Take off a Load: Load-Adjusted Video Quality Prediction and Measurement

Ruairí de Fréin[†] [††]
[†]KTH - Royal Institute of Technology, Stockholm,
Sweden
[††]Waterford Institute of Technology,
Ireland
web: https://robustandscalable.wordpress.com

in: 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, to appear. See also BIBTEX entry below.

BIBTEX:

```
@article{rdefrein15DASCTake,
author={Ruair\'{i} de Fr\'{e}in$^\dagger$ $^{\dagger\dagger}$},
journal={2015 IEEE International Conference on Computer and Information Technology;
Ubiquitous Computing and Communications;
Dependable, Autonomic and Secure Computing;
Pervasive Intelligence and Computing, to appear},
title={Take off a Load: Load-Adjusted Video Quality Prediction and Measurement},
year={2015},
pages={9},
month={Oct},}
```

Take off a Load: Load-Adjusted Video Quality Prediction and Measurement

Ruairí de Fréin
KTH Royal Institute of Technology, Sweden
rdefrein@gmail.com

***Abstract*—An algorithm for predicting the quality of video received by a client from a shared server is presented. A statistical model for this client-server system, in the presence of other clients, is proposed. Our contribution is that we explicitly *account for* the interfering clients, namely the load. Once the load on the system is understood, accurate client-server predictions are possible with an accuracy of 12.4% load adjusted normalized mean absolute error. We continue by showing that performance measurement is a challenging sub-problem in this scenario. Using the correct measure of prediction performance is crucial. Performance measurement is miss-leading, leading to potential over-confidence in the results, if the effect of the load is ignored. We show that previous predictors have over (and under) estimated the quality of their prediction performance by up to 50% in some cases, due to the use of an inappropriate measure. These predictors are not performing as well as stated for about 60% of the service levels predicted. In summary we achieve predictions which are ≈50% more accurate than previous work using just ≈2% of the data to achieve this performance gain –a significant reduction in computational complexity results.**

I. Introduction

Understanding and predicting performance metrics for telecom clouds services is a challenging, open problem [1], [2]. The authors of [1] take a Statistical Learning (SL) approach –they apply variants of the Lasso [3], Random Forests [4] and Ridge Regression [5]– to predict the client-side metrics for a video streaming service. Their approach is significant as they collect statistics from the Linux kernel of a server machine to achieve this; their initial prediction performance results are promising; and finally, they have made their traces publicly available. To evaluate the performance of their video quality metric prediction algorithms they designed three sub-components: a test-bed, a video quality metric prediction model and learning algorithm, and finally, a measurement approach for evaluating the quality of the predictions in a fair way. The focus of this paper is on improving the performance of the second and third component above given that the test-bed, described in [1], and resulting traces, have been used in several papers and have gained acceptance.

Related work: The position of Yanggratoke et al., in [1] is that 1) by collecting thousands of kernel variables their prediction approach is service independent (they can omit service specific instrumentation etc.). We illustrate that this service-independent-prediction assumption does not always hold, that comparison of the predictor's performance across services is not justified, and that unless the service is modelled correctly, artifacts are introduced into the predictor and the measurement system. 2) The prediction of client-side metrics such as RTP packet rates (with 10%–15% error) across different scenarios and loads are possible. We demonstrate that if the approach in [1] is adopted the results are in fact dominated by the choice of scenario and load when prediction is performed. In short, this approach may inadvertently *game* the performance of the predictor positively based on the selection of a favourable scenario; or, on the other hand lead to the unfair dismissal of an approach due to an unfavourable scenario.

Domain knowledge is often inferred and then used in Signal Processing [6] and Computational Finance [7]; these techniques are referred to as *Blind* inference, learning and prediction [8]. Blind inference has not yet been widely embraced by the Network Management community. This is due to the number of different active network services, and also, whether or not it is feasible to model highly dynamic network services automatically. We desire learning algorithms that are powerful enough to learn from data without any domain knowledge or human intervention, namely *Blind* [9], [10] or *autonomic* approaches. With out loss of generality, to evaluate our approach we place an adaptive sinusoidal user request load on a video server; in practice the load is an arbitrary trace. To improve prediction performance it is reasonable to assume that we need accurate knowledge of this trace. Somewhat suprisingly we show that a Blind inference procedure is not necessary in our scenario to estimate the trace. We obtain a good estimate of it from the TCP socket count of the server. Our non-blind approach uses exactly the same information as the previous work [1], prediction accuracy improvement is obtained *for-free*, and the approach is applicable irrespective of the load on the system. We agree with the authors in [1] that collecting statistics from the Linux kernel and client-side, and learning the mapping between them, is a promising first approach. Intrusion detection systems have a long history of using such parameters –sequences of system call executed by running processes– to discriminate between normal and abnormal operating characteristics of UNIX programs [11]. However, a general model, which accounts for network, service and client delay is needed. The authors focused on the simple instantaneous case in [1] as the lab configuration considered has sufficient resources for these delays to have little affect.

The current trend of running software systems on general purpose platforms without real-time guarantees with the expectation that one can safeguard revenues, is dichotomous. The choice of video service level prediction by [1], as an exemplar instance of this problem, is timely given that Cisco

Dr de Fréin is affiliated with TSSG, Waterford Institute of Technology, Ireland. This work was supported by an ELEVATE Irish Research Council International Career Development Fellowship co-funded by Marie Cure Actions award "EOLAS": ELEVATEPD/2014/62.

[12] predicts that network traffic volumes in the order of tens of exabytes are not that far off, and 90% will be video related [13]. A SL approach –similar to [1]– is preferable to developing and fitting complex analytical models for the different layers of soft/hardware in these complex systems. The authors of [14] make the case that modern multi-core (parallel online) learning algorithms are limited by the bandwidth bottleneck. It is hard to justify expending bandwidth resources on predicting why a service is not meeting service level agreements if this bandwidth could be used to meet the service delivery shortfall. If a SL approach with low complexity, which increases linearly in the feature set size, is unable to perform predictions with low enough latency (using one of the computational architectures in [15]), it is unlikely that a significantly more complex, hierarchical analytical model will exhibit sufficiently good performance; our philosophy is to explore the simplest approach in depth first before we discard it. As a first result we demonstrate that we can outperform the results in [1] by using just 2% of the data, which contradicts the assertion that we need large amounts of data for successful SL. This performance gain is achieved by incorporating a small amount of –already present– knowledge into the SL algorithms.

The application of SL for prediction in cloud and network environments is in its infancy. A method for identifying and ranking servers with problematic behavior is proposed in [16]. The authors use Random Forest classifiers to select candidate servers for modernization. A predictive model is then used to determine the impact of modernization actions. A Support Vector Regression predictor is used in [17] to perform lightweight TCP throughput prediction. Prediction is based on prior file transfer history and measurements of path properties. A method for modeling application servers in order to detect performance degradation due to aging is presented by [18]. The authors use classification algorithms to perform proactive detection of performance degradation. Finally, the authors attempt to reduce the size of the data-stream that is forwarded to an operators' operations support system by removing uncorrelated noise events in [19]. A heuristic cross-correlation function determines the degree of inter-relationship between the events in the data-stream. To the best of our knowledge there is no work which explicitly deals with the effects of adaptive loads on these systems.

Contribution 1: We claim that a different prediction model should be used when there is a different load on the system under observation. We propose a simple hierarchical model, namely Load-Adjusted RR (LA-RR), and demonstrate performance gains of up to 30% are achievable using our model over traditional Ridge Regression (T-RR). **Contribution 2:** We propose a performance measure for analyzing the performance of the new model, namely Load-Adjusted Normalized Mean Absolute Error (LA-NMAE). Empirical results support the claim that the Traditional Normalized Mean Absolute Error (T-NMAE) is an inappropriate performance measure. We quantify how big of an issue this is. **Contribution 3:** We complete our study by proposing a new hierarchical prediction algorithm, LA-RR. We compare the performance of the measure LA-NMAE when evaluating the RR algorithm [20] used in [1], namely T-RR, with our new load-adjusted hierarchical solver, LA-RR. We also compare the performance of LA-RR to the previously proposed RR technique in [1], T-RR, using the performance measure in [1], T-NMAE. We do not exhaustively evaluate each of the methods in [1] because: 1) the solver cannot correct the formulation of the problem; 2) T-RR gives the best performance on the periodic load traces used in [1]; 3) a thorough empirical comparison of the Lasso [3], for example, with RR, involves the selection of different regression parameters for each algorithm. We have focused on comparing RR for both the model in [1], T-RR, and our hierarchical model, LA-RR, using the same regularization parameter for both, as the purpose of this paper is to motivate the candidacy of our hierarchical load-adjusted statistical model.

TABLE I. ACRONYMS: LOAD-ADJUSTED & TRADITIONAL STATISTICAL MODEL/ALGORITHM & PERFORMANCE MEASURE.

	Statistical Model/Algorithm	Performance Measure
	Ridge Regression	NMAE
Load-adjusted (new)	LA-RR	LA-NMAE
Traditional (old)	T-RR	T-NMAE

Organization: This paper makes both a theoretical and practical contribution. It starts by introducing the theoretical tools we need to perform improved predictions in Section II, III and IV. We introduce a statistical model for the client-server system in Section II. In Section III we support this model empirically using a statistical test which compares the probability of our model being valid given the data (in [1]), with the probability of the state-of-the-art model being valid given the same data. This test finds that the state-of-the-art model is implausible, given the data, and that our new model is more plausible. The second part of our theoretical contribution, in Section IV, demonstates that prediction performance measurement is challenging. In Section IV we illustrate that prediction performance measurement is highly dependent on the load on the system. We introduce a new measure to account for this type of error. We continue by showing, using the data in [1], that using the inappropriate performance measure may unjustifiably inflate or deflate the quality of our predictions. In the final two sections we introduce some practical tools for making service level predictions. We introduce a practical hierarchical RR prediction technique in Section V which follows from the analysis in Section II, III and IV. We perform a thorough simulation study that empirically evaluates and compares this technique with T-RR in Section VI. In this empirical study we use exactly the same information as [1] and our improved model yields significant performance gains.

II. SYSTEM MODEL: LA-RR

A client is connected to a server via a network and s/he requests Video on Demand, which runs on the server in [1]. Assume for the purpose of exposition that the system is operating under a light to medium load (we relax this assumption later); we, like the authors of [1], do not model the network state. What happens when a client requests video? The response of the server, with respect to kernel metric n, *the n-th feature*, to one request for video at time i is expressed as:

$$\mathbf{x}_i[n] = \hat{\mathbf{u}}_i[n] + \boldsymbol{\epsilon}_i[n], \qquad \text{where } i \in \mathbb{Z}, \mathbf{x}_i[n], \hat{\mathbf{u}}_i[n] \in \mathbb{R}. \quad (1)$$

A feature refers to a metric on the operating system level, for example, the number of active TCP connections. The feature set $\mathbf{x}_i[n]$ is constructed using the System Activity Report[1a]

[1a] http://linux.die.net/man/1/sar; [1b] http://www.videolan.org/vlc; [1c] http://www.ntp.org/

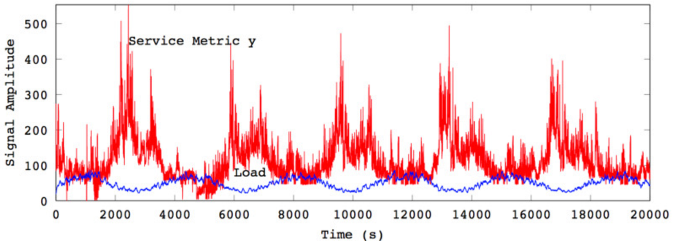

Fig. 1. Service level metric and system load trace for a periodic-type load.

(SAR) which computes system metrics over a given time interval. The term $\mathbf{x}_i[n]$ denotes the n-th feature at time index i. On the client-side we observe an application level metric, the RTP packet rate, y_i at time i. VLC[1b] media player provides Video-on-Demand on the test-bed in [1].

Problem Statement: The objective of this paper is to predict unseen values of y_i using the features $\mathbf{x}_i[n], \forall n$. We assume that a global clock can be read on both the client and server to match up the $\{\mathbf{x}_i[n], y_i\}$ pairs.

The signal $\hat{\mathbf{u}}_i[n]$ in (Eqn. 1), a square-wave (off-on-off) signal, corresponds to an increase in the CPU workload, for example, an extra X units per additional user for the duration of the video requested by the user. We assume that $\hat{\mathbf{u}}_i[n]$ is scaled in order to account for the sensitivity of a given feature to the effect of adding a new user to the system. This scaling is specific to each service, feature and machine. The signal, $\boldsymbol{\epsilon}_i[n]$, captures deviations from the ideal performance of the server with respect to the n-th feature. We assume that this deviations signal is normally distributed with 0 mean and variance σ^2. If there is more than one deviation signal, they are uncorrelated. For example, for two simultaneous video requests (of the same duration) the response of the n-th feature is

$$\mathbf{x}_i[n] = 2\hat{\mathbf{u}}_i[n] + \boldsymbol{\epsilon}_i[n,1] + \boldsymbol{\epsilon}_i[n,2]. \tag{2}$$

The deviation from the ideal performance due to the second user is $\boldsymbol{\epsilon}_i[n,2]$. A video server is only really useful if (up to K) clients can start and stop watching video at arbitrary times, simultaneously. That is, the server must be able to deal with time-varying loads. Let $K(i)$ be the number of user requests being serviced at time i. It follows that the response of the n-th feature to this load is

$$\mathbf{x}_i[n] = K(i)X + \sum_{k=1}^{K(i)} \boldsymbol{\epsilon}_i[n,k]. \tag{3}$$

We drop the square-wave and use the more flexible and general notation $K(i)X$, the number of active users at time i times the resources one user uses, X. We call this signal $\mathbf{l}_i[n] = K(i)X$, the load signal. Traces are available from an independent study in which the traces have a strong sinusoidal-like component [1]. The service level, RTP Packet Count, and load, TCP socket count, are plotted in Fig. 1. To fix ideas, we propose that a simple model for the load in these traces has the form

$$l_i[n] = K(i)X \approx a[n]\cos(\omega_n i + \phi_n) + c. \tag{4}$$

The observed n-th feature is the linear combination:

$$\mathbf{x}_i[n] = a[n]\cos(\omega_n i + \phi_n) + c + \hat{\mathbf{x}}_i[n]. \tag{5}$$

The real-valued scalars, $a[n] \in \mathbb{R}$, $\omega_n \in \mathbb{R}$ and $\phi_n \in \mathbb{R}$, are the amplitude, radial frequency, and phase of the load –they describe the user video request pattern. The constant c ensures that $l_i[n]$ is a positive signal; the demand for resources should not be negative. We make the simplifying assumption that these parameters are constant. In a real-world system this is unlikely to be true, but it serves as a good first approximation. We also simplify our notation by introducing the notation $\hat{\mathbf{x}}_i[n] = \sum_{k=1}^{K(i)} \boldsymbol{\epsilon}_i[n,k]$, for the aggregate deviation.

This model is general: The amplitude scales the load to give it a response in the correct range for the n-th feature; if the n-th feature is not a function of the load, $a[n] = 0$, and the n-th feature is $\mathbf{x}_i[n] = \hat{\mathbf{x}}_i[n]$ in system (Eqn. 5). The phase ϕ_n may capture the network and machine delay between when the request is made and the response given (cf. Fig. 1). This model is further generalized by considering loads which are parametric signals and/or stochastic processes. The service level metric is a linear function of the set of features (where the effects of the network are ignored as it is assumed to be sufficiently well resourced). The service level metric is:

$$y_i = \sum_n \mathbf{w}[n] \left\{ \overbrace{(a[n]\cos(\omega_n i + \phi_n + \varphi_n) + c)}^{l_i[n]} + \sum_{k=1}^{K(i)} \boldsymbol{\epsilon}_i[n,k] \right\} \tag{6}$$

The additional phase terms $\varphi_n, \forall n$ capture the delay in the effect of the load due to client requests on the server machine, and network and server delays. In this paper we assume $\phi_n = \varphi = 0$ as the bandwidth is assumed to be large enough (due to the light load assumption). The clocks of the server and client are synchronized in [1] using NTP[1c] and samples are collected every second. Note that we can substitute in an arbitrary expression for the load in (Eqn. 6) and the analysis in the rest of the paper holds.

System Characterization: We have introduced a deviations signal for each feature, $\hat{\mathbf{x}}_i[n]$, to explain the deviation of each feature from its ideal performance for a given load. A deviations signal is also required for the service level metric, $\hat{y}_i$. We want to explain the signal that captures the deviation of the service level metric from its ideal performance, as a function of the effect of the user requests on each of the video server's features (the deviation of each of the features from their ideal performance). The following model states that the observed deviation in the service metric is a weighted sum of the external causes, e.g. deviations in the features, plus internal causes, η_i, which captures non-idealities on the client's side.

$$\hat{y}_i = \mathbf{w}^T\hat{\mathbf{x}}_i + \eta_i. \tag{7}$$

However this model is significantly different from the model in (Eqn. 6). We make this explicit by indicating what we want (don't want) to model on both sides of the system:

$$\underbrace{y_i}_{\text{observed}} = \overbrace{\mathbf{w}^T\mathbf{l}_i}^{\text{don't want}} + \overbrace{\hat{y}_i}^{\text{want}} = \underbrace{\overbrace{\mathbf{w}^T\mathbf{l}_i}^{\text{don't want}} + \overbrace{\mathbf{w}^T\hat{\mathbf{x}}_i}^{\text{want}}}_{\text{observed features}} + \eta_i. \tag{8}$$

The problem is that the signals that we observe, the pairs $\{\mathbf{x}_i, y_i\}$, are mixtures of what we want, $\hat{\mathbf{x}}$, and a high energy load component, $\mathbf{l}$, which we do not want. The reason why we distinguish between these two problems is that the load may potentially drown-out the deviation signals, $\hat{y}_i$, $\hat{\mathbf{x}}_i[n], \forall n$. In general a good approximation of the load is known, and therefore, there is little point in approximating it if it is already known, or worse, letting it bias the learning algorithm. The ability to estimate and predict deviations from ideal performance is the problem that is crucial to solve. The approximation of the load comes from the TCPSCK field of the UNIX SAR command. The TCP socket count gives a good indication of the load on the kernel.

A Regression Tree, Random Forest [4], Lasso [3], RR or any other valid solver for problems of the form $y_i = \mathbf{w}\mathbf{x}_i$ or $\hat{y}_i = \mathbf{w}\hat{\mathbf{x}}_i$ will learn weights that solve the problem put to it. If the load is present in a model, when it should not be, e.g. if we pass $\{y_i, \mathbf{x}_i\}$ instead of $\{\hat{y}_i, \hat{\mathbf{x}}_i\}$ to the solver, we cannot expect the solver to correct the problem that is being asked. Asking the wrong question will generally yield the right answer for the wrong question. How can we learn a mapping between the kernel and service metrics which is independent of the load such that we can ask the right questions? Even more crucially, how can we measure the success of an approach that asks the correct questions? Are off-the-shelf measurement functions adversely affected by artifacts in the learning algorithm that arise due to the inappropriate presence of the load? In the next two sections we show that they are.

III. Using the Data to Support the Model

We investigate the extent to which the mean of the samples of the service level metric y depend on the underlying load on the system when these samples were drawn. Fig. 2 summarizes the statistics that characterize the values of y for different values of the load on the system, e.g. $\mathbf{l} = 19, \ldots k \ldots$. The set of points used to construct each box-plot is the set of values of y corresponding to a given value of the load signal, $\mathbf{l} = k$, and each set, and the set of the associated indices, are denoted

$$\mathcal{H}(y)|_{\mathbf{l}=k}, \text{ and } \mathcal{I}(y)|_{\mathbf{l}=k} \text{ respectively.} \tag{9}$$

Firstly, the mean of each of the sets, $\mathcal{H}(y)|_{\mathbf{l}=k}$, which we denote $\mu(\mathcal{H}(y)|_{\mathbf{l}=k})$, is different for each value of the load. For $19 \leq k \leq 30$ it is reasonable to assume that the model described above (in Eqn. 8) holds. However above $k = 30$, the values obtained by y generally decrease. In summary,

$$\begin{aligned} &\mu(\mathcal{H}(y)|_{\mathbf{l}=k}) \propto k, \quad \text{for } 22 \leq k \leq 30 \\ &\mu(\mathcal{H}(y)|_{\mathbf{l}=k}) \propto -k, \quad \text{for } 31 \leq k \leq 87 \end{aligned} \tag{10}$$

In words, the mean of the set of points of y for a given value of the load, $\mu(\mathcal{H}(y)|_{\mathbf{l}=k})$, is proportional to the value of the load for loads less than 30 active requests, and proportional to the negative of the load when the load is greater than 30 active requests. Alternatively fitting a quadratic of the form

$$\mu(\mathcal{H}(y)|_{\mathbf{l}=k}) \propto -a(k-30)^2 + b \tag{11}$$

using the scalars $\{a, b\}$ gives a more concise description of the behaviour of the mean of y. Increasing the order of the polynomial (in Eqn. 10) increases the quality of the fit. Fig. 2 is significant because (Eqn. 7) assumes that both $\hat{y}_i$ and $\hat{\mathbf{x}}_i[n]$ are zero-mean signals. What is clear from Fig. 2 is that the mean of the y_i depends on the value of the load that was present on the system when it was observed. We do not attempt to fit parameters to the model (Eqn. 8) or assume that the mean of y_i and the features $\mathbf{x}_i[n]$ are the same irrespective of what the load was on the system.

Hypothesis testing: We use a hypothesis test as a first demonstration that the model (Eqn. 7) is of interest. The null hypothesis, namely 'the load has no effect', as used in [1], is that the service level metric has a mean which is approximately equal to the $\mu(y) = 119.44$ irrespective of which samples are used to approximate it. We assume that the population standard deviation $\sigma(y)$ is unknown; we approximate it with the sample standard deviation. The value 119.44 is the mean of the ≈ 50k observed values of y. In words if we select any N_s samples of the signal y it should give a good estimate of $\mu(y)$. The alternative hypothesis, 'the load has an effect', is that we believe that the load has an effect on the values of y. The mean of the signal conditional on the load $\mathbf{l} = k$ is $\mu(\mathcal{H}(y)|_{\mathbf{l}=k})$. We also need the sample standard deviation of the service metric y conditional on the load, which is $\sigma(\mathcal{H}(y)|_{\mathbf{l}=k})$. In summary, our hypotheses are

$$\begin{aligned} H_o: &\quad \mu(y) \equiv \mu(\mathcal{H}(y)|_{\mathbf{l}=k}) = 119.44\,\forall k \\ H_a: &\quad \mu(y) \not\equiv \mu(\mathcal{H}(y)|_{\mathbf{l}=k}), \, \forall k, k \neq k^\star \end{aligned} \tag{12}$$

Does the load have an effect on the mean? Let us consider whether or not to accept or reject the null hypothesis. If the null hypothesis is true, what is the probability that we would have measured $\mu(\mathcal{H}(y)|_{\mathbf{l}=k})$ as our estimate of the mean, $\mu(y)$. If the probability of the null hypothesis is really small we can reject it. We compute the z-statistic for each k

$$Z = \sqrt{N_s}\left(\frac{|\mu(y) - \mu(\mathcal{H}(y)|_{\mathbf{l}=k}|)}{\sigma(\mathcal{H}(y)|_{\mathbf{l}=k})}\right) \tag{13}$$

and tabulate the associated probabilities in Table II. We compute the probability that the null hypothesis is true for values of the load that arise more than 100 times in the traces (and our choice of the z-statistic is justified). The values of the load for when this is true are indicated. The probability that the null hypothesis is true is zero in every case except for when the load is $k \in \{41, 42, 43, 44\}$, that is when $\mu(\mathcal{H}(y)|_{\mathbf{l}=k}) \approx \mu(y)$. Fig. 3 illustrates the difference between the means. When $k = k^\star = 42$, then $\mu(y) \approx \mu(\mathcal{H}(y)|_{\mathbf{l}=k^\star}$ with $\approx 14\%$ chance. In every other case applying RR to the data assuming that $\mu(y) \approx \mu(\mathcal{H}(y)|_{\mathbf{l}=k^\star}$, and thus identically distributed, is valid with less than 2% chance in one case, and $\approx 0\%$ in all other cases [2]. This analysis motivates the following conclusions: 1) Different conditional means and variances for each load value imply that we need to learn a different regression model for each value of the load. 2) The linear model described in the previous section is insufficient because the RTP Packet Rate can increase with the number of active users until $k = 30$. Above this, the system begins to become saturated and the RTP Packet Rate begins to decrease as the number of active users increases. A piece-wise linear model, or some higher order polynomial model is required. Fig. 3 illustrates how good an estimate of the mean of the entire set of samples, $\mathcal{H}(y)|_{\mathbf{l}=k}$ is.

IV. Prediction Quality Measurement: LA-NMAE

Using the correct measure of prediction performance is crucial if we are to distinguish between the performance of

[2] Remark: Our application of the z-statistic has a number of drawbacks. There is correlation between the samples which affects the conditional standard deviation of each sample. The sets $\mathcal{H}(y|_{l=k})$ are in some sense anti-correlated and so the samples that we choose are not independent. The sample size used to generate each p-value is in general different, which affects the resolution of our estimates (under/over estimation of the sample standard deviation). Despite these shortcomings, the p-value gives a very strong recommendation that we reject the null hypothesis; for all loads but one. The assumption that the trace values are identically distributed holds with probability 0.

TABLE II. What is the probability that the mean value of a set of samples drawn under load $\mathbf{l}$ have the mean $\mu(y)$? Only when $\mathbf{l} \in \{41, 42, 43, 44\}$ is the likelihood non-zero. If the null hypothesis is indicative of the assumptions made by the learning algorithm, they do not hold.

	load values l																																																									
k	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	**41**←	**42**←	**43**←	**44**←	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
p-val.	3.01e-13	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	4.57e-10	1.80e-03	1.45e-01	2.10e-02	9.01e-06	2.21e-08	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00

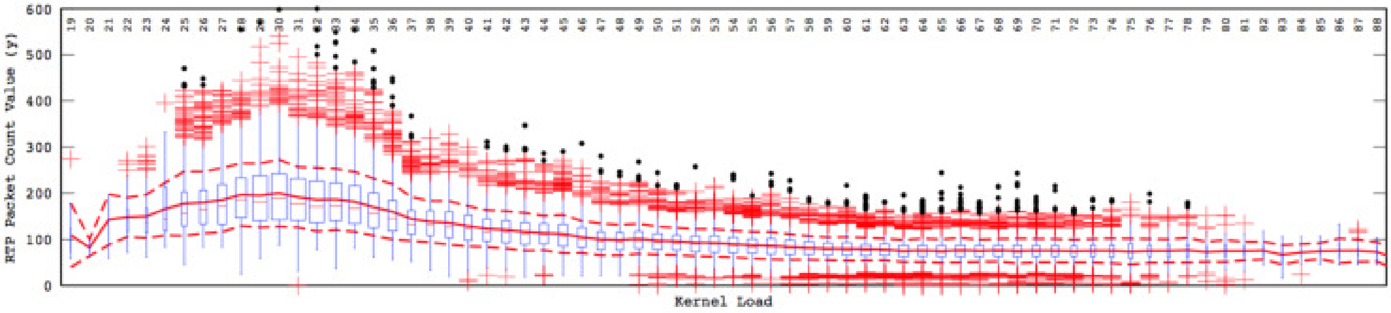

Fig. 2. The distribution of the service level metric y is illustrated for a range of loads on the system $19 \leq \mathbf{l} \leq 88$. The largest load observed was 110, but above $\mathbf{l} > 88$ there were too few samples to generate box-plots. X-tick labels are removed (for plotting purposes) as certain load values are never observed. The load values are indicated above each box-plot. We plot the mean (full line) and 1 standard-deviation (dashed lines).

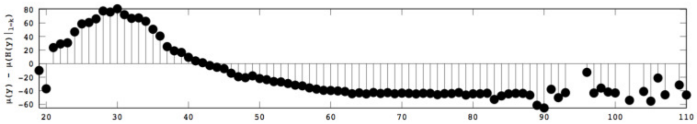

Fig. 3. Difference between the mean of the entire set of service level samples, $\mu(y)$, and the service level samples conditional on the mean, $\mu(\mathcal{H}(y)|_{\mathbf{l}=k})$. For each load value a stem is drawn from zero to the amplitude of the difference. In most cases, the stem length is greater than ±20 units.

competing prediction algorithms with confidence. Previous works have considered the T-NMAE between the signal to be predicted, y_i and the prediction estimate, $\hat{y}_i$

$$S = \frac{100}{\mu(y)N_s}\left(\sum_i |y_i - \hat{y}_i|\right) = \sum_i \frac{100}{\mu(y)N_s}|y_i - \hat{y}_i| \quad (14)$$

They scale the score by 100 to obtain a percentage. Reordering the summation and the constant is useful as it makes the argument below more intuitive.

In this section we show that 1) the load dependence of the mean described above renders the T-NMAE measure suspect to overestimation of the prediction accuracy in many cases; 2) the dependence of the T-NMAE on the population mean $\mu(y)$ potentially dominates the sensitivity of the T-NMAE measurement to just one (unimportant) statistic of the signal being predicted. Given that the first step of many estimation procedures is to center the signal, it is troubling for the mean to have such a dominant effect on prediction performance.

1) Load dependence: The T-NMAE produces an aggregate score for the prediction error between y_i and $\hat{y}_i$ for a set of signal values y_i, $i = 1, \ldots N_s$. The underpinning assumption is that the signal values to be estimated are picked from the same distribution. Each prediction produces an error $\epsilon_i = y_i - \hat{y}_i$. The T-NMAE assumes that each prediction error is equally important, because the signal values y_i are taken from the same distribution. Therefore each error is scaled by $\frac{100}{N_s}$. Finally, in order to give this number context, the T-NMAE scales the weighted error by the typical value that the signal, y_i, achieves, e.g. $\mu(y)$, and sums up the values. Herein lies the problem however. When the load affects the mean in the manner we described above, $\mu(\mathcal{H}(y)|_{\mathbf{l}=k})$, the value of the "typical value" of y_i changes too, as a function of the load. For a given error in our prediction, the global mean, $\mu(y)$ and the load dependent mean scale the error differently. So how does the load, in particular the mean dependent load, $\mu(\mathcal{H}(y)|_{\mathbf{l}=k})$, affect the contribution of the error in one sample to the T-NMAE? To answer this question, we consider the maximum possible prediction error in the trace above. For a given value of the load, e.g. $\mathbf{l} = k$, the maximum value is $\max \mathcal{H}(y)|_{\mathbf{l}=k}$ (assuming y is non-negative). The NMAE should depend on the load, because failing this, some errors are scaled unfairly by a mean value which is not representative of the distribution from which they were drawn. Therefore the load-adjusted NMAE (LA-NMAE) is

$$S_k = \sum_{i \in \mathcal{I}(y)|_{\mathbf{l}=k}\}} \frac{100}{N_s \mu(\mathcal{H}(y)|_{\mathbf{l}=k})}|\epsilon_i|. \quad (15)$$

In this new measure, the LA-NMAE, the errors ϵ_i due to the predictions $\hat{y}_i$ correspond to the samples y_i which were generated under the load condition $\mathbf{l} = k$. To evaluate how important it is to select the correct mean in the LA-NMAE, we vary the error ϵ_i, as a function of each value of the load, from 0 to $\max \mathcal{H}(y)|_{\mathbf{l}=k}$. We compute the NMAE using the two definitions above, for each value of the error $0 \leq \epsilon_i \leq \max \mathcal{H}(y)|_{\mathbf{l}=k}$. We plot the pairs of resulting NMAEs for each value of the load l. Fig. 4 illustrates how both NMAEs penalize errors in the case where the load is $49 \leq l \leq 56$. These results should be read as follows. If the absolute error in the prediction, *irrespective of which prediction algorithm was used to generate the prediction*, for a given value of the load l, was e, the associated contribution of this value to the total T-NMAE is $p\%$ if the global mean was used, e.g. $\mu(y)$. The percentage p can be found by finding the y-value of e on the black dashed line. Whereas if the load-adjusted mean was used $\mu(\mathcal{H}(y)|_{\mathbf{l}=k})$, the percentage, p, may be obtained by finding the y value corresponding to e using the blue line.

Some crucial observations are listed as follows: 1) The full

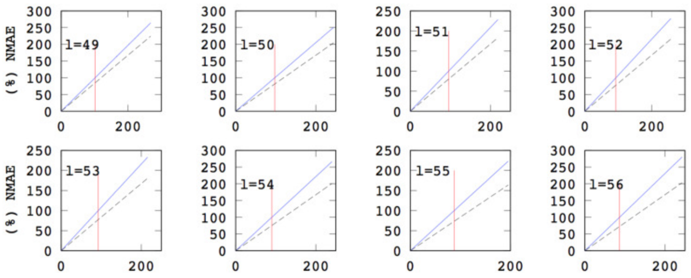

Fig. 4. For a given prediction error and system load condition, what is the contribution of the error to the prediction accuracy score? The traditional NMAE (T-NMAE), dashed line, under-estimates the contribution of a prediction error to the score for many load conditions compared to the Load-Adjusted-NMAE (LA-NMAE). We illustrate this for loads $49 \leq \mathbf{l} \leq 56$.

TABLE III. PATHOLOGICAL PREDICTION PROBLEM SOLVER

```
1. Initialization: Set the constant $a = 0$, store the serv. metrics: $y_{orig} = y$.
2. Pick $N_s$ values from any distribution (e.g. a normal dist.), assign $\hat{y}_i \leftarrow \mathcal{N}(0,1)$,
3. Assign $a = a + 1$;
4. Assign $y_i \leftarrow y_i + a$ and $\hat{y}_i \leftarrow \hat{y}_i + a$
5. Compute the T-NMAE $S = \sum_i |y_i - \hat{y}_i| \frac{100}{N_s \mu(y)}$
6. If $S < 1$ break (the prediction is $\hat{y}_i$); else return to step 2.
```

line is higher for all prediction errors, ϵ_i, for approximately 74.7% of the values the load can take and 61.295% of all of the signal values; 2) The scaled-error values computed using the global mean $\mu(y)$ are significantly smaller for approximately 61% of the values used to compute the T-NMAE than they should be; 3) The prediction performance given using the global mean scaled T-NMAE is quoted to be better (using the measure S) than it actually is (using the measure S_k). It is straightforward to compute by how much S over-inflates the accuracy of the prediction algorithm compared to S_k. For a given load value $\mathbf{l} = k$ let the weight of proportionality be α

$$\alpha S_k = S, \quad \text{which implies } \alpha = \frac{\mu(\mathcal{H}(y)|_{\mathbf{l}=k})}{\mu(y)}. \tag{16}$$

In words, if we divide a given T-NMAE, S, by the inflation weight α we get the correct LA-NMAE. For example in Fig. 4, $\mu(\mathcal{H}(y)|_{\mathbf{l}=k}) < \mu(y)$ when the load is above $k^\star = 42$. Taking the case when $k = 70$, $\mu(y) \approx 119$ and $\mu(\mathcal{H}(y)|_{\mathbf{l}=k}) \approx 80$. These values imply that $\alpha = \frac{80}{119} \approx 0.66$, which means that an T-NMAE of $S = 11\%$ equals a LA-NMAE of $S_k = 17\%$, an T-NMAE of $S = 20\%$ equals a LA-NMAE of $S_k = 30\%$ and so on. In short, the correct LA-NMAE is 50% worse in many cases above, when the load-adjusted mean is used.

This is a very practical result. If the absolute prediction error ϵ is 1 unit for a particular sample the difference between the NMAEs, S and S_k, is $.5\%$; for $\epsilon = 5$, the difference is 2.5%; for $\epsilon = 20$, the difference is 10% and so on. A break down of these differences for given values of the load can be obtained in Table IV. The numbers of samples that fall into each category are listed along with the load values. For example, 822 samples in the traces are drawn under a load $k = 70$. The error in the reported error using S, when the errors range from 1 to 40 units, ranges from $.48\%$ to 19.11%. Note that the error can easily be greater than 40, and in this case the error in the reported percentage is larger.

2) Dominance of $\mu(y)$: It is not reasonable to claim that the T-NMAE allows for the comparison of prediction accuracy of different service level metrics across different scenarios and loads. We demonstrate this by considering the following pathological problem-solver pair, which illustrates the counter-intuitive behaviour of the T-NMAE S.

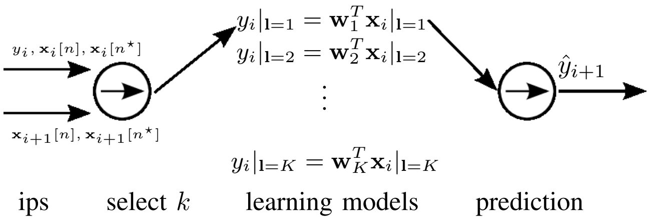
$y_i, \mathbf{x}_i[n], \mathbf{x}_i[n^\star]$
$\mathbf{x}_{i+1}[n], \mathbf{x}_{i+1}[n^\star]$
$y_i|_{\mathbf{l}=1} = \mathbf{w}_1^T \mathbf{x}_i|_{\mathbf{l}=1}$
$y_i|_{\mathbf{l}=2} = \mathbf{w}_2^T \mathbf{x}_i|_{\mathbf{l}=2}$
$\vdots$
$y_i|_{\mathbf{l}=K} = \mathbf{w}_K^T \mathbf{x}_i|_{\mathbf{l}=K}$
$\hat{y}_{i+1}$
ips
select k
learning models
prediction

Fig. 5. During the learning phase, kernel and service metrics enter on the upper LHS arrow into a switch that determines –based on the load value $\mathbf{x}_i[n^\star]$– which learning model to learn. During the prediction phase, the same 2-level approach is taken. The load, $\mathbf{x}_{i+1}[n^\star]$, and features, $\mathbf{x}_{i+1}[n]$, enter on the lower LHS arrow, and the appropriate prediction model is chosen based on $\mathbf{x}_{i+1}[n^\star]$, to produce the prediction $\hat{y}_{i+1}$ on the RHS.

Problem 1: Predict N_s values of y_i, using any prediction algorithm, such that the T-NMAE, S, of the errors $y_i - \hat{y}_i$ are less than 1%. Consider the valid approach in Table III. At first glance, Problem 1 looks like a reasonable statement of the video service level prediction we are interested in solving. Note however that the value of the mean $\mu(y)$ is crucial. As a increases, the T-NMAE goes to zero $S \mapsto 0$, in general, irrespective of what $\hat{y}_i$ is, or how it was generated. This is because $\mu(y) = \mu(y_{orig}) + a$. In the more general setting of comparing the performance of predictions of service level metrics it is clear that the mean of the service level metric observed is crucial as it sets the sensitivity of the performance to deviations in performance. In summary, the comparison of prediction performance across services is not meaningful unless the services have the same mean. If they do not have the same mean, what value is the appropriate value for the mean so that the comparison is fair? We cannot use 0 as this gives a T-NMAE of ∞. We risk inflating the performance of our predictor by picking an arbitrary value. Therefore it is not reasonable to claim that the T-NMAE allows for the comparison of prediction accuracy of different service level metrics. We draw the following conclusions: 1) prediction performance across the samples using the T-NMAE is unreliable; performance should be measured relative to the load on the system. 2) The prediction performance across services is unreliable using the T-NMAE due to the difference in the mean values of the traces. The LA-NMAE does fix the first problem with prediction performance measurement, the load; and we have raised awareness of the problems associated with comparing prediction performance across services, loads and scenarios. As an aside, we note that Signal-to-Noise-Ratio-like (SNR) measures and the Root Mean Square Error (RMSE) suffer from a similar dependance on the load. We do not give a full treatment for these measures in this paper as measures derived from the NMAE are sufficient to provide a like-for-like comparison with the state-of-the-art.

V. LOAD-ADJUSTED LEARNING AND PREDICTION

We have contributed practical results for modeling video metrics under different load conditions. We have also demonstrated how to measure the performance of the predictor under different load conditions. We now illustrate the flow of control of a learning and prediction algorithm pair that use knowledge

TABLE IV. INFLATION/DEFLATION OF RESULTS BY CHOOSING THE T-NMAE, S, OVER LA-NMAE, S_k. THE T-NMAE INFLATES THE RESULTS.

		load values l																												
		19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
$\epsilon = \lvert y_i - \hat{y}_i \rvert$	1	0.08	0.38	-0.14	-0.16	-0.17	-0.24	-0.28	-0.28	-0.30	-0.33	-0.33	-0.34	-0.32	-0.30	-0.30	-0.29	-0.25	-0.21	-0.14	-0.11	-0.10	-0.06	-0.03	-0.01	0.02	0.04	0.06	0.11	0.16	0.18
	10	0.77	3.75	-1.38	-1.63	-1.71	-2.35	-2.75	-2.82	-2.98	-3.30	-3.26	-3.38	-3.15	-3.00	-3.02	-2.87	-2.49	-2.12	-1.45	-1.15	-1.02	-0.61	-0.27	-0.10	0.19	0.39	0.55	1.12	1.61	1.76
	20	1.53	7.51	-2.76	-3.27	-3.43	-4.71	-5.50	-5.64	-5.96	-6.59	-6.51	-6.76	-6.30	-6.01	-6.04	-5.74	-4.99	-4.25	-2.90	-2.29	-2.03	-1.22	-0.55	-0.20	0.39	0.79	1.11	2.23	3.22	3.51
	30	2.30	11.26	-4.14	-4.90	-5.14	-7.06	-8.25	-8.46	-8.94	-9.89	-9.77	-10.13	-9.46	-9.01	-9.06	-8.61	-7.48	-6.37	-4.35	-3.44	-3.05	-1.83	-0.82	-0.30	0.58	1.18	1.66	3.35	4.83	5.27
	40	3.07	15.02	-5.52	-6.53	-6.85	-9.42	-11.00	-11.28	-11.91	-13.19	-13.03	-13.51	-12.61	-12.01	-12.08	-11.48	-9.98	-8.49	-5.80	-4.58	-4.07	-2.44	-1.10	-0.40	0.78	1.57	2.22	4.47	6.44	7.02
Number of samples for each value of l		10	3	10	111	209	347	561	762	900	1142	1535	1610	1607	1551	1290	1226	1101	931	824	755	790	805	823	866	944	891	931	875	789	791

		load values l																												
		49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	...
$\epsilon = \lvert y_i - \hat{y}_i \rvert$	1	0.15	0.19	0.21	0.24	0.24	0.27	0.30	0.32	0.36	0.38	0.41	0.41	0.43	0.44	0.50	0.47	0.50	0.46	0.49	0.47	0.49	0.48	0.49	0.50	0.48	0.49	0.52	0.49	0.49	...
	10	1.48	1.89	2.06	2.38	2.43	2.71	3.04	3.17	3.56	3.83	4.11	4.15	4.25	4.40	4.95	4.73	4.99	4.64	4.88	4.67	4.93	4.78	4.86	4.99	4.85	4.86	5.20	4.91	4.90	...
	20	2.96	3.78	4.13	4.76	4.87	5.43	6.07	6.34	7.13	7.66	8.23	8.29	8.50	8.81	9.91	9.45	9.98	9.29	9.76	9.33	9.86	9.55	9.71	9.98	9.69	9.72	10.40	9.81	9.79	...
	30	4.44	5.67	6.19	7.15	7.30	8.14	9.11	9.51	10.69	11.49	12.34	12.44	12.75	13.21	14.86	14.18	14.97	13.93	14.64	14.00	14.79	14.33	14.57	14.97	14.54	14.58	15.59	14.72	14.69	...
	40	5.92	7.56	8.26	9.53	9.73	10.86	12.14	12.68	14.26	15.31	16.46	16.59	17.00	17.61	19.82	18.91	19.96	18.58	19.52	18.66	19.72	19.11	19.43	19.96	19.39	19.44	20.79	19.63	19.59	...
Number of samples for each value of l		906	980	917	962	1006	984	999	967	1054	1041	968	1035	1109	1067	1116	1040	956	990	959	966	904	822	708	642	520	492	411	337	254	...

of the presence of the load to accurately: 1) model the video metrics and 2) measure the error in the resulting predictions (in Fig. 5). We perform RR with a least-squares objective function. The crucial property of our approach is that instead of learning one set of prediction weights, $\mathbf{w}$, and using them to predict video service level metrics (irrespective of the distribution from which they were drawn), we learn K sets of weights $\mathbf{w}_k$, e.g. a set for each of the possible number of coincidental active users of the system. We then use the TCPSCK feature, which has index $n^\star$ in the feature set, to determine which prediction and learning model to use in the learning and prediction phases.

Learning models: To fit a learning model for each value the load assumes we determine the set of samples corresponding to the load $\mathbf{l} = k$ in the training data, using the TCPSCK feature of the kernel. This results in the set of dependent and independent variables for each load value:

$$\{\mathcal{H}(y)|_{\mathbf{l}=k}, \mathcal{H}(\mathbf{x}_i[n])|_{\mathbf{l}=k}\}. \quad (17)$$

This set is the training data under load condition $\mathbf{l} = k$. In our experiments, we randomly take 90% of the service-kernel level metric pairs for each value of the load and fit a RR model, yielding the weights $\mathbf{w}_k$. The remaining data is test data. We save the set of mean values, $\mu(\mathcal{H}(y)|_{\mathbf{l}=k})$, and use these means in the load-adjusted measurement function (Eqn. 15). To fit each model, first we center the service level metrics and the features using the corresponding load dependent means, e.g. $\mu(\mathcal{H}(y_i)|_{\mathbf{l}=k})$ and $\mu(\mathcal{H}(\mathbf{x}_i[n])|_{\mathbf{l}=k}), \forall n$ and thus the effect of the load is removed. We fit the model

$$(y_i|_{\mathbf{l}=k}) = \mathbf{w}_k^T(\mathbf{x}_i[n]|_{\mathbf{l}=k}) \quad (18)$$

using a Least Squares solver. The learning process is summarized in Fig. 5. The service and kernel metrics enter the system using the top arrow on the LHS. The feature $n^\star$, e.g. the TCPSCK count, is used in a K-level switch to determine under which load conditions the data was generated. A model is learned for each value the load assumes, resulting in the weights $\mathbf{w}_k, \forall k$ (stacked in the center column of Fig. 5).

Prediction: Given a set of models $\mathbf{w}_k$, $k = 1, 2, \ldots K$, and a previously unseen set of features $\mathbf{x}_{i+1}[n], \forall n$ at time index $i+1$, we desire a prediction of the value of y_{i+1}. We denote the prediction $\hat{y}_{i+1}$. In Fig. 5 the TCPSCK value $\mathbf{x}_{i+1}[n^\star]$, associated with the service level metric to be predicted, y_{i+1} enters the prediction system on the LHS using the bottom arrow at time $i+1$. The K-level switch selects the appropriate prediction model, $\mathbf{w}_k$. An estimate of $y_{i+1}|_{\mathbf{l}=k}$ is generated using $\mathbf{w}_k$ and $\mathbf{x}_{i+1}[n]$, and presented on the RHS $\hat{y}_{i+1}$.

Discussion: One consequence of using a K-level switch to learn K models is that we have less data to train each model. It is important to clarify that having less data is not necessarily a bad thing if the data that we have is the correct data. If the load assumes 1 of 80 different levels with equal probability and we use all of the data to fit a RR model for a given load value, only 1.25% of the training data that we use is drawn from the distribution we want to be able to predict from.

VI. NUMERICAL EVALUATION

We present two results: 1) We compare the performance of our LA-RR routine with T-RR. T-RR, used in [1], makes no assumptions about the effects of the load on the system. 2) We demonstrate the performance of the LA-NMAE prediction measurement function and the T-NMAE prediction measurement function using our LA-RR routine and T-RR.

LA-RR vs T-RR: For the LA-RR routine, we adopt the load-adjusted learning and prediction algorithms described in the previous section. We pre-process the trace by removing all non-numeric and constant valued features from the set of features. There are 231 remaining features. We learn a model for each load value, e.g. a set of weights $\mathbf{w}_k$, if the set of training data associated with a particular load value, e.g. the pairs $\{\mathcal{H}(y)|_{\mathbf{l}=k}, \mathcal{H}(\mathbf{x}_i[n])|_{\mathbf{l}=k}\}$, has greater than 231 entries (the total set of data for a given load value should therefore be > 257 entries). We partition the data using a 9:1 split of training to test data. We perform Monte Carlo trials by re-training the learning models (and performing prediction) on 100 randomly generated training-test data splits. For each Monte Carlo experiment, we predict all of the values in the test set and use the accuracy of the predictions as the basis for our evaluation. We use the statistics of the prediction performance for both the LA-RR and the T-RR, which we describe below, to compare the two learning and prediction procedures. Given that we center the service and kernel metrics, we fit the LA-RR routine without the bias parameter [3]. The hypothesis that we test is that if the presence of the load is acknowledged by the learning algorithm, and predictions are made conditional on the load, $\mathbf{l} = k$, the predictions are better than if we deploy an algorithm that ignores the load. To test the validity of this hypothesis, we apply T-RR to the same data. In order for the comparison to be fair, we use a training and test set which is the same size as the training and test set for the LA-RR. However, similar to the approach in [1], we generate the training and test sets using any sample from the trace, irrespective of what the load value was for that sample. If the load plays no role in the way that the samples were generated and all samples in the trace are drawn independently from an identical distribution, this T-RR approach is valid. We set the regularization for both RR algorithms to be 5000. This is the smallest value that ensures that RR is well-conditioned (for T-RR).

Comparison of LA-RR and T-RR: We tabulate some statis-

TABLE V. LA-RR: COMPARISON OF LA-NMAE, T-NMAE, DIFFERENCE IN NMAE MEASUREMENT.

	←−load values l−→																									
	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49
$S_k(\%)$	14.20	13.58	12.43	13.38	13.16	13.71	13.26	13.86	13.88	14.96	13.48	14.81	15.05	13.60	14.05	14.93	15.03	15.81	16.64	14.90	15.58	16.02	16.39	16.89	16.58	15.85
$S(\%)$	19.76	20.22	18.74	20.77	21.71	22.43	22.23	22.22	21.64	23.40	20.51	21.09	20.16	16.45	16.28	16.99	16.21	16.34	16.85	14.56	14.88	15.02	14.46	14.16	13.70	13.47
$S - S_k(\%)$	**5.56**	5.56	6.64	6.31	7.39	8.55	8.73	8.97	8.36	7.76	8.44	7.03	6.28	5.11	2.85	2.23	2.06	1.18	0.54	0.20	**-0.34**	**-0.70**	**-0.99**	**-1.93**	**-2.72**	**-2.87**

	←−load values l−→																										
	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76
$S_k(\%)$	17.00	16.12	16.41	16.35	17.69	16.57	16.72	16.34	15.27	15.87	16.19	16.66	16.38	16.10	16.79	16.73	18.06	16.74	16.65	18.68	18.10	17.12	18.71	19.21	18.69	20.63	19.39
$S(\%)$	13.87	12.94	12.78	12.67	13.36	12.16	12.13	11.46	10.48	10.64	10.82	11.05	10.74	10.12	10.73	10.48	11.61	10.58	10.69	11.76	11.53	10.83	11.72	12.17	11.83	12.72	12.23
$S - S_k(\%)$	**-2.38**	**-3.13**	**-3.19**	**-3.63**	**-3.68**	**-4.33**	**-4.41**	**-4.59**	**-4.88**	**-4.79**	**-5.23**	**-5.36**	**-5.61**	**-5.65**	**-5.99**	**-6.06**	**-6.25**	**-6.44**	**-6.17**	**-5.96**	**-6.92**	**-6.58**	**-6.28**	**-6.99**	**-7.04**	**-6.87**	**-7.90**

TABLE VI. T-RR: COMPARISON OF LA-NMAE, T-NMAE, DIFFERENCE IN NMAE MEASUREMENT (NAN: NOT A NUMBER).

	load values l																									
	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49
$S_k(\%)$	> 100	15.27	15.65	14.23	14.18	13.75	13.96	13.93	14.69	14.76	15.19	15.01	14.28	20.61	16.14	16.31	16.30	17.34	16.30	17.77	17.23	17.43	17.28	18.57	25.94	20.56
$S(\%)$	> 100	23.03	24.29	23.48	23.21	23.06	22.39	21.72	22.98	22.46	21.63	20.10	17.27	23.88	18.37	17.59	16.85	17.55	15.93	16.97	16.16	15.38	14.49	15.35	22.05	16.77
$S - S_k(\%)$	> 100	7.75	8.64	9.25	9.03	9.30	8.43	7.79	8.29	7.70	6.44	5.10	2.99	3.27	2.23	1.28	0.55	0.21	**-0.37**	**-0.80**	**-1.07**	**-2.05**	**-2.79**	**-3.22**	**-3.89**	**-3.79**

	load values l																										
	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76
$S_k(\%)$	17.64	17.40	17.30	22.09	21.24	25.33	18.52	17.57	17.01	17.54	19.24	18.61	18.15	20.44	20.69	20.35	20.11	19.96	20.49	19.57	29.92	20.23	39.58	26.37	31.35	39.78	NaN
$S(\%)$	14.15	13.55	13.41	16.68	15.59	18.37	12.99	12.06	11.41	11.73	12.76	12.20	11.40	13.07	12.97	13.09	12.71	12.82	12.90	12.46	18.93	12.68	25.07	16.68	19.34	25.08	NaN
$S - S_k(\%)$	**-3.49**	**-3.85**	**-3.90**	**-5.41**	**-5.65**	**-6.96**	**-5.53**	**-5.51**	**-5.61**	**-5.81**	**-6.48**	**-6.41**	**-6.75**	**-7.38**	**-7.73**	**-7.26**	**-7.41**	**-7.14**	**-7.59**	**-7.11**	**-10.98**	**-7.56**	**-14.51**	**-9.69**	**-12.01**	**-14.70**	NaN

TABLE VII. LA-RR PERFORMANCE GAIN

Measure	μ_{gain}	σ_{gain}	min gain	max gain
LA-NMAE Gain	3.76%	5.57	0.03%	32.20%
T-NMAE Gain	2.77%	3.52	0.048%	20.30%

tics which describe the performance of the LA-RR and T-RR algorithms in Table VII. We use both the T-NMAE and LA-NMAE to evaluate the performance. The mean performance gain achieved by the LA-RR is 3.76% if the LA-NMAE is used and 2.77% if the T-NMAE is used to compare performance. These mean values are computed by taking the mean of the mean LA-NMAE for each load value, and the mean of the mean T-NMAE for each load value. For completeness, we list the minimum and maximum performance gain achieved by the LA-RR routine over all load values. These performance gains are positive and range from 0.03% to 32.30% for the LA-NMAE measure and 0.048% to 20.30% for the T-NMAE measure. We remove the outliers when we compute the maximum performance gain, e.g. if the performance gain is greater than 100% this has generally occurred because the T-RR routine has failed to find good prediction weights. Even with a regularization parameter which is as large as 5000, the T-RR algorithm is ill-conditioned. The LA-RR is generally successful with a much lower regularization parameter –the problem is better behaved because the load has been removed. The minimum performance gain corresponds to the case where the load is $\mathbf{l} = k^{\star}$, in this case, the load conditional mean is approximately equal to the mean value of the trace. We also list the standard deviations of the performance gains for both NMAE measures. In summary, for all values of the load, k, both the T-NMAE and LA-NMAE scores support the claim that LA-RR outperforms T-RR. In some cases, the performance gain is 30%, which is significant.

Comparison of prediction performance metrics: We have established that on average the LA-RR outperforms the T-RR routine for all values of the load irrespective of whether the LA-NMAE measurement function is used, or the T-NMAE measurement function is used. We examine the performance of the LA-NMAE and T-NMAE measurement functions in more detail. Tables VI and V list the LA-NMAE, S_k, the T-NMAE, S, and the difference between the T-NMAE and LA-NMAE, $S - S_k$. We draw the following conclusions:

1: In both Table VI and V, S_k the LA-NMAE, is lower than S, the T-NMAE, when the load is less than $\mathbf{l} = 39$. When the load is low, the LA-NMAE states that the prediction performance is better than the T-NMAE. Using the measure, S, penalizes the prediction algorithm unfairly for when the load on the system is small. Unfortunately there are more load values greater that $\mathbf{l} = 39$ than less than $\mathbf{l} = 39$.

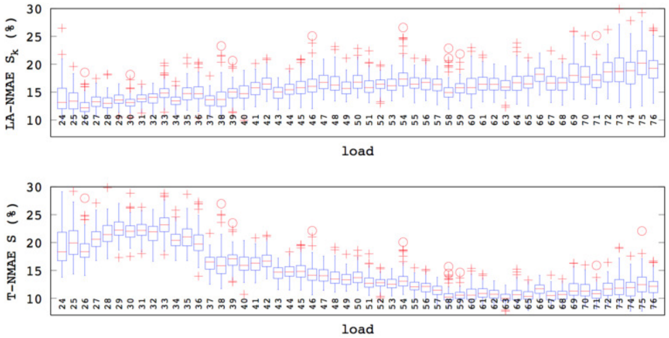

Fig. 6. T-NMAE is correlated with the $\mu(\mathcal{H}(y)_1)$, LA-NMAE is not.

2: In both Table VI and V, S_k the LA-NMAE is higher than S, when the load is $\mathbf{l} > 39$. When the load is high, the T-NMAE states that the prediction performance is better than it actually is, giving rise to inflated but unjustified confidence in the performance of the predictor. In row three of both tables, $S - S_k$ lists the difference in the prediction performance measurements for the T-NMAE and LA-NMAE. A negative percentage for a given load value, means that on average the error in the measurement of the prediction performance by the T-NMAE acts to reduce the error in the prediction performance. For example, when $\mathbf{l} = 71$ active users, the T-NMAE gives the performance as being 13.41% accurate, whereas the LA-NMAE gives the accuracy as 21.7%. This discrepancy, -8.33% leads to over-confidence, it is founded on a measure which has reduced the sensitivity of the measurement function to errors by standardizing the score with a mean value which has no relation to the distribution from which the predictions are made. Similarly, when the discrepancy is positive, it follows that the T-NMAE gives a pessimistic measure of the prediction performance.

3: In general more result inflation than result deflation occurs. For 64% of the load values, which have greater than 231 samples associated with them, the percentage of the T-NMAE, S, measurement is made smaller which inflates our confidence of the success of the prediction algorithm. Generally, the inflation is in the range 4–7%.

4: A number of high level trends warrant reporting. We plot box-plots of the spread of LA-NMAE and T-NMAE measurements in our Monte Carlo trials in Fig. 6. As the load on the system increases its behaviour becomes more volatile; prediction becomes harder because the behaviour of the system is more unpredictable. What is significant is that the LA-NMAE does not seem to penalize as a function of the parabolic nature of the relationship between the RTP Packet Count metric and the underlying load on the system (cf. Fig. 2). We posit that this indicates that the LA-NMAE measures the error in the predictions, *independently* of the artifacts introduced by considering the load. Note that the T-NMAE measure, S, increases initially until $k = 30$ and then decreases in Fig. 6. The T-NMAE's measurements are clearly correlated with the parabolic relationship between the RTP Packet Count and the underlying load on the system. It is unusual that the T-NMAE's measurement of prediction error should decrease with the load, that is, that we can predict the behaviour of the system better if the load is increased. If we accept this measure as being a reasonable measure of performance, a *misguided* conclusion we could draw from this plot is that we should be able to make better predictions of how resource behave if we have fewer resources per user. In comparison the LA-NMAE, S_k, is an approximately linearly increasing function of the load; it does not exhibit change-points like the T-NMAE measure S. When there are more active users using a fixed set of resources the performance of the resource becomes less predicable.

Implementation Considerations: In terms of prediction performance, the LA-RR gives empirically more accurate predictions irrespective of which NMAE measure is used than T-RR. It is a well-principled approach; it acknowledges the effect changing the load has on the observations drawn from the system –it is a better model. In terms of complexity and the ability to perform real-time prediction, we achieve better results for all load values using only 2% of the data; what at first looks like a big data problem, is in fact manageable even with limited bandwidth and compute resources. Instead of using ≈ 50k samples training data, we use $\approx .2$ to 1k and achieve better results. Given that the asymptotic complexity of RR is O($N^2 N_s$), when $N_s > N$, where N_s is the number of training samples and $N = 231$ is the number of features, a comparison of the complexity of LA-RR, O($230^2 \times 1000$), with T-RR, O($230^2 \times 50000$), motives the selection of LA-RR. This reduction supports the claim that real-time prediction of video service level metrics are potentially computationally achievable. Finally, we observed that T-RR quickly runs into numerical difficulty due to the condition of its matrix inverse. This is due to the presence of the load. When the load signal is dominant the matrix inverse involved in computing T-RR becomes more singular and thus a large regularization parameter was necessary to ensure non-singularity. "NaN" is included in Table. VI when the solver is ill-conditioned. However, a large regression parameter increases the chances of having a large bias [3] in the prediction weights. The LA-RR matrix inverse is generally better conditioned.

VII. Conclusions

Many SL tools exist to organize data, measure prediction performance, make predictions about future video service level metrics –essentially to guess at the state of the system (server-client metrics) that gave rise to the data. The object of any such study should be the system, and not artifacts of the way we use SL tools to probe its secrets. We have demonstrated that applying SL techniques and measurement functions, without first examining the underlying structure of the system, may introduce artifacts. Any subsequent effort to predict what the system will do next is quixotic and may lead to over-confidence. In response to this we introduced the idea of load-adjusted learning, prediction and measurement to remove these artifacts. Our empirical and analytical findings support the claim that adopting a load-adjusted processing philosophy gives more accurate prediction. They also support the assertion that if this approach is not adopted the load on the system can inadvertently affect the result. Given the recent interest in leveraging SL in Networking this results warrants reporting.

References

[1] R. Yanggratoke, J. Ahmed, J. Ardelius, C. Flinta, A. Johnsson, D. Gillblad, and R. Stadler, "Predicting real-time service-level metrics from device statistics," *IFIP/IEEE Int. Sym. Int. Net. Man.*, pp. 1–8, 2015.

[2] Ruairí de Fréin, C. Olariu, Y. Song, R. Brennan, P. McDonagh, A. Hava, C. Thorpe, J. Murphy, L. Murphy, and P. French, "Integration of QoS Metrics, Rules and Semantic Uplift for Advanced IPTV Monitoring," *Journal of Network and Systems Management*, pp. 1–36, 2014.

[3] R. Tibshirani, "Regression Shrinkage and Selection Via the Lasso," *J. Roy. Stat. Soc., Series B*, vol. 58, pp. 267–88, 1994.

[4] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[5] A. Hoerl and R. Kennard, "Ridge regression: Applications to nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 69–82, 1970.

[6] Ruairí de Fréin, "Effect of system load on video service metrics," in *IEEE Irish Sig. Sys. Conf. (ISSC)*, 2015, pp. 1–6.

[7] Ruairí de Fréin, K. Drakakis, and S. Rickard, "Portfolio diversification using subspace factorisations," in *42nd An. IEEE Conf. Inf. Sc. Sys. (CISS)*, 2008, pp. 1075–80.

[8] M. Zibulevsky and B. Pearlmutter, "Blind source separation by sparse decomposition in a signal dictionary," *Neur. Comp.*, vol. 13, no. 4, pp. 863–82, 2001.

[9] Ruairí de Fréin, "Ghostbusters: A parts-based NMF algorithm," in *IET Irish Sig. Sys. Conf. (ISSC)*, 2013, pp. 1–8.

[10] Ruairí de Fréin, "Formal concept analysis via atomic priming," in *Formal Concept Analysis*, P. Cellier, F. Distel, and B. Ganter, Eds., vol. 7880 of *LNCS*, pp. 92–108. Springer Berlin Heidelberg, 2013.

[11] S. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion detection using sequences of system calls," *J. Comp. Sec.*, vol. 6, pp. 151–80, 1998.

[12] Cisco, "Cisco visual networking index, Global IP Traffic Forecast, 2011-2016.," *white paper*, 2011.

[13] A. Begen, T. Akgul, and M. Baugher, "Watching video over the web Part 2: Applications, standardization, and open issues," *IEEE Internet Comp.*, vol. 15, no. 3, pp. 59–63, 2011.

[14] M. Zinkevich, A. Smola, and J. Langford, "Slow learners are fast," in *NIPS*, 2009, pp. 2331–9.

[15] D. Hsu, N. Karampatziakis, J. Langford, and A. Smola, "Parallel online learning," *CoRR*, vol. abs/1103.4204, 2011.

[16] J. Bogojeska, D. Lanyi, I. Giurgiu, G. Stark, and D. Wiesmann, "Classifying server behavior and predicting impact of modernization actions," in *Int. Conf. Net. and Serv. Man. (CNSM)*, 2013, pp. 59–66.

[17] M. Mirza, J. Sommers, P. Barford, and X. Zhu, "A Machine Learning Approach to TCP Throughput Prediction," *Networking, IEEE/ACM Trans.*, vol. 18, no. 4, pp. 1026–39, 2010.

[18] A. Andrzejak and L. Silva, "Using machine learning for non-intrusive modeling and prediction of software aging," in *IEEE Net. Op. Man. Symp. (NOMS)*, 2008, pp. 25–32.

[19] F. Zaman, S. Robitzsch, Zhuo W., J. Keeney, S. van der Meer, and G. Muntean, "A heuristic correlation algorithm for data reduction through noise detection in stream-based communication management systems," in *IEEE Net. Op. Man. Symp. (NOMS)*, 2014, pp. 1–8.

[20] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: Applications in R*, Springer Texts in Stats. 2013.