2016-05-05

# Source Separation Approach to Video Quality Prediction in Computer Networks

Ruairí de Fréin
*Dublin Institute of Technology*, ruairi.defrein@dit.ie



Recommended Citation
De Fréin, R. (2016) Source Separation Approach to Video Quality Prediction in Computer Networks. *IEEE Communications Letters* , vol. PP, no.99, pp.1-1, 2016. doi;10.1109/LCOMM.2016.2563418

# Source Separation Approach to Video Quality Prediction in Computer Networks

Ruairí de Fréin[†] [††]

[†]KTH - Royal Institute of Technology, Stockholm, Sweden

[††]Dublin Institute of Technology, Ireland

web: https://robustandscalable.wordpress.com

in: IEEE Communication Letters. See also BibTeX entry below.

BibTeX:

```
@article{rdefrein16Source,
author={Ruair\'{i} de Fr\'{e}in$^\dagger$ $^{\dagger\dagger}$},
journal={IEEE Communication Letters},
title={Source Separation Approach to Video Quality Prediction in Computer Networks},
year={2016},
volume={20},
number={7},
pages={1333--1336},
month={May},
keywords={Computational Modelling; Discrete Fourier Transform; Kernel; Load Modelling;
Measurement; Servers; Source Separation; Prediction Methods; Video-on-Demand};
ISSN={1089-7798},
doi={10.1109/LCOMM.2016.2563418},
url={http://ieeexplore.ieee.org/document/7465715/},}
```



document created on: March 14, 2018
created from file: IEEErdefreinComsLtr_v2.tex
cover page automatically created with CoverPage.sty
(available at your favourite CTAN mirror)

# Source Separation Approach to Video Quality Prediction in Computer Networks

Ruairí de Fréin

***Abstract*—Time-varying loads introduce errors in the estimated model parameters of service-level predictors in Computer Networks. A *load-adjusted* modification of a traditional *unadjusted* service-level predictor is contributed, based on Source Separation (SS). It mitigates these errors and improves service-quality predictions for Video-on-Demand (VoD) by $\approx .6$ to 2dB.**

***Index Terms*—Prediction Methods, Source Separation, Video-on-Demand.**

## I. Introduction

Cisco predicts that Internet traffic volumes of the order of tens of exabytes are imminent [1]. Given that 90% of the bits transmitted will be video-related it is crucial to be able to: (1) model the client's video quality; (2) predict future video-quality metrics so that corrective actions can be taken to ensure service-level agreements are met. The success of services like Netflix and Youtube underlines the number of potential engineering applications of a system that can correctly predict video-quality metrics. The application of Statistical Learning (SL) methods for Network Management is in its infancy. Previous works have examined the problem of video-quality prediction [2]. However system identification techniques were not applied to better understand the system under examination. Armed with recent Source Separation (SS) results –originally for separating a target speaker from a mixture of speakers– we pose the problem of video-quality prediction as a novel supervised deconvolution problem. The resulting predictor performs significantly better than previous approaches.

## II. Hierarchical Mixing Model

Source Separation (SS) of instantaneous signal mixtures has received much attention since the mid 1990s [3], [4]. SS explains a set of signals, $\mathbf{x}(i) = [x_1(i), \ldots, x_N(i)]^T$, by estimating the set of source signals, $\mathbf{s}(i) = [s_1(i), \ldots, s_M(i)]^T$, that gave rise to them using a model of the form

$$\mathbf{x}(i) = \mathbf{H}(i)\mathbf{s}(i) + \phi(i), \tag{1}$$

where $\mathbf{H}(i)$ is the mixing matrix. The vector $\phi(i)$ is often modelled as real-valued Gaussian noise and $i$ is a discrete time index. In Computer Networks, a client-server relationship –which is established when the client requests VoD from a server whose resources are potentially shared between many users– may be similarly expressed. If the number of active user VoD sessions, $a(i) \in \mathbb{Z}_+$, is integer-valued and non-negative, the mixing matrix and the source signals have the form,

$$\mathbf{H}(i) = [\mathbf{I}_N | a(i)\boldsymbol{\alpha}],\ \mathbf{s}(i) = [s_1(i), \ldots, s_{N-1}(i), 0, 1]^T. \tag{2}$$

In this setting, the $N$ mixtures, $\mathbf{x}(i)$, namely *features*, are sampled from the server's [L]UNIX kernel every $T$ seconds by calling a System Activity Report (SAR). The vector $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_N]^T$ determines the *relative* usage of the UNIX kernel's resources (features) per user request; $\mathbf{I}_N$ is an $N \times N$ identity matrix. For example, each active video client causes an $\alpha_1$ rate of context switching, an $\alpha_2$ additional memory usage, and finally, an $\alpha_N$ number of active TCP sockets. Each VoD request increments the TCP socket count by 1. The $N$th feature equals the number of active VoD requests $x_N(i) = a(i)$, e.g. $\alpha_N = 1$. Servers are not ideal; the deviation of a server from its ideal performance is represented by the source signal $s_n(i)$. Each source is the sum of, $a(i)$, perturbation signals, $\epsilon_n(i,k)$, for the $n$th feature. Special cases include the TCP socket count $s_N(i) = 0$ and the load $s_{N+1}(i) = 1$, but in general,

$$s_n(i) = \sum_{k=1}^{a(i)} \epsilon_n(i,k), \quad n = 1, \ldots, N-1. \tag{3}$$

The mixing matrix, $\mathbf{H}(i)$, states that each of the kernel features has a component which is due to the load on the system, $a(i)\alpha_n$, and also due to the non-ideal behaviour of the server, $s_n(i)$, e.g. the $n$th feature is $x_n(i) = a(i)\alpha_n + s_n(i) + \phi_n(i)$. This model is reasonable: (1) it states that usage of the server's kernel with respect to the $n$th feature increases when there are more clients requesting video; (2) it allows the volatility of the kernel features to increase when more users request video.

In this paper the client's VoD quality, $y(i)$, is indicated by the RTP Packet Count. There exist many other VoD quality metrics. Some notable metrics include, the Video Frame Rate, or the Audio Buffer Rate. The purpose of this paper not to perform an exhaustive evaluation of our ability to predict every possible metric, but to demonstrate (1) that the prediction of VoD quality metrics is affected by time-varying server load, and (2) that it is possible to increase the accuracy of VoD service-level predictions, RTP Packet Counts, by load-adjusting the statistical learning algorithm used to tune the predictor. The evaluation of different combinations of learning techniques and VoD service level metrics is the topic of on-going work. We use the RTP Packet Count as a representative VoD service level metric. It is obtained by extending the functionality of VLC Media player and sampled every $T = 1$s. The measured kernel signals, $\mathbf{x}(i)$ are instantaneously mixed

R. de Fréin is with Institiúid Teicneolaíochta Bhaile Átha Cliath (Dublin Institute of Technology) e-mail: rdefrein@gmail.com. Manuscript received December 7, 2015; revised February 7, 2016; accepted April 25, 2016. This work was supported by an Elevate Irish Research Council International Career Development Fellowship co-funded by Marie Curie Actions award "EOLAS": ELEVATEPD/2014/62.

by $\mathbf{w}(i)$, corrupted by iid noise $\varphi(i) \sim \mathcal{N}(0, \sigma^2)$, and the quality of the service received by the client is

$$y(i) = \mathbf{w}^T(i)\mathbf{x}(i) + \varphi(i). \quad (4)$$

**Alg. UnAdjusted (UA) Learning:** Previous work, in [2], solved a Maximum A Posteriori (MAP) problem to learn the coefficients $\mathbf{w}$ using $L$ pairs of training data, $\{y(i), \mathbf{x}(i)\}_{i=1}^{L}$,

$$\min_{\mathbf{w}} \sum_{i=1}^{L} (y(i) - \mathbf{x}^T(i)\mathbf{w})^2 + \lambda ||\mathbf{w}||^2. \quad (5)$$

To ease notation the matrices $\mathbf{y} = [y(1), \ldots, y(L)]^T \in \mathbb{R}^{L\times 1}$, and $\mathbf{X} = [\mathbf{x}^T(1), \ldots, \mathbf{x}^T(L)] \in \mathbb{R}^{L\times N}$ are introduced. The MAP solution can be written in the form of a pseudo-inverse:

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X} + \lambda \mathbf{I}_N)^{-1}\mathbf{X}^T\mathbf{y}. \quad (6)$$

They used a scalar regularizer, $\lambda$, to weight the relative importance of the terms related to the Gaussian likelihood, $\sum_{i=1}^{L}(y(i) - \mathbf{x}^T(i)\mathbf{w})^2$, and the Gaussian prior, $||\mathbf{w}||^2$, and assumed that the model parameters (in Eqn. 5) did not change with time. Once the model weights were learned, predictions of $\hat{y}(i+1)$ given $\mathbf{x}(i+1)$ were obtained via the inner product

$$\hat{y}(i+1) = \hat{\mathbf{w}}^T\mathbf{x}(i+1). \quad (7)$$

This benchmark predictor is *unadjusted* (UA) as it does not treat the load $a(i)$ as a factor that could affect system performance. We show that $\mathbf{x}(i)$ is more accurately modelled using Eqns. 1–3 because an arbitrary number of users can request VoD at any time; the parameters $\mathbf{w}$ are adjusted by load. Eqns. 1–3 formulate an instantaneous mixing model whose form is motivated: (1) the SAR command that generates the features, $\mathbf{x}(i)$, computes running sums/averages, etc. of the metrics over a configurable interval, and thus the role of delay (when it is $\ll T$) due to server and network delay is *integrated-out*. This summation action, coupled with (i) the desire to avoid modeling each feature and (ii) the fact that performance of the predictor is good, motivates the Gaussianity assumption. (2) We gave evidence in [5] that the load, $\alpha_n a(i)$, accounts for $> 50\%$ of the energy of a high proportion of the client-and-service metric traces, $\{y(i), \mathbf{x}(i)\}$ –this effect cannot be explained by Eqn. 5. A modification of the MAP solver in Eqn. 5 that describes the load's presence is derived. We derive a Load-Adjusted (LA) correction term for a new LA predictor, which is computed using Power Weighted Estimators (PWE).

## III. Load Removal Via SS

**Definition:** Let the discrete time and frequency indices be $i$ and $\omega$. We define the linear transform $T\{y(i)\}(\omega) : y(i) \mapsto Y(\omega)$ where $y(i) \in \mathbb{R}$ and $Y(\omega) \in \mathbb{C}$. We would like this transform to have two properties: (**p1**) the load, $a(i)$, and the perturbations, $s_n(i)$, are disjoint under the action of the transform and (**p2**) there exists a function, $\alpha_n(\omega) = F(T\{x_n(i)\}(\omega), T\{x_N(i)\}(\omega))$, that produces instantaneous estimates of the relative usage of each server feature relative to the server load, e.g. $\alpha_n(\omega)$.

**p1:** The load on the system, $a(i)$, has compact support in the frequency domain; it is sparsely represented when $T\{\cdot\}$ is the Discrete Fourier Transform (DFT). Using the DFT, the $n$th feature signal consists of two terms

$$T\{x_n(i)\}(\omega) = T\{a(i)\alpha_n\}(\omega) + T\{s_n(i)\}(\omega) = X_n(\omega) = \alpha_n A(\omega) + S_n(\omega). \quad (8)$$

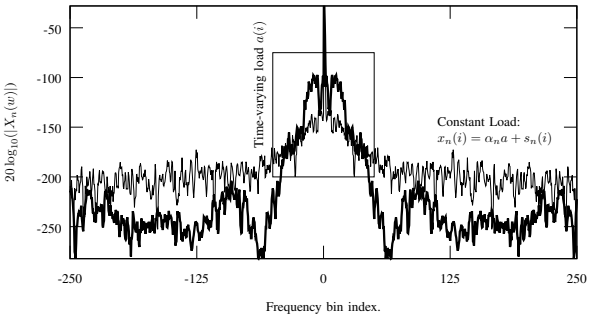


Fig. 1. Eqn. 9: Magnitude spectrogram of a time-varying FC load on the server vs. the magnitude spectrogram of one of the server features under a constant load conditions. The time-varying load dominates the constant load feature in approximately 80 low frequency bins (indicated by a rectangle).

**p1** is evaluated (for the DFT) by computing $|T\{a(i)\}(\omega)|$ of a time-varying load $a(i)$ and comparing it with the $n$th feature of the kernel, $|T\{x_n(i)\}(\omega)|$, which is under a constant load (cf. Fig. 1 where $a(i) = 25$). In Fig. 1 when the load is constant the perturbation signal, $s_n(i)$ has a broader support than when the load is time-varying. In practical terms, **p1** assumes that the frequency at which users request, or stop, watching video is lower than the frequencies of fluctuations in the kernel metrics. Frequency representations have been used by the SS community for their separation properties [6], [7]. We use the fact that a windowed DFT promotes approximate equality, and thus disjointness, in

$$A(\omega)S_n(\omega) \approx 0, \quad \forall n, \omega. \quad (9)$$

**Definition:** Approximate separation of the two components of $X_n(\omega)$ may be achieved by constructing the indicator function for the support of the load, $a(i)$,

$$M_{\theta_l}(\omega) = \begin{cases} 0, & \omega = 0, \text{ or } \omega \notin \theta_l \\ 1, & \omega \in \theta_l. \end{cases} \quad (10)$$

We use knowledge of the TCP socket count, $X_N(\omega) = A(\omega)$. We construct the set of frequencies $\theta_l = \{\omega || A(\omega)| > t\}$ using a threshold $t$, which corresponds to the cumulative sum of 90% of the load signal power (sorted from largest to smallest). We then use $\theta_l$ to estimate $\alpha_n(\omega), \forall n, \omega$.

**p2**: Instantaneous estimates of the relative usage of each server feature, $x_n(i)$, relative to the server load, $a(i) = x_N(i)$ can be determined by computing the ratio

$$\alpha_n(\omega) = \left| \frac{X_n(\omega)}{X_N(\omega)} \right|, \quad w \in \theta_l \quad (11)$$

in the frequency bins where the load dominates. An estimator that combines these estimates, $\alpha_n(\omega)$, is required.

A Maximum Likelihood Estimate (MLE) of $\alpha_n$, which we denote $\alpha_n^\star$, is used to motivate a Power Weighted Estimator (PWE) for $\alpha_n$, e.g. $\alpha_n^p$. The PWE estimates the relative usage of the UNIX kernel with respect to each of its features, $\alpha_n$,

relative to the load signal, $X_N(\omega) = A(\omega)$. The perturbation signal, in the frequency bins where $A(\omega)$ is dominant, $S_n(\omega)$, is treated as an interfering signal. It is modeled as complex iid white Gaussian noise, with zero mean and variance $\sigma_s^2$. Maximizing $L(\alpha_n)$ is equivalent to maximizing the likelihood $L_0$ of $\alpha_n$, e.g. $L_0(\alpha_n) := p(X_n(\omega), X_N(\omega), A(\omega)|\alpha_n)$.

$$L(\alpha_n) := -\sum_{\omega\in\theta_l} |X_n(\omega) - \alpha_n A(\omega)|^2. \tag{12}$$

Solving $\partial L/\partial\alpha_n = 0$ for $\alpha_n$ yields

$$\alpha_n^\star = \frac{\sum_{\omega\in\theta_l} \mathrm{Re}\{X_n(\omega)\bar{A}(\omega)\}}{\sum_{\omega\in\theta_l} |A(\omega)|^2}. \tag{13}$$

Given that we have a clean version of the load signal, $a(i)$, we use it. When $S_n(\omega) \approx 0$, in the set of frequency bins, $\theta_l$, and $C_{AA}(\omega) = |A(\omega)|^2$, then

$$\mathrm{Re}\{X_n(\omega)\bar{A}(\omega)\} \approx \alpha_n(\omega) C_{AA}(\omega). \tag{14}$$

The relative usage factor of the $n$th feature, $\alpha_n$, can be approximated by computing the estimate in the $\omega$-th frequency bin, $\alpha_n(\omega)$, and weighting the estimate by the load signal strength, which yields the Power Weighted Estimators, $\alpha_n^p$,

$$\alpha_n^p = \frac{\sum_{\omega\in\theta_l} C_{AA}(\omega)\alpha_n(\omega)}{\sum_{\omega\in\theta_l} C_{AA}(\omega)}. \tag{15}$$

PWEs allow us to focus in on the bins of $A(\omega)$, where the power of the signal is large, by weighting the contribution of each bin by $|A(\omega)|^2$, namely auto-weighting the instantaneous estimates. The compactness of the support of the load means that the MLE estimate (Eqn. 13) is corrupted by bins in which the load is not present. We also investigate the case where we cross-weight the estimates with $C_{AX}(\omega) = |A(\omega)X_n(\omega)|^2$, as cross-weighting generally reduces the risk of divisions by zero, which arise with auto-weighting. The MLE can take negative values because an increase in the load on the server may cause an increase or a decrease in the measure of some kernel resource. We correct the PWE sign as follows $\alpha_n^p = \mathrm{sgn}(\alpha_n^\star)\alpha_n^p$. The estimators $\alpha_y(\omega)$ and $\alpha_y^p$ follow directly by substituting in $Y(\omega)$ for $X_n(\omega)$ in Eqn. 15. Computation of the PWE costs $25F + 7$ FLOPS per estimate, where $F$ is the number of elements in $\theta_l$, which is remarkably cheap.

## IV. Load-Adjusted Estimation and Prediction

**Definition:** The load-adjusted feature, $\hat{\mathbf{x}}_n$, and service metric, $\hat{y}$, are reconstructed by *separating-out* the load

$$\hat{x}_n(i) = x_n(i) - \alpha_n^p a(i), \quad \hat{y}(i) = y(i) - \alpha_y^p a(i). \tag{16}$$

Let $\mathcal{A}$ denote the set of values the load on the server can assume. Let $\mathcal{K}_k = \{j|a(j) = k, j = 1,\ldots,L\}$ denote the set of time indices where the load is equal to $k$ active users. Let $\mathbf{y}_k = [\ldots|y(j)|\ldots]_{j\in\mathcal{K}_k} \in \mathbb{R}^{|\mathcal{K}|\times 1}$ and $\mathbf{X}_k = [\ldots|\mathbf{x}(j)^T|\ldots]_{j\in\mathcal{K}_k} \in \mathbb{R}^{|\mathcal{K}|\times N}$ denote the dependent and independent variables corresponding to the case where the load is $a(i) = k$. When $a(i) = k$ we define load-adjusted variables:

$$\hat{\mathbf{X}}_k = ([\mathbf{I}_N|k\,(\boldsymbol{\alpha} - \boldsymbol{\alpha}_n^p)]\mathbf{S}_k^T)^T, \quad \hat{\mathbf{y}}_k = \mathbf{y}_k - \alpha_y^p k. \tag{17}$$

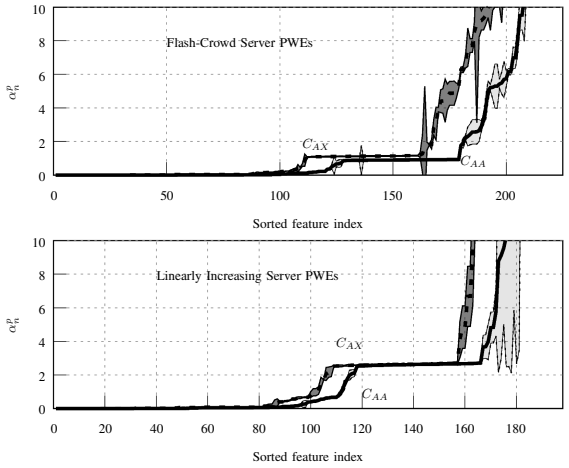


Fig. 2. The estimated feature usage estimates $\alpha_n^p$, estimated using the auto-weighting function, $C_{AA}$, and cross-weighting function, $C_{AX}$, for the Flash-Crowd (top) and the Linearly Increasing (bottom) traces. These estimates are sorted from smallest to largest for illustration purposes. Cross power weighted estimates, $C_{AX}$, give larger estimates with a lower variance, because multiplying $X_n(\omega)$ by $A(\omega)$ causes large values of $X_n(\omega)$ to compensate for $A(\omega)$ when $A(\omega)$ is small, which improves the numerical stability of the cross-weighted estimator.

**Alg. Load-Adjusted (LA) Learning:** A LA estimate of $\hat{\mathbf{w}}_k$, is obtained by solving a MAP problem for each $k \in \mathcal{A}$

$$\min_{\hat{\mathbf{w}}_k} \left(\hat{\mathbf{y}}_k - \hat{\mathbf{X}}_k\hat{\mathbf{w}}_k\right)^T \left(\hat{\mathbf{y}}_k - \hat{\mathbf{X}}_k\hat{\mathbf{w}}_k\right) + \lambda\hat{\mathbf{w}}_k^T\hat{\mathbf{w}}_k \tag{18}$$

We estimate the parameters

$$\hat{\mathbf{w}}_k = (\hat{\mathbf{X}}_k^T\hat{\mathbf{X}}_k + \lambda\mathbf{I}_N)^{-1}\hat{\mathbf{X}}_k^T\hat{\mathbf{y}}_k \tag{19}$$

for each load value using historical data. In summary, LA learning comprises of: (1) Computing the set $\theta_l$ (cf. Eqn. 10); (2) Estimating the PWEs $\alpha_n^p, \forall n$ (cf. Eqn. 15); (3) load adjusting and constructing $\hat{\mathbf{X}}_k$ and $\hat{\mathbf{y}}_k$ using (cf. Eqn.17); and finally, (4) estimating the weights $\hat{\mathbf{w}}_k, \forall k$ (cf. Eqn. 19).

**LA Predictor:** If a new vector of features $\mathbf{x}(i+1)$ is drawn on the server machine and we desire a prediction of $y(i+1)$: (1) we select the appropriate set of parameters $\hat{\mathbf{w}}_j$ using the TCP socket count $j = x_N(i)$; (2) we compute $\hat{y}(i+1) = \hat{\mathbf{w}}_j^T\mathbf{x}(i+1)$; (3) we re-adjust for the load

$$\bar{y}(i+1) = \hat{y}(i+1) + \alpha_y^p j. \tag{20}$$

## V. Results

We use the test scenarios, "Flash-Crowd" (FC) and "Linearly Increasing" (LI), described in [2] to evaluate our PWEs and the improvement in Signal-to-Noise-Ratio (SNR), obtained by using LA versus UA prediction in a RTP packet count prediction task. The server responds to VoD requests from a target client. RTP Packet Counts, $y(i)$, and the features $\mathbf{x}(i)$, are collected on this client's machine and the server respectively. A load-balancer generates interfering user requests on the server for ≈5hrs, according to a Poisson distribution, whose mean is modulated by a LI and then a FC process [2].

We plot the mean of the cross- and auto-weighted PWEs in Fig. 2 for the LI and FC traces. We illustrate PWEs in the

range $0 \leq \alpha_n^p \leq 10$ to highlight when the load is present. The line thickness indicates the standard deviation of the PWEs. PWEs for the FC and LI traces are obtained using a DFT with 2000s long windows, every second, for 13000s for each server feature. Fig. 2 demonstrates that (1) the load is present in approximately 50% of the server features using the cross-weighted PWE, e.g. the relative usage estimate is greater than zero, $\alpha_n^p > 0$, for both test traces, which supports the case for a LA learning and prediction model. (2) The variance of the estimates is generally small irrespective of the value of the load. For example, the standard deviation of the $\alpha_n^p$ estimates is small up to sorted index 160 for both traces. The auto-weighted PWE of $\alpha_n^p$ is generally smaller than the cross-weighted PWE. This is because when the power of the load signal is small in a frequency bin, this frequency bin reduces the overall estimate of the PWE. This effect contributes to the large variance in some of the larger feature usage estimates because the estimator is more sensitive to spurious instantaneous relative usage estimates. Cross-weighting tends to desensitize the PWEs to spurious instantaneous relative usage estimates. Their variances are generally smaller, but cross-weighted PWEs are also larger. We use the cross-weighted PWEs in our prediction experiments as they produce PWE estimates which have a smaller variance. (3) It is interesting to note that both forms of PWE produce estimates which are approximately zero, $\alpha_n^p \approx 0$ for some features. This demonstrates that the PWEs are able to identify when the load is not present in the associated feature, and no load-adjustment is performed. Recall that the amount of adjustment applied to the features and the RTP Packet Count is defined as $\hat{x}_n(i) = x_n(i) - \alpha_n^p a(i)$ and $\hat{y}(i) = y(i) - \alpha_y^p a(i)$. If PWEs are zero for all features, $\alpha_n^p \approx 0$, then LA learning reduces to traditional UA learning, which is crucially important.

We evaluated the proposed LA approach and the benchmark UA method described in [2] by performing predictions under different load values, $k$, for the FC and LI traces. The traces were partitioned into 5:1 splits of training:test data, 1000 times for each value of the load $k$, e.g. $20 \leq k \leq 120$ and $20 \leq k \leq 70$ user requests for the FC and LI traces respectively. We used the cross-weighted PWE in the LA approach as its variance is smaller in Fig. 2. The features and service level metrics were then load-adjusted. We performed cross-validation for $\lambda$ independently for each load value and algorithm variant. Fig. 3 plots the average gain in SNR achieved by LA vs UA prediction for each load $k$. These averages are computed by predicting $\frac{1}{5}$th of the values in the FC and LI traces 1000 times. The average improvement in accuracy, over all loads, is 2dB for the FC and .6dB for the LI trace. The improvement is greater for the FC because this trace contains higher loads.

The number of samples drawn for the Flash-Crowd trace is non-uniformly distributed over the different load values. The number of samples per load value is approximately uniform for the linearly increasing case. (1) Note that the improvement in prediction accuracy for the linearly increasing trace increases as the load increases (this increase is not as smooth for the Flash-Crowd). This is because the size of the load correction term in $\hat{x}_n(i) = x_n(i) - \alpha_n^p a(i)$ and $\hat{y}(i) = y(i) - \alpha_y^p a(i)$ increases as the load increases. (2) For some load values,

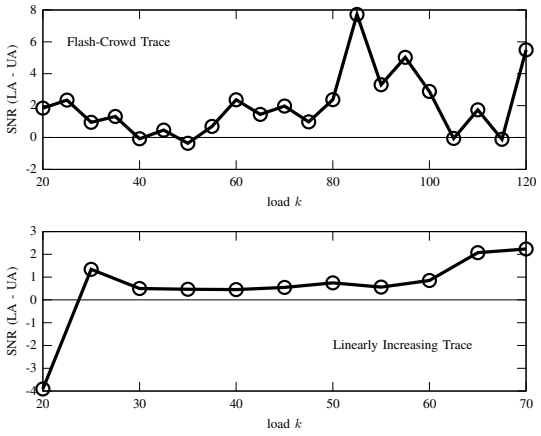


Fig. 3. The Signal-to-Noise Ratio improvement achieved by load-adjusted prediction over unadjusted prediction for the Flash-Crowd (top figure) and the Linearly Increasing (bottom figure) load traces is illustrated. A 2dB and .6dB improvement is achieved on average over all predictions for the Flash-Crowd and Linearly Increasing load respectively. Improvement is achieved in all but one load value for each trace. We attribute this failure to improve the SNR via load-adjustment to over estimates of $\alpha_n^p$.

the improvement in SNR achieved by LA learning is either approximately zero, or negative. We have observed that the PWEs exhibit a large variance for some of the features. We posit that this PWE variance may cause the LA to over adjust, and thus subtract too much load from some feature. The problem of dealing with these problem features is the subject of on-going work. In general, LA learning improves the SNR of the predictions. (3) We conclude by stating that LA learning is fast; each PWE costs $25F+7$ FLOPS, the FFT costs $O(L \log L)$ FLOPS and finally the estimator $\hat{\mathbf{w}}_k$ has an asymptotic complexity of $O(N^2|\mathcal{K}|)$. LA learning may often be faster than UA learning as UA learning has an asymptotic complexity of $O(N^2 L)$ and LA learning is easily parallelized.

## VI. Conclusion

A predictor that predicts a client's VoD RTP Packet count in the presence of a time-varying load is presented. Three main results emerge that warrant reporting: (1) the relative usage of the server features and service level metric can be estimated using cross-weighted PWEs; (2) the LA predictor is up to 2dB more accurate than the UA predictor; (3) this gain comes at a low computational cost.

## References

[1] "Cisco visual networking index, Global IP Traffic Forecast, 2011-2016."
[2] 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. Symp. on Int. Net. Man., pp. 1-8, 2015
[3] R. Comon, "Independent Component Analysis: A new concept?", *Sig. Proc.*, vol. 36, no. 3, pp. 287-314, 1994.
[4] S.C. Douglas, A. Cichocki and S. Amari, "Bias removal technique for blind source separation with noisy measurements", *Elec. Let.*, vol. 34, no.14, pp. 1379–80, 1998.
[5] Ruairí de Fréin, "Effect of System Load on Video Service Metrics", *Irish Signals and Systems Conference*, pp. 1-6, 2015.
[6] Ruairí de Fréin and S.T. Rickard, "The Synchronized Short-Time-Fourier-Transform: Properties and Definitions for Multichannel Source Separation", *Sig. Proc., IEEE Trans.*, col. 59, no. 1, pp. 91-103, 2011.
[7] O. Yilmaz and S.T. Rickard, "Blind separation of speech mixtures via time-frequency masking", *Sig. Proc., IEEE Trans.*, vol. 52, no.7, pp. 1830–47, 2004.