Workshop IA en Sistemas Ciber-Fisicos
Estimating Conformance to Service Level Agreements (SLAs) using Machine Learning

Rafael Pasquini
Universidade Federal de Uberlândia - UFU
`rafael.pasquini@ufu.br`

Christian Esteve Rothenberg
Universidade Estadual de Campinas - Unicamp
`chesteve@dca.fee.unicamp.br`

November 11, 2024

Overview

Telecom and Internet services are vital to our daily life. Such services involve large and complex software systems that increasingly run on general-purpose platforms and operating systems. For the proper function of those services, a service provider performs real-time service assurance, which ensures the service-level agreement (SLA) between the provider and a customer. In this project, we study the quality of service of a video-on-demand (VoD) service, and the SLA towards a customer is expressed as the minimum number of video frames per second on the customer terminal. The project focuses on a critical part of service assurance, namely, on the capability of a provider to estimate the service quality based on measurements on the provider's infrastructure. The estimation uses machine-learning techniques, specifically regression and classification.

Background

SLAs remount back to late 1980s, when fixed line telecom operators needed to describe their service on contracts with their corporate customers. Nowadays, SLAs are spread in practically all industries and markets. Specifically for the telecom market, a SLA enables the quality of service (QoS) benchmarking. Essentially, SLAs are output-based, i.e., the agreement is based on the service level metrics as received by the customer. Therefore, the service providers can demonstrate their value by organizing themselves with capability and knowledge to deliver the service-level metrics required, perhaps in an innovative way, such as the machine learning approach we take in this project.

Given the telecom cloud scenario, in which applications are moved from dedicated hardware into the cloud, operators are required to deliver the same or even higher levels of service than classical installations. SLAs for telecom cloud focus on characteristics of the data center and more recently include characteristics of the network to support end-to-end SLAs. We advocate that a key component for a SLA conformance system is an accurate prediction model for real-time service-level metrics developed using machine learning. As defined by the pioneer in machine learning, Arthur Samuel:

"Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed"

Most of machine learning problems fall in two main classes - Supervised and Unsupervised Learning, but there is also a third class which is defined as Semi-Supervised Learning, and other classes like reinforcement learning, which are not introduced in this secttion. In supervised learning, we are given a data set and already know what our correct output should look like, having the idea that there is a relationship between the input and the output. In essence, supervised learning works on labeled data.

For each observation of the system-of-interest measurement(s) $x_i, i = 1, ..., n$ there is an associated response (label) measurement y_i. In supervised learning we wish to fit a model that relates the label y_i to the observed measurement(s) (variables/features) x_i, with the aim of accurately estimating the response for future observations (prediction) or better understanding the relationship between the response and the measurement(s) (inference) [1]. The supervised learning can be split into two problems:

- Regression - In a regression problem, we are trying to predict results within a continuous output, meaning that we are trying to map input variables to some continuous function;
- Classification - In a classification problem, we are instead trying to predict results in a discrete output. In other words, we are trying to map input variables into discrete categories.

In contrast to supervised learning, unsupervised learning allows us to approach problems with little or no idea what our results should look like. We can derive structure from data where we do not necessarily know the effect of the variables. In essence, unsupervised learning works on unlabeled data.

Unsupervised learning describes the somewhat more challenging situation in which for every sample $i = 1, ..., n$, we observe a vector of measurements x_i but no associated response y_i. One learning tool that we may use in this setting is cluster analysis, or clustering. The goal of cluster analysis is to ascertain, on the basis of $x_1, ..., x_n$, whether the observations fall into relatively distinct groups.

Many problems fall naturally into the supervised or unsupervised learning paradigms. However, sometimes the question of whether an analysis should be considered supervised or unsupervised is less clear-cut. For instance, suppose that we have a set of n observations. For m of the observations, where $m < n$, we have both measurements and a response (label). For the remaining $n - m$ observations, we have measurements but no labels. Such a scenario can arise if the measurements can be obtained relatively cheaply but the corresponding labels are much more expensive to collect. We refer to this setting as a semi-supervised learning problem. In this setting, we wish to use a statistical learning method that can incorporate the m observations for which labels are available as well as the $n - m$ observations for which they are not.

Project Objective and Approach

The objective of this project is to investigate SLA conformance of a video-on-demand (VoD) service using a service metric Y on the client side and device statistics X on the server side. Figure 1 gives the basic configuration of the system we consider [2]. It consists of a client machine that is connected to a server via a network. The client accesses the VoD service that runs on the server. In this setting, device statistics refer to operating-system metrics on server side, while service metrics Y refer to statistics on the client side. Table 1 describes the metrics X and Y used in the project.

Using machine-learning techniques, the problem is to find a function (i.e., a learning model) $M : X \to \hat{Y}$, such that $\hat{Y}$ closely approximates Y for a given X. If Y is a real number, the problem is referred to as a regression problem; if Y is a label, the problem is called a classification problem. This project considers both of these problems. To estimate M, measurement pairs (or observations) of the form (X,Y) are needed. A set of measurement pairs is also called a trace, and we use in this project a trace of 3600 observations, collected once every second from running the system over the course of an hour (periodic trace in[3]).

For this project, we say that the VoD service conforms to the SLA at a particular time, if $Y \geq 18 frames/second$ at that time; otherwise, we say that the VoD service violates the SLA at that time.

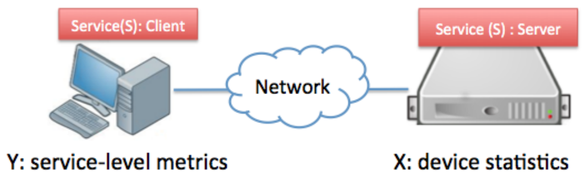

Figure 1: System configuration for estimating service metrics.

Field ID	Description
all_..idle	Percentage of idle CPU time (%)
X..memused	Percentage of used memory (%)
proc.s	Rate of process creation
cswch.s	Rate of context switching
file.nr	Number of used file handles
sum_intr.s	Rate of interrupts
ldavg.1	Load average for the last minute
tcpsck	Number of used TCP sockets
pgfree.s	Rate of freeing pages

(a) Device statistics X

Field ID	Description
DispFrames	Video frame rate

(b) Service metric Y

Table 1: Device statistics X and service metric Y. Rates are given in units per second.

Obviously, Figure 1 is a simplification of a practical system. It does not take into account network statistics and client device statistics. Also, the service platform is generally much more complex.

Project tasks

The project is composed of **four** tasks.

TBD - VERIFICAR numero de tasks!

Task I - Data Exploration

The objective of this task is to familiarize you with Python-based data exploration tools. We use the above mentioned trace with 3600 observations as the data set for the following computations.

1. Compute the following statistics for each component of X and Y: mean, maximum, minimum, 25th percentile, 90th percentile, and standard deviation.
2. Compute the following quantities:
   (a) the number of observations with memory usage larger than 80%;
   (b) the average number of used TCP sockets for observations with more than 18000 interrupts/sec;
   (c) the minimum memory utilization for observations with CPU idle time lower than 20%.
3. Produce the following plots:
   (a) Time series of percentage of idle CPU and of used memory (both in a single plot);
   (b) Kernel Density Estimate (KDE) plots of idle CPU and of used memory.

Resources:

- “Jupyter Notebook.” https://jupyter.org/
- “scikit-learn.” https://scikit-learn.org/

Task II - Estimating Service Metrics from Device Statistics

The objective of this task is to estimate the frame rate of a VoD service from the device statistics of a VoD server (see Figure 1). Our approach is to gather observations from the system and apply linear regression on this data to estimate the service metric Y from device statistics X. For this task, we use the trace with 3600 observations described above. The device statistics are described by the feature set in Table 1(a), and the service metric is shown in Table 1(b).

Your task is to compute (i.e., to train) a model M that accurately maps device statistics onto service metrics.

You train and test your model M with the so-called validation-set technique. This technique entails that you split the set of observations into two parts: the *training set* for computing the model M and the *test set* for evaluating the accuracy of M. From the complete set of observations, you select uniformly at random, 70% of the observations (2520 observations) to form the training set and then assign the remaining 30% (1080 observations) to the test set.

1. **Evaluate the Accuracy of Service Metric Estimation**

   (a) Model Training - use linear regression to train a model M with the training set. Provide the coefficients $(\Theta_1, ..., \Theta_9)$ of your model M.

   (b) Accuracy of Model M - compute the *estimation error* of M over the test set. We define the estimation error as the *Normalized Mean Absolute Error* $(NMAE) = \frac{1}{\bar{y}}(\frac{1}{m}\sum_{i=1}^{m}|y_i - \hat{y}_i|)$, whereby $\hat{y}_i$ is the model estimation for the measured service metric y_i, and $\bar{y}$ is the average of the observations y_i of the test set, which is of size $m = 1080$. [2]. We consider an estimation accurate if $NMAE < 15\%$.

   (c) Produce a time series plot that shows the measurements and the model estimations for the Video Frame Rate values in the test set (see example of such a plot in Figure 4(a) of [2]).

   (d) Produce a kernel density estimate (KDE) plot for the Video Frame Rate values in the test set (see Figure 4(c) of [2]).

   (e) Produce a KDE plot for the absolute prediction errors $|y_i - \hat{y}_i|$ in the test set.

   (f) Discuss the accuracy of estimating the Video Frame Rate in this way.

2. **Study the Relationship between Estimation Accuracy, the Size of the Training Set, and the Model Training Time**

   Training sets Preparation - from the above training set with 2520 observations, create five training sets by selecting uniformly at random, 50 observations, 500 observations, 1000 observations, 1500 observations, and 2520 observations (which is the originally proposed size for the trainning set).

   (a) Model Training - use the same linear regression method as above to train five models $M_1, ..., M_5$, one for each training set. Provide a table with the coefficients of these models.

   (b) Training Time of the Models - on your computer, measure the execution time (in milliseconds) to train each of the five models.

   (c) Accuracy of Models - compute the $NMAE$ for each of the five models for the test set.

   (d) Produce a plot that shows both the $NMAE$ and Training Time for all models. Use two y-axes, one on the left side and one on the right side, to include both curves in a single plot.

   (e) Discuss the relationship between the number of observations in the training set and the accuracy of the model estimations.

(f) Discuss the relationship between the number of observations in the training set and the training time of a model.

Resources:

If you are not familiar with the basic concepts of machine learning or the method of linear regression, we recommend the following short videos by Andrew Ng from Stanford University.

- What is Machine Learning?

  `https://www.youtube.com/watch?v=e0_bdQV9BVE&list=PLiPvV5TNogxIS4bHQVW4pMkj4CHA8COdX&index=16`

- Supervised Learning

  `https://www.youtube.com/watch?v=1xkTVdnxPMk&list=PLiPvV5TNogxIS4bHQVW4pMkj4CHA8COdX&index=17`

- Model Representation

  `https://www.youtube.com/watch?v=4YRN8UDYL7o&list=PLiPvV5TNogxIS4bHQVW4pMkj4CHA8COdX&index=19`

- Cost Function

  `https://www.youtube.com/watch?v=0oIjN6LgT5Y&list=PLiPvV5TNogxIS4bHQVW4pMkj4CHA8COdX&index=20`

- Multiple Features

  `https://www.youtube.com/watch?v=B6x96RKSKvA&list=PLiPvV5TNogxIS4bHQVW4pMkj4CHA8COdX&index=33`

Task III - Estimating SLA Conformance and Violation from Device Statistics

The objective for this task is to build a binary classifier function that estimates whether the VoD service conforms to the given SLA (see above) for specific device statistics X, or whether the service violates the SLA for a specific value of X.

As in the previous task, build a test set and a training set from the trace data, containing 70% of the observations and 30% of the observations, respectively.

1. **Evaluate the Accuracy of SLA Estimation, Training time, and their dependence on Training Set Size**

   Training sets Preparation - from the above training set with 2520 observations, create five training sets by selecting uniformly at random 50 observations, 500 observations, 1000 observations, 1500 observations, and 2520 observations (which is the originally proposed size for the trainning set).

   (a) Classifier Training - use logistic regression to train five classifiers $C_1, ..., C_5$, one for each training set. Make a table showing the coefficients (Θ) of the classifiers;

   (b) Time to Train the Classifiers $C_1, ..., C_5$ - measure the execution time in milliseconds to train each classifier;

   (c) Accuracy of the Classifiers $C_1, ..., C_5$ - Compute the classification error (ERR) on the test set for each classifier. We define the classification error as $ERR = 1 - \frac{TP+TN}{m}$, whereby TP is the fraction of observations classified as true positives, TN is the fraction of observations classified as true negatives, and m is the number of observations in the test set. A true positive is an observation that is correctly classified by the classifier as conforming to the SLA; a true negative is an observation that is correctly classified by the classifier as violating the SLA. Consider a classifier as accurate when $ERR < 15\%$;

(d) Produce a plot that shows both the Classification Error (ERR) on the test set and Training Time for each of the five classifiers $C_1, ..., C_5$. Use two y-axes to include both curves in a single plot;

(e) Using the test set produce a time series plot that shows on the y-axis the observed video frame rate. For each point in the plot indicate the classification obtained with the most accurate classifier C_i. Use different symbols (or colors) for a correctly classified and a incorrectly classified observation.

(f) Using the above plot discuss how the estimation accuracy relates to the evolution of the video frame rate over time;

(g) Discuss the relationship between the number of observations in a training set and the estimation accuracy for $C_1, ..., C_5$;

(h) Discuss the relationship between the number of observations in a training set and the training time for $C_1, ..., C_5$.

Resources:

- Classification

  `https://www.youtube.com/watch?v=f06WBsQVn4M&list=PLiPvV5TNogxIS4bHQVW4pMkj4CHA8COdX&index=47`

- Hypothesis Representation

  `https://www.youtube.com/watch?v=SlaVCiibLyO&list=PLiPvV5TNogxIS4bHQVW4pMkj4CHA8COdX&index=48`

- Decision Boundary

  `https://www.youtube.com/watch?v=VaRH-0-0cDU&list=PLiPvV5TNogxIS4bHQVW4pMkj4CHA8COdX&index=49`

- Cost Function

  `https://www.youtube.com/watch?v=hiUjnnNXkyU&list=PLiPvV5TNogxIS4bHQVW4pMkj4CHA8COdX&index=50`

Task IV - 5G Datasets Challenge

Various methodologies can be explored to maximize the capabilities of 5G. The use of video streaming, such as YouTube, can serve as a powerful tool to understand 5G network performance in real-time multimedia content delivery. By capturing datasets that include information on streaming service usage and the performance metrics of 5G networks in operation, researchers can analyze service quality, link metrics, among other factors affecting user experience.

For experimental evaluation, test scenarios can be created in which devices connect to 5G networks to measure latency, data transfer speed, and connection reliability. These data can be used to develop machine learning models that predict network performance under different conditions.

Combining real-world data with machine learning techniques opens doors to valuable insights and the development of solutions that further enhance the quality and efficiency of 5G networks across various applications.

Different to the previous tasks I to III, this task is a challenge-like type of open-ended activity, i.e. there is not a single correct answer, solutions may develop in several ways. Different approaches and dimensions of the problem and solution spaces can be explored.

The grade of this task will be based on the complexity of the formulated problem of the selected activities and the deepness and correctness of the proposed solutions.

This task was adapted from SBRC 2023 Hackathon Smartness / 5G Dataset Challenge.[1] Below you can find additional information (in Portuguese, but we believe you know ChatGPT or alternative ways to

[1] `https://sbrc.sbc.org.br/2023/hackathon/`

translate text, if needed) about a candidate Jupyter Notebook environment, auxiliar Python scripts and the relevant datasets for each suggested activity.

- Datasets

  `https://github.com/intrig-unicamp/hackathon5G/tree/main/datasets`

- Execution Environments (Google Colab, Jupyter Notebook, Mamba, Conda)

  `https://github.com/intrig-unicamp/hackathon5G/blob/main/README.md`

- Auxiliary Python scripts

  `https://github.com/intrig-unicamp/hackathon5G/tree/main/scripts`

For further reading and understanding of the dataset collection process and performance benchmarks, refer to the following material related to our work "YouTube goes 5G".

- Main scientific publication:
  Raza Ul Mustafa, Chadi Barakat, Christian Esteve Rothenberg. YouTube goes 5G: QoE Benchmarking and ML-based Stall Prediction. IEEE Wireless Communications and Networking Conference (WCNC), Apr 2024. `https://inria.hal.science/hal-04400816`

- IEEE Dataport

  `https://ieee-dataport.org/documents/youtube-goes-5g-benchmarking-youtube-4g-vs-5g`

- Documentation:

  `https://ieee-dataport.s3.amazonaws.com/docs/36297/YouTube_5g_VF.pdf`

- Github dataset repository

  `https://github.com/razaulmustafa852/youtubegoes5g`

- Related scientific publication: EFFECTOR: DASH QoE and QoS Evaluation Framework For EnCrypTed videO tRaffic R. U. Mustafa, M. T. Islam, C. Rothenberg and P. H. Gomes, "EFFECTOR: DASH QoE and QoS Evaluation Framework For EnCrypTed videO tRaffic," NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium, Miami, FL, USA, 2023

  `https://github.com/razaulmustafa852/youtubegoes5g`

- EFFECTOR GitHub: `https://github.com/smartness2030/EFFECTOR`

Activity A: Signal Quality Prediction

The objective of this activity is to infer signal quality based on the attributes provided in the g-nettrack dataset. Tthe provided data can be combined with information from other datasets to complement and enrich the available data. For example, the g-nettrack dataset can be combined with the Mosaico dataset to provide information on nearby antennas, such as location, azimuth, antenna density, frequency, and technology.

You can consider simpler statistical models and/or more elaborated machine learning techniques, such as univariate and multivariate prediction models, to infer the values of a chosen signal quality indicator, such as QUAL, CQI, or SNNR.

Activity B: Mobility Type Prediction

This activity involves using unsupervised machine learning methods to predict the type of mobility (pedestrian, vehicle, metro/train) of a mobile device based on the unlabeled location data provided in the g-nettrack dataset. This task can be approached in various ways, such as a clustering-based classification problem (unsupervised classification), where the machine learning model should classify each record in the dataset into one of the possible mobility classes.

Activity C: Video Transmission Quality of Experience (QoE) Prediction

The QoE prediction activity for adaptive video transmission (YouTube) involves using machine learning techniques to predict the QoE of video streaming on mobile devices. The youtube-qoe dataset provides video transmission metrics. The objective is to create two prediction models that take into account information such as the mobile device's coordinates, network characteristics, the technology used, and other variables, to estimate the QoE of the video transmission. This can improve user experience by ensuring that video streaming is performed with the best possible quality, considering network conditions and the user's location.

The developed models should predict the QoE of the transmission based on distinct inputs:

- **Model 1:** predicts QoE based on location data, obtained from the g-nettrack dataset correlated with the Mosaico dataset to identify nearby antennas.
- **Model 2:** predicts QoE based on network metrics (such as signal quality and technology), obtained from the g-nettrack dataset.

In summary, Model 1 should only consider the availability of nearby antennas (such as technology, power, and other factors of the base stations that could influence optimal network quality in a given area) to infer QoE. In contrast, Model 2 considers only the network conditions (network metrics).

In addition to model development, teams should create a function to define QoE. A simple alternative could relate QoE directly to video resolution; however, it's clear that high-resolution video with frequent buffering does not provide good QoE. Conversely, a completely buffer-free, fluid video may not have good QoE if streamed at very low resolution.

Activity D: Infer the Connected Cell/Base Station of a Mobile Device

The proposed activity is to infer which Cell/Base Station (BS) the mobile device is connected to, based on the device's coordinates, using the g-nettrack dataset correlated with the Mosaico dataset to identify nearby antennas. This involves using data processing and machine learning techniques to analyze location data and nearby antennas, as well as basic knowledge of mobile network physical architecture deployment.

It should be noted that antennas on a base station can be directional (Azimuth ¿ 0) or omnidirectional (Azimuth = 0), meaning they can either direct the transmitted signal in a specific direction or transmit the signal in all directions around them, respectively.

References

[1] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*, vol. 112. Springer, 2013.

[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," tech. rep., 2014. `http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-152637`.

[3] R. Yanggratoke, J. Ahmed, J. Ardelius, C. Flinta, A. Johnsson, D. Gillblad, and R. Stadler, "Linux kernel statistics from a video server and service metrics from a video client.," 2014. Distributed by Machine learning data set repository [MLData.org]. `http://mldata.org/repository/data/viewslug/realm-im2015-vod-traces`.