This can be achieved through attempting to regress on only a subset of the relevant information for determining the state of the simulator, provided that the residuals from the restricted regression do not exhibit strong spatial correlations with the included regression predictors (Conti and OHagan, 2010).

In practice, for any reasonably compact set of predictors, computational efficiency can reasonably be considered to result from a random simulator, and although it stated that $C(\cdot)$ theoretically represents a deterministic simulator, by observing that

$$\begin{aligned} \mathbb{C}\text{ov}(C(\theta), C(\theta) \mid B, \Sigma, r) &= c(\theta, \theta)\Sigma \\ &= 1\Sigma = \Sigma \end{aligned} \tag{3.16}$$

represents the variance between the computational efficiencies at any fixed regression point θ. Where the regression is conditional on machine type, $\omega \in \Omega$, the number of slots per machine, S, and the dimensions of submatrices. For convenience, denote the collection of relevant Cumulon settings for a deployment as $\theta \in \Theta$. Currently, the model learning is executed separately for each machine type ω, in the collection of machines available Ω, we therefore suppress ω from the expressions in conditional distributions throughout, unless specifically relevant.

It is theoretically possible to consider regressing on split factors instead of submatrix dimensions, however this is potentially unattractive depending on the range of split-factors explored during benchmarking. The product of the split factors is equivalent to the number of tasks in a job, and is thus naturally useful for job completion times, however this also implies that the dimensions of submatrices are highly non-linear in the split factors. Furthermore, since computation is local, the parametrization which is most stable in changes in the computational load for each task under perturbations in different regions of the space of possible deployment plans will be more naturally in line with the fixed correlation function kernel $a(\cdot, \cdot)$ employed by the specified Gaussian process in equation (3.15). The total amount of work done by a task is still nonlinear in the submatrix dimensions, since submatrix areas will increase more rapidly when the aspect ratios are nearing symmetry. Augmenting with these is akin to direct assessment based on FLOPs, which as discussed may actually not be helpful for prediction. Our advice is to augment only if the error distribution suggests that insufficient information is being specified for the space of the mean function of the Gaussian process.

Additionally, while in practice S is practically quite limited exempli gratia by the working memory limits of the system et cetera, in clusters consisting of very powerful machines, it may make sense to use a transform S limiting towards a constant in some situations. This reflects the idea that splitting the system in half can potentially significantly alter computational efficiency as opposed to letting a single task occupy the machine at a time, yet increasing S by 1 when it is already large represents marginally significantly less loss of resources per task. Care should be exercised when utilizing this approach to not take S too large, as computational efficiency can suffer without sufficient working memory for the algorithm being executed. Working memory is not the only constraint, as matrix multiplication is quite computationally intensive, and assigning too many tasks to the same machine could lead to time spent waiting for core use.

The functions appearing inside the arguments of the likelihood in equation (3.15) and prior on regression parameters are respectively given by

$$m(\theta_1) = B^T h(\theta_1) \tag{3.17}$$

$$a(\theta_1, \theta_2) = \exp\{-(\theta_1 - \theta_2)^T R(\theta_1 - \theta_2)\} \tag{3.18}$$

$$\mathsf{p}(B, \Sigma \mid r) \propto |\Sigma|^{-1} \tag{3.19}$$

where we must further specify the vector of regression functions $h(\cdot)$ of length n_h, with reasonable choices being defined by the aforementioned heuristics. This essentially defines the space of the mean function for the Gaussian process regression. This prior choice in equation (3.19) is intended to be non-informative, and is generally appropriate for a wide variety of applications.

Let the parameter settings observed during benchmarking be given by $\{\eta_1, \ldots, \eta_{n_\mathsf{C}}\}$ with corresponding observed task computation times be given by $\mathcal{D}_\mathsf{C} = \{\mathcal{D}_{\mathsf{C}_1}, \ldots, \mathcal{D}_{\mathsf{C}_{n_\mathsf{C}}}\}$. Conditional on the model parameters, the observed data follows a matrix normal distribution

$$\mathcal{D}_\mathsf{C} \mid B, \Sigma, r \sim \mathsf{N}_{n_\mathsf{C},1}(HB, A, \Sigma) \tag{3.20}$$

where $H^T = [h(\eta_1) \ldots h(\eta_{n_\mathsf{C}})]$, and $A = [a(\eta_j, \eta_k)]$.

3.5.1 Computation Time Posterior Emulator

As was the case for the input and output models, due to limiting conjugate prior specifications, the parameters B and Σ may be integrated out of the conditional posterior emulator according to their joint posterior distribution to obtain a posterior emulator conditional only on the observed computation times $\mathcal{D}_{\mathsf{C}}$ and r. The conditional posterior emulator is then given by

$$C(\cdot) \mid r, \mathcal{D}_{\mathsf{C}} \sim \mathsf{TP}(m^{**}(\cdot), a^{**}(\cdot, \cdot)\hat{\Sigma}_{\text{GLS}}; n_{\mathsf{C}} - n_h) \tag{3.21}$$

Unfortunately, the completely collapsed posterior emulator, $C(\cdot) \mid \mathcal{D}_{\mathsf{C}}$, incorporating prior uncertainty in r is unavailable. This is a consequence of the fact that integrating equation (3.21) with respect to the posterior distribution, $r \mid \mathcal{D}_{\mathsf{C}}$, yields a complicated kernel for $C(\cdot)$ which is difficult to sample from. It is possible to specify a prior for r and proceed forward via approximate Bayesian computation or Markov chain Monte Carlo methods to obtain the collapsed posterior emulator, or minimally to obtain an optimal conditional posterior emulator which corresponds to the 'best' choice of r as given by its joint distribution with the collected data, for example as was done with the knot location for the output efficiency submodel. In practice, however, sufficiently good choices of r can be discovered by 'manual' search, which result in both well behaved eigenvalues and accurate conditional posterior predictive distributions.

where

$$\begin{aligned}
m^{**}(\theta_1) &= \hat{B}_{\text{GLS}}^T h(\theta_1) + (\mathcal{D}_{\mathsf{C}} - H\hat{B}_{\text{GLS}})A^{-1}t(\theta_1) \\
\hat{B}_{\text{GLS}} &= (H^T A^{-1} H)^{-1} H^T A^{-1} \mathcal{D}_{\mathsf{C}} \\
t^T(\theta_1) &= [a(\theta_1, \eta_1), \ldots, a(\theta_1, \eta_{n_{\mathsf{C}}})] \\
a^{**}(\theta_1, \theta_2) &= a^*(\theta_1, \theta_2) + [h(\theta_1) - H^T A{-}1t(\theta_1)]^T (H^T A^{-1} H)^{-1} [h(\theta_2) - H^T A^{-1} t(\theta_2)] \\
a^*(\theta_1, \theta_2) &= a(\theta_1, \theta_2) - t^T(\theta_1) A^{-1} t(\theta_2) \\
\hat{\Sigma}_{\text{GLS}} &= (n_{\mathsf{C}} - n_h)^{-1} (\mathcal{D}_{\mathsf{C}} - H\hat{B}_{\text{GLS}})^T A^{-1} (\mathcal{D}_{\mathsf{C}} - H\hat{B}_{\text{GLS}})
\end{aligned}$$

This model builds upon the computation time estimation strategy of Cumulon, since the mean function of the collapsed posterior emulator can be interpreted as predicting interpolating between the different parameter settings used during benchmarking. This mean function can be used in lieu of the emulator when the variance is small, or for extra speed. Despite the improper priors, a relatively small number of training benchmarks are theoretically required to construct a valid collapsed posterior emulator for computation time with only, $n_h + 1$ being required at minimum.

Additional benchmarking information serves to inform the shape of the regression function in different parts of the space and reduct uncertainty towards $\hat{\Sigma}_{\text{GLS}}$ in regions of the space of deployment parameters where we have abundant data. As we move farther away from the benchmarking design points, the $a^{**}(\theta_1, \theta_2)\hat{\Sigma}_{\text{GLS}}$ term provides a measure of our uncertainty in run time. This can additionally be used to more efficiently explore the parameter space by choosing training points which reduce this uncertainty.

3.6 Aggregation to Higher Levels

While it is important to understand the modeling implications on the various subcomponents of task completion, that is I/O et cetera, within the Cumulon execution model in their own rights, this information can also be used to aggregate to the task and job levels. Furthermore, it is possible to explore their implied distributional changes over different deployment parameter sets, to obtain more direct, practical implications on actionable deployment choices for runtime or cost constrained optimization. Modeling the subcomponents of task completion potentially allows statements to be made about the distribution task and job completion times after collecting a relatively small set of benchmarking data, however this comes at the cost of increased complexity of the distributions for aggregations.

3.6.1 Conditional Posterior Task Time Emulation

Unlike tasks in the MapReduce framework, tasks in the Cumulon execution model have direct control of their I/O, additionally there is neither a reduce phase nor shuffling. This greatly simplifies the process of obtaining task level completion times, and they are well approximated by

$$\text{Task} = \text{Read Time} + \text{Write Time} + \text{Computation Time} + \epsilon_{\mathsf{T}} \tag{3.22}$$

Initialization time does not have a large theoretical impact for the tasks of multiplication intensive jobs which are brought to the cloud in practice, and empirically its effect appears to be on the order of a few seconds. For reasonably sized tasks, this initialization time is a small fraction of the total task time; additionally, it is generally possible to correct for this bias, by recording total task completion time along with the tile level I/O time and task level computation time. Initialization times do not appear to vary significantly on the scale of task completion times for a specific deployment parameter setting, or over multiple deployment parameter settings. Therefore, while it is possible to formally specify distributions for modeling initialization times as we have done for I/O and computation, it is sufficient in practice to correct this bias by simply adding a constant to the predictions from equation

(3.22), so task completion times are not consistently under-estimated. Let μ_0 denote average initialization time, then set

$$\mu_0 = \bar{\epsilon}_{\mathsf{T}} \tag{3.23}$$

By our modeling choices and the properties of the Cumulon execution model, the models described in sections 3.3.1, 3.4.1 and 3.5.1 for input, output and computation time can be reasonably assumed to be conditionally independent. This implies that samples from the conditional posterior emulator for task time can be obtained by sampling from the posterior emulator for each input tile, equations (3.8 then 3.6, 3.7), and the conditional posterior emulators for output, equation (3.13), and computation time, equation (3.21), and summing their results together as per equation (3.22), optionally adding the bias correction from equation (3.23).

More generally, the conditional posterior task completion time emulator is given by a linear combination of these T distributed random variables for each of the conditional posterior emulators for computation and output efficiencies, and the mixture of T distributions for read efficiency. The coefficients of the linear combination are defined by relevant operator parameter settings, and are given by the inverse transforms from the respective modeled efficiencies to total time spent in each phase. Unfortunately, linear combinations of T distributed random variables do not always have a recognizable form.

For simplicity let $df_{\mathsf{I}_1}, df_{\mathsf{I}_0}, df_{\mathsf{O}}, df_{\mathsf{C}}$ represent the degrees of freedom of the four types of T distributions under consideration and denote the locations and scales after transformation from efficiency to time by μ, Σ with the appropriate, matching subscript. Utilizing the scale mixture of Normals representation for the T distribution, it is easy to see that these individually remain T distributed with the same degrees of freedom after transformation:

$$\begin{aligned}
\mathsf{P}(\delta \mathsf{T}_{df}(\mu,\Sigma) \leqslant k) &= \mathsf{P}\left(\delta\left(\frac{\mathsf{N}(0,\Sigma)}{\sqrt{\mathsf{Gamma}\left(\frac{df}{2},\frac{df}{2}\right)}} + \mu\right) \leqslant k\right) \\
&= \mathsf{P}\left(\frac{\delta\sqrt{\Sigma}\mathsf{N}(0,1)}{\sqrt{\mathsf{Gamma}\left(\frac{df}{2},\frac{df}{2}\right)}} + \delta\mu \leqslant k\right) \\
&= \mathsf{P}\left(\frac{\mathsf{N}(0,\delta^2\Sigma)}{\sqrt{\mathsf{Gamma}\left(\frac{df}{2},\frac{df}{2}\right)}} + \delta\mu \leqslant k\right) \\
&= \mathsf{P}(\mathsf{T}_{df}(\delta\mu,\delta^2\Sigma) \leqslant k) \qquad (3.24)
\end{aligned}$$

Furthermore, it is not necessary to keep careful track of the mean parameters of the T distributions in the convolution. This is because the distribution of the convolution of the transformed distributions is simply a translation of the transformed, re-centered distributions. It is again easy to see this utilizing the scale mixture of Normals representation of the T distribution.

$$\begin{aligned}
\mathsf{P}(\mathsf{T}_{df_1}(\mu_1,\Sigma_1) + \mathsf{T}_{df_2}(\mu_2,\Sigma_2) \leqslant k) &= \mathsf{P}\left(\frac{\mathsf{N}(0,\Sigma_1)}{\sqrt{\tau_1}} + \mu_1 + \frac{\mathsf{N}(0,\Sigma_2)}{\sqrt{\tau_2}} + \mu_2 \leqslant k\right) \\
&= \mathsf{P}\left(\frac{\mathsf{N}(0,\Sigma_1)}{\sqrt{\tau_1}} + \frac{\mathsf{N}(0,\Sigma_2)}{\sqrt{\tau_2}} \leqslant k - \mu_1 - \mu_2\right) \\
&= \mathsf{P}\left(\mathsf{T}_{df_1}(0,\Sigma_1) + \mathsf{T}_{df_2}(0,\Sigma_2) \leqslant k - \mu_1 - \mu_2\right)
\end{aligned}$$

Where

$$\tau_i \sim \mathsf{Gamma}\left(\frac{df_i}{2},\frac{df_i}{2}\right)$$

If $df = 1$ in all cases, then each distribution is Cauchy with locations given by $\mu_{\mathsf{I}_1}, \mu_{\mathsf{I}_0}, \mu_{\mathsf{O}}, \mu_{\mathsf{C}}$ and scales given by $\Sigma_{\mathsf{I}_1}, \Sigma_{\mathsf{I}_0}, \Sigma_{\mathsf{O}}, \Sigma_{\mathsf{C}}$. Utilizing the fact that the characteristic function of a convolution of independent random variables is given by the product of the characteristic functions of the constituent random variables, it is easy to see that the sum is a mixture of Cauchy distributions each of the form $j\mu_{\mathsf{I}_1} + (J-j)\mu_{\mathsf{I}_0} + \mu_{\mathsf{O}} + \mu_{\mathsf{C}}$ and scale $j\Sigma_{\mathsf{I}_1} + (J-j)\Sigma_{\mathsf{I}_0} + \Sigma_{\mathsf{O}} + \Sigma_{\mathsf{C}}$, where J here denotes the number of tiles each task must read, i.e. $J \approx \frac{l}{B^2 f_l}\left(\frac{m}{f_m} + \frac{n}{f_n}\right)$.

The potentially large number of mixture elements is generally not a cause for concern in practice, due to the implications of equation (3.4). This implies that the mixture weights for each of these distributions are then given by the binomial probability of a specified number of local reads, j, relative to the task under consideration. Consequentially, a high accuracy reconstruction can be obtained by only using a few mixture elements for many clusters, as both the binomial coefficients, and probability products decay rapidly when far from the expected number of local reads.

If $\mathit{df} = \infty$ in all cases, then each distribution is normal, where the means and variances match the location and scale parameters respectively from the Cauchy derivation above. Furthermore, just as the Cauchy is a member of the family of stable distributions, linear combinations of Normals remain normally distributed. Therefore, the distribution of the sum of the subcomponents of task time is given by a mixture of normal distributions with mixture weights and parameters matching those obtained for the $\mathit{df} = 1$ case.

Derivation of the exact distribution of a sum of T distributed random variables for intermediate cases is notoriously complicated, and has been historically studied within the context of the Behrens-Fisher problem (Fisher and Healy, 1956). Due to the stochastic ordering of standard T distributed random variables in their degrees of freedom, these intermediate cases can be loosely interpreted as interpolations between the aforementioned cases based on stable distributions.

Formulae based hypergeometric functions, which reduce to closed forms for odd degrees of freedom, on for the distribution of the convolution of two T densities for special cases were given by Ghosh. In general, with the convolution of potentially many T distributions, providing all of their degrees of freedom are odd, it is possible to express the distribution of the convolution as a sum of 'pre-T' variables. Where a pre-T random variable is equal in distribution to a unit scale T distribution divided by the square root of its degrees of freedom. Through the use of Bessel functions, it is possible to obtain a closed form representation of the characteristic function of the pre-T random variable whenever the degrees of freedom are odd, and ultimately leads to an expression for the cumulative distribution function of the linear combination of T distributions by a finite polynomial in the cumulative distribution functions for the pre-T random variables (Walker and Saw, 1978; Walker, 1977).

The characteristic function for each of the pre-T random variables is given by

$$\phi(t, df) = e^{-|t|} q_{\lfloor \frac{df}{2} \rfloor}(|t|) \tag{3.25}$$

$$q_{\lfloor \frac{df}{2} \rfloor}(|t|) = \frac{\lfloor \frac{df}{2} \rfloor !}{(2\lfloor \frac{df}{2} \rfloor)!} \sum_{k=0}^{\lfloor \frac{df}{2} \rfloor} \frac{(2\lfloor \frac{df}{2} \rfloor - k)!}{k!(\lfloor \frac{df}{2} \rfloor - k)!} (2|t|)^k \tag{3.26}$$

Define T* to be the desired convolution, and define T** to be the convolution replacing each T density with it's pre-T counterpart, and without loss of generality, renormalize so that the coefficients on the pre-T random variables sum to 1. Since the pre-T variables are independent, the characteristic function of T** is given by the product of the set of pre-T characteristic functions, each of which can be calculated from equations (3.25) and (3.26). Let λ denote the vector of leading coefficients on powers of $|t|$ contained within the finite polynomial expression in $|t|$ from the product of the characteristic functions. Next define Q^{-1} to be a lower triangular matrix with the entry in position i, j defined by

$$Q^{-1}_{i,j} = \frac{(-1)^{i-j}(2j)!(i+1)!}{2^i(i-j)!j!(2j-i+1)!} \tag{3.27}$$

Then the cumulative distribution function of T^{**} evaluated at a given threshold is given by a linear combination of the cumulative distribution functions of the pre-T distributions each evaluated at the same threshold. Since the pre-T random variables are simply rescaled T random variables, it is sufficient for calculation to have access to a function or table which contains the percentiles of the T distribution. The coefficients of the linear combination of pre-T random variables is given by

$$\eta = \lambda Q^{-1} \tag{3.28}$$

$$\sum \eta = 1 \tag{3.29}$$

These coefficients should sum to one and while the entries of Q^{-1} may not be well behaved numerically for the degrees of freedom in the range of the data, this is balanced by the behavior of λ. Rather than calculating each individually, it can be beneficial to expand equation (3.28) with equations (3.27) and the coefficients on powers of $|t|$ from the product of all characteristic functions as given by equation (3.26) before calculations are explicitly performed.

Due to the location-scaling result for T distributions from equation (3.24), it is always possible to rewrite a linear combination of T distributions with general scale parameters, into a linear combination of unit scale T distributions by absorbing the scales into the linear combination coefficients. Given this, to calculate the cumulative distribution function of T^{*} at a desired threshold, it is sufficient to calculate T^{**} at the threshold divided by the sum of the adjusted linear combination coefficients.

Since the cumulative distribution functions are all evaluated at the same threshold, each side of the equation can be differentiated, to show that with odd degrees of freedom, that each convolution element of the mixture distribution for the conditional posterior task completion time emulator, is in turn a mixture distribution over T distributions with odd degrees of freedom.

In the case of even degrees of freedom, it is possible to provide upper and lower bounds for this result, based on the heaviness of the tails of the offending pre-T distributions with even degrees of freedom. This is achieved by an absolute stochastic ordering, $\overset{D}{>}$, which states that for any two random variables X and Y, $X \overset{D}{>} Y$ if and only if $\mathsf{P}(|X| > k) \geqslant \mathsf{P}(|Y| > k)\ \forall k$, and $\exists k$ such that $\mathsf{P}(|X| > k) > \mathsf{P}(|Y| > k)$. The midpoint of the tail probability approximation for the cumulative distribution of the convolution of the desired T densities at a desired cutoff can be used as a proxy for the exact result.

It is important to observe that not all T distributions will have the same degree of freedom in practice. For example, there are typically many tiles of data handled by a given task, leading n_{I} to be much larger than n_{C}. As discussed above, despite the complexity of this problem, it is possible to obtain relatively tight bounds for the distribution of the convolution in practice especially if the degrees of freedom for all of the distributions are not especially small. Furthermore, information is shared across each of reads, writes, and computation. This implies that even relatively short explorations of each of many deployment parameter settings will likely generate enough data points to render the normal approximation sufficient for most purposes, additionally leading to weak dependence between samples from the (conditional) posterior emulators. In this situation, an approximation for the combined posterior task completion time emulator can be obtained by simply taking a linear combination of the emulators from each section above. This linear combination is defined by relevant operator parameter settings, and is given by the inverse transforms from the respective modeled efficiencies to total time spent in each phase of task execution, and avoids the potential difficulties in calculating the mixture coefficients for the pre-T distributions.

3.6.2 Job Completion Times

In this section, we address the problem of aggregating from the conditional posterior task emulator to estimators of job completion time and performance. We are primarily concerned here with the issue of predicting completion time, since cost is a linear function of time and the quality and quantity of machines rented.

The completion time of a job is indicated by the time the last task finished writing it's output back to storage. Recall that N denotes the number of machines in the cluster, and S denotes the number of logical threads per machine. Tasks complete concurrently in each of the $N \cdot S$ logical threads, and are conceptually divided into waves of execution which correspond to their position within the sequential execution of each logical thread. When ever a task is finished, the scheduler steps in to assign responsibility to a new portion of the output to that execution location. Randomness in the completion time of tasks causes waves to intermix over time, smoothing out increases in job times for increases in workload. The specific behavior of the scheduler is not explicitly modeled here, and while that may introduce a small negative bias in the job time predictions, the effect is not empirically noticeable on the scale of job completion times.

Given this it is natural to consider that job times are determined by order statistics of convolutions of task time based on the number of tasks executed in the thread to finish last. However, if the thread which will finish last is conditioned on, it will likely change the distribution of it's order statistics. Furthermore, it is in general difficult to even know how many tasks will have executed in the thread which finishes last. Below we will consider different conditions on the distribution of task completion times, and their implications for reasonable strategies for obtaining estimates of job completion times.

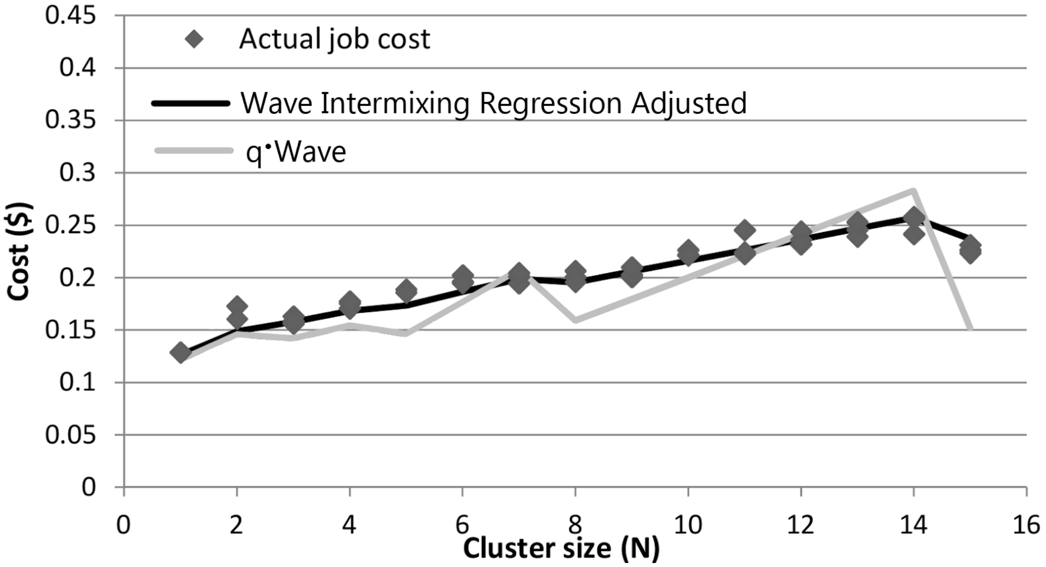


FIGURE 3.10: Graphical summary of illustrating the smooth increase of job cost over intermixed waves. Increasing cluster size decreases the estimate of the number of waves required to complete the job, q as defined by equation (3.33), and hence leads to the jagged line observed for the method based on the raw number of waves. Given the mean time it takes a task to complete, the job cost equation was recalibrated through adjusting the number of waves in a second stage regression procedure in Huang et al. (2013). This was done in the slow mixing case, which implies that the second stage regression procedures only needed to make small adjustments to the raw wave based calculation, as is evidenced by the closeness of the above curves (Huang et al., 2013).

Slow Wave Mixing

When the variance of task completion times are relatively small compared to their completion times, jobs execute in a series of mostly distinct waves which slowly mix together over time, as is the case for dense matrix multiplication. The speed of the mixing is asymptotically related to the coefficient of variation of the components of the conditional posterior task completion time emulation distribution, where coefficients near zero imply that waves will be initially mostly distinct and mix slowly over time, while large coefficients of variation will indicate large amounts of uncertainty in wave location.

Since the distribution for task completion times can be incredibly complicated as detailed in section 3.6.1, we simplify the notation somewhat for this section, so as to not obfuscate the theoretical importance of these derivations behind complicated mixture equations. Suppose that the density of the conditional posterior task emulator is given by

$$\mathsf{p}(t \mid \theta) = \sum_{j=1}^{J|\theta} \pi_j \mathsf{p}_j(t \mid \theta) \tag{3.30}$$

$$\mu|_\theta = \sum_{j=1}^{J|\theta} \pi_j \mu_j|_\theta \tag{3.31}$$

where μ_j represents the mean of each of a random variable with density p_j. If samples sizes are not sufficiently large, each p_j may in turn be specified by a finite mixture distribution with weights specified by η_j defined in equation (3.28), in accordance with the results of section 3.6.1.

$$\mathsf{p}_j(t \mid \theta) = \sum \eta_{j,i} \mathsf{p}_{j,i}(t \mid \theta) \tag{3.32}$$

Recall that each $\mathsf{p}_{j,i}$ distribution is given by a pre-T distribution with odd degrees of freedom, with even distributions of even degree being bounded above and below by subtracting and adding one to the degree of freedom respectively (Walker, 1977). An estimate of the number of waves to be traversed is given by

$$q_1 = \left\lceil \frac{f_m \cdot f_l \cdot f_n}{NS} \right\rceil \tag{3.33}$$

With the capacity of the final wave approximated by

$$q_2 = f_m \cdot f_l \cdot f_n \mod NS \tag{3.34}$$

With the exception that the last wave will potentially be full when $f_m \cdot f_l \cdot f_n \mod NS = 0$ (Huang et al., 2013). In order to be relatively sure that the job completion time will be specified by the maximum of q_2 distributions, each of which are a convolution of $p(\cdot)$ of order q_1, we further need to estimate the variance in the distribution of said convolution. By design, the posterior emulators for each of the subcomponents of task time are assumed conditionally independent by our modeling strategies. Therefore, define σ^2_{I}, σ^2_{O}, and σ^2_{C} to be the variances of the (conditional) posterior emulations of I/O and computation time respectively, then define

$$\sigma^2 = \sigma^2_{\mathsf{I}} + \sigma^2_{\mathsf{O}} + \sigma^2_{\mathsf{C}} \tag{3.35}$$

Without knowledge of the underlying parameters, successive samples from the posterior emulators are weakly dependent on one another, but as discussed above this dependence decays quickly with as more benchmarking data are collected. Therefore, we can approximate the variance of the convolution of the completion of q_1 task completion times by summing the variance in input times, output times, and computation time for each task and multiplying the result by q_1. Then, in cases where

$$\frac{\sqrt{q_1 \sigma^2}}{K(q_2)} < \mu \tag{3.36}$$

we presume that the above estimates for wave completion and population of the final wave are with high probability correct assertions. As the amount of benchmarking data increases the distribution of each of the posterior emulators converges upon normality, causing K in the above calculation to tend towards the Z-score corresponding to a specified probability cutoff that the assertions from equations (3.33) and (3.34) are indeed correct. In practice it is sufficient to inflate the Z-score slightly to guarantee that at least as much probability as desired is obtained.

Given the above conditions, moments of the distribution of job completion, and in particular expected job completion time, are available through numerical integration.

Let T_i be a random variable which is the sum of q_1 random variables with density $\mathsf{p}(\cdot \mid \theta)\ \forall i$. Then, the cumulative distribution function of T_1 can be obtained via the iterative application of the methods in section 3.6.1, and

$$\begin{aligned}
\mathsf{P}(\max(T_1, \ldots, T_{q_2}) \leqslant t) &= \mathsf{P}(T_1 \leqslant t) \cdot \ldots \cdot \mathsf{P}(T_{q_2}) \leqslant t) \\
&= \mathsf{P}(T_1 \leqslant t)^{q_2} \\
\frac{\partial}{\partial t}\mathsf{P}(\max(T_1, \ldots, T_{q_2}) \leqslant t) &= \frac{\partial}{\partial t}\mathsf{P}(T_1 \leqslant t)^{q_2} \\
\mathsf{p}(\max(T_1, \ldots, T_{q_2}))(t) &= q_2 \mathsf{P}(T_1 \leqslant t)^{q_2 - 1}\mathsf{p}(T_1)(t) \qquad (3.37)
\end{aligned}$$

Then expected job completion time, J^*, is given by the integral

$$J^* \mid \theta = \int_{-\infty}^{\infty} t \cdot q_2 \mathsf{P}(T_1 \leqslant t)^{q_2 - 1}\mathsf{p}(T_1)(t)dt \qquad (3.38)$$

In practice it is sufficient to truncate most of these distributions, as the tails of the binomial distribution for the origin of local reads decay quite rapidly. Furthermore, since the premise underlying this approach is that there is low variance in task time, the limits on the integral can be reasonably shrunk to within a few multiples of μ without altering the solution.

Then the optimal job deployment parameter setting with respect to minimizing expected job completion time is given by

$$\theta^* = \arg\min_{\theta} \left(J^* \mid \theta\right) \qquad (3.39)$$

In the situation where the normal approximations are used, and given the moment approximations defined above, the expression for expected job completion time J^* for the mode mixture component can be simplified to the form

$$q_1\mu + \sqrt{q_1}\sigma \int_{-\infty}^{\infty} t \cdot q_2 \Phi(t)^{q_2 - 1}\phi(t)dt \qquad (3.40)$$

which corresponds to a probabilistically motivated version of the wave intermixing regression adjusted approach employed by Huang et al. (2013).

3.6.3 Fast Wave Mixing

In the situation where equation (3.36) fails to hold, obtaining the explicit distribution of job completion time is incredibly complicated. Even in the case where the variance of task completion time is sufficiently small relative to the length of the job and the mean task completion time, several steps of approximations are used in practice to obtain the desired quantities for expected job completion times.

However given a one time calculation of the parameters of the relevant distributions, task times can be generated utilizing only normal and gamma random number generators. Therefore, we can directly use samples from the posterior emulators to generate samples of job completion times, by generating and scaling a small handful of T distributed variables.

This allows us to partially explore multiple deployment plans, borrowing relevant information across each of them, to decide which of them to ultimately utilize to finish execution of the complete matrix multiplication job.

3.7 Results & Conclusions

We often take the problem of matrix multiplication for granted, but the complexity of the distributions involved for emulating the various components of task time, and the myriad of conditions for their aggregation to the task and subsequently to the job levels indicate that there is indeed a rich, complex mathematical landscape underlying this seemingly simple problem.

This research has achieved its initial primary goals, through the creation of predictive models for job completion times based on only a subset of tasks, and the reverse validation of its expected job completion time predictions, by establishing the conditions upon which they reduce back to previously obtained expressions. Convergence to accurate predictions tends to be relatively fast in practice, owing partially to the representations of the likelihood based on normal distributions, as well as the theoretical strength of parsimonious parametric models for parts of the problem which are relatively more well understood. The results contained herein further allow for optimization of deployment parameter settings obtainable through borrowing information across benchmarks executed on only a small subset of tasks for each entertained set of deployment parameters, which is a common point of uncertainty for many researchers who would like to bring their algorithms featuring multiplication, whether it be iterative, or simply of infeasible scale to tackle with commonly available consumer level computing instruments.

While there is some tolerance built into these statistical models used herein for higher coefficients of variation, the primary goal of this analysis was to understand the sources of uncertainty in dense matrix multiplications, and to reproduce non-probabilistic estimates of job completion times obtained previously, for example equation (3.40).

Empirical results confirm the utility of comparing deployment parameter settings via equation (3.39), whether through explicit approximation of the job completion time distribution, or if obtained via sampling as in section (3.6.3).

In this situation, several deployment parameters have reasonably similar job time distributions. Therefore, it is possible that the specific index for which deployment parameter setting resulted in optimal run time might change if the experiment was repeated. That being said, the job time corresponding to the minimizer of $J^* \mid \theta$ is 2700, which is about two minutes off the best possible run time observed in this experiment, 2571, with about 5% more time spent to complete the job relative to choosing the $6 \times 6 \times 3$ split the foresight. More telling is that the next best deployment parameter settings according to $J^* \mid \theta$ is $6 \times 6 \times 6$, which misses the fastest observed

Table 3.1: Expected completion time is able to identify both highly efficient and highly inefficient deployment parameter settings for a test workload.

f_m	f_l	f_n	Job_1	J^*
6	6	6	2600	2426
6	6	3	2571	2563
6	6	9	2881	2722
9	6	3	2700	2422
9	6	6	2940	2707
9	6	9	3120	2737
6	9	3	2948	2984
6	9	6	3211	3002
9	9	9	3458	2972
9	9	3	3111	2988
18	3	6	3029	2770
18	3	3	2860	2818
3	18	6	4419*	4267*

job completion time by only 29 seconds, with about 1% more time spent to complete the job relative to the fastest observed split.

In addition to identifying highly efficient deployment parameter settings, we can also utilize $J^* \mid \theta$ to avoid bad deployment parameter settings. Take for example $\max_\theta J^* \mid \theta = 4267$, the observed job time corresponding to this deployment parameter setting is by far the worst of all entertained with a time of 4419, which is equivalent to about 72% more time spent to complete the job relative to the fastest observed split.

If explicit calculation of the distribution of job completion time is avoided, even more complex models for I/O and computation time can be entertained, providing that they can be easily sampled from. This could for example feature skewed error distributions for write efficiencies in accordance with the results summarized in Figure (3.9). The number of terms in the theoretically exact convolutions which appear for both task and job completion times are extremely sensitive to the number of mixture components in each of the basic I/O and computational efficiency submodels, however these additional terms do not significantly impact the difficulty of simulating from the conditional posterior emulators. Such extensions will likely be pivotal in the adaptation of this methodology to the high volatility patterns observed in sparse and sparse-by-dense multiplications which is the next frontier of this work.

Bibliography

Bastos, L. S. and OHagan, A. (2009), “Diagnostics for Gaussian process emulators,” *Technometrics*, 51, 425–438.

Caspi, A. and Silva, P. A. (1995), “Temperamental qualities at age three predict personality traits in young adulthood: Longitudinal evidence from a birth cohort,” *Child development*, 66, 486–498.

Chai, K. M. A. (2012), “Variational multinomial logit gaussian process,” *The Journal of Machine Learning Research*, 13, 1745–1808.

Chu, W. and Keerthi, S. S. (2007), “Support vector ordinal regression,” *Neural computation*, 19, 792–815.

Constantine, P. G. and Gleich, D. F. (2011), “Tall and skinny QR factorizations in MapReduce architectures,” in *Proceedings of the second international workshop on MapReduce and its applications*, pp. 43–50, ACM.

Conti, S. and OHagan, A. (2010), “Bayesian emulation of complex multi-output and dynamic computer models,” *Journal of statistical planning and inference*, 140, 640–651.

Diaconis, P., Holmes, S., and Montgomery, R. (2007), “Dynamical bias in the coin toss,” *SIAM review*, 49, 211–235.

Dikkers, H. J. (2005), “Support vector machines in ordinal classification,” .

Fisher, R. A. and Healy, M. (1956), “New tables of Behrens’ test of significance,” *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 212–216.

Fricker, T. E., Oakley, J. E., and Urban, N. M. (2013), “Multivariate Gaussian process emulators with nonseparable covariance structures,” *Technometrics*, 55, 47–56.

Friedman, J., Hastie, T., and Tibshirani, R. (2010), “Regularization paths for generalized linear models via coordinate descent,” *Journal of statistical software*, 33, 1.

Ghosh, J. and Clyde, M. A. (2011), "Rao–blackwellization for bayesian variable selection and model averaging in linear and binary regression: A novel data augmentation approach," *Journal of the American Statistical Association*, 106.

Huang, B., Babu, S., and Yang, J. (2013), "Cumulon: Optimizing statistical data analysis in the cloud," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pp. 1–12, ACM.

Huang, B., Jarrett, N., Babu, S., Mukherjee, S., and Yang, J. (2014), "Cumulon:Cloud-Based Statistical Analysis from Users Perspective," pp. 77–89.

Jordan, A. (2002), "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," *Advances in neural information processing systems*, 14, 841.

Kennedy, M. C. and O'Hagan, A. (2001), "Bayesian calibration of computer models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63, 425–464.

Krueger, R. F., Caspi, A., Moffitt, T. E., and Silva, P. A. (1998), "The structure and stability of common mental disorders (DSM-III-R): a longitudinal-epidemiological study." *Journal of abnormal psychology*, 107, 216.

Krueger, R. F., Caspi, A., and Moffitt, T. E. (2000), "Epidemiological Personology: The Unifying Role of Personality in Population-Based Research on Problem Behaviors," *Journal of personality*, 68, 967–998.

Lin, H.-T. and Lin, C.-J. (2003), "A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods," *submitted to Neural Computation*, pp. 1–32.

Macskassy, S. and Provost, F. (2004), "Confidence bands for ROC curves: Methods and an empirical study," Proceedings of the First Workshop on ROC Analysis in AI. August 2004.

Mazzucco, M. and Dumas, M. (2011), "Achieving performance and availability guarantees with spot instances," in *High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on*, pp. 296–303, IEEE.

McGue, M., Bacon, S., and Lykken, D. T. (1993), "Personality stability and change in early adulthood: A behavioral genetic analysis." *Developmental psychology*, 29, 96.

Mitchell, T. M. (1997), "Machine learning. 1997," *Burr Ridge, IL: McGraw Hill*, 45.

Murray, H. A. (1938), *Explorations in personality*, Oxford Univ. Press.

Newsom, S. (2006), “Pioneers in infection control: John Snow, Henry Whitehead, the Broad Street pump, and the beginnings of geographical epidemiology,” *Journal of Hospital Infection*, 64, 210–216.

Silva, P. A. and Stanton, W. R. (1996), *From child to adult: The Dunedin multidisciplinary health and development study*, Oxford University Press Auckland.

Tellegen, A. (1982), “Brief manual for the multidimensional personality questionnaire,” *Unpublished manuscript, University of Minnesota, Minneapolis*, pp. 1031–1010.

Tellegen, A., Lykken, D. T., Bouchard, T. J., Wilcox, K. J., Segal, N. L., and Rich, S. (1988), “Personality similarity in twins reared apart and together.” *Journal of personality and social psychology*, 54, 1031.

Tibshirani, R. (1996), “Regression shrinkage and selection via the lasso,” *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288.

Vapnik, V. (2000), *The nature of statistical learning theory*, Springer Science & Business Media.

Walker, G. A. (1977), “On the Distribution of a Linear Combination of T-distributed Variables,” Ph.D. thesis, University of Florida.

Walker, G. A. and Saw, J. G. (1978), “The distribution of linear combinations of t-variables,” *Journal of the American Statistical Association*, 73, 876–878.

Zhang, Q., Zhu, Q., and Boutaba, R. (2011), “Dynamic resource allocation for spot markets in cloud computing environments,” in *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, pp. 178–185, IEEE.

Zhang, Y., Zhang, W., and Yang, J. (2010), “I/O-efficient statistical computing with RIOT,” in *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, pp. 1157–1160, IEEE.

Biography

Nicholas Walton Daniel Jarrett was born in Elmira, New York on May 5, 1988. In August 2006, he enrolled in University at Albany, where he completed his B.S. in Actuarial Science and M.A. in Mathematics by May 2010, and graduated Summa Cum Laude. He then continued his graduate career at Duke University, enrolling in the statistical science Ph.D. program in August 2010. In December 2013, he recieved his M.S. in Statistics and his Ph.D. in May 2015.