Final Report for cs229:

Machine Learning for Pre-emptive Identification of Performance Problems in UNIX Servers

Helen Cunningham

Abstract. The goal of this work is to use machine learning to understand server performance in a SunRay™ client-server network. Preliminary work shows differences between the data emitted by servers who are performing well versus data emitted by those that are performing poorly. These differences emerge early in an operational day, and so give an opportunity to pre-emptively identify servers that will be operating in an abnormal manner later in the day. The project reduces the size of the dataset by computing a covariance matrix between the different system variables and doing supervised and unsupervised learning in this space (instead of the time domain). K-means clustering found 7 variable groups corresponding to hardware and OS subsystems. Support Vector Machine algorithm was used to classify correlation output and raw data.

Introduction

Client-server networks require high levels of performance assurance to guarantee fast and uniform response time for all clients, applications, threads, and network processes operating at any given time. In the SunRay™ thin client system, this is particularly important as nearly all application processing -- including graphics rendering -- occurs on the server and is communicated to the client in UDP packet streams. In this setting, performance decrements manifest in subtle ways, often appearing suddenly and disappearing just as quickly.

Preliminary studies of memory, floating point operations, and network variables have shown that no single variable effectively captures the state of a system for the purposes of performance evaluation, and it is apparent that an effective characterization of system performance must use a multivariate model of system state.

The purpose of this project is to observe and model server performance and to devise measures that can be used to classify "healthy" vs. "suspect" servers.

A Sun internal application samples and stores data from the Solaris "kernel" at 10-minute intervals. The 10-minute samples are corrected for timestamp irregularities, differentiated where cummulative, then interpolated and smoothed.

I. Building a Model of Normal Behavior

A sample of Unix variables for 15 consecutive Wednesdays for 44 server machines was obtained. After procedures detailed in "Milestone" Report (but omitted here), we ended up with 445 server-day datasets that we knew to be Normal, and 78 that we knew to be Atypical.

Covariance or multiple correlation reduces the time series to a single number that relates each variable to each of the other variables, and so realizes a large reduction in size of the dataset.

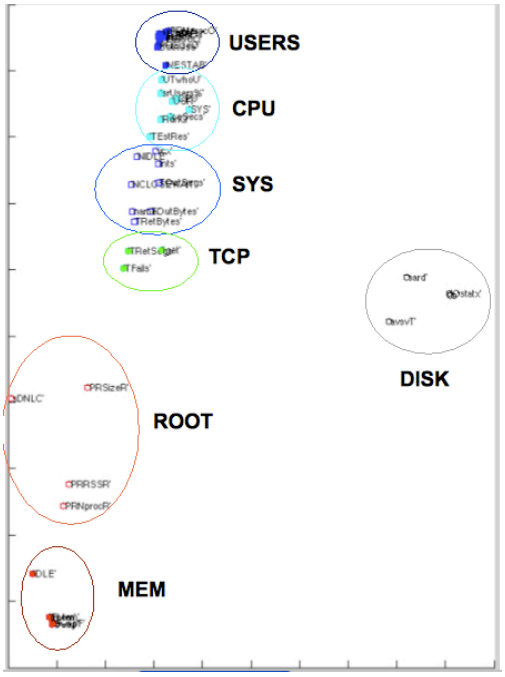
A correlation matrix was formed for each of the 445 'normal'datasets, and a randomly selected subset of 50 were inspected visually by running the matrix through MATLAB's "surface" function which maps a value to a color. An example surface "heat map" is shown in Figure 2, where data from several similar-looking datasets have been aggregated. Dark red cells correspond to high positive correlation; dark blue cells correspond to high negative correlation; pale green cells correspond to zero correlation.

After removing variables lacking non-zero data, we had 44 variables. The 445 44 x 44 correlation matrices were subjected to k-means clustering to see if variables from sub-systems of the operating

system (e.g, CPU-, user-, disk-, memory-, and network-related variables) would cluster together. If so, then we have reason to believe that an automated learning algorithm will, at the very least, recover what we already know about the system. In order to find "k" for the k-means clustering, the Bayesian information criterion (BIC) was found by running 100 iterations of each of 10 values for k (1-10) with random starting points. See the "Milestone Report" for details.

Figure 1 shows the 44 clustered variables plotted in 2 dimensions. Ovals added manually to highlight structure, and descriptive labels derived from each cluster's members. With clusters identified, the heat map visualization was improved by reordering the variables to group them by cluster. Figure 2 shows the resulting aggregate heat map.

Figure 1. Variable Clusters in 2-D Plot



II. Finding Servers with Atypical Behavior

The 43-dataset template was used as a standard to which all 445 datasets were compared by unraveling the correlation matrix into a vector and correlating against the template. The distribution of resulting correlations looks truncated normal (see Figure 3).

Figure 2. Normal Heat Map

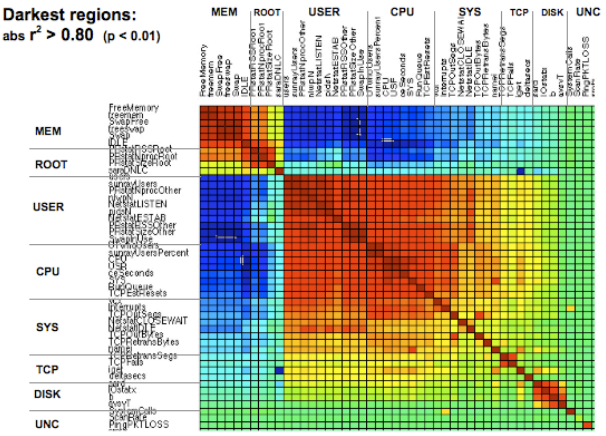
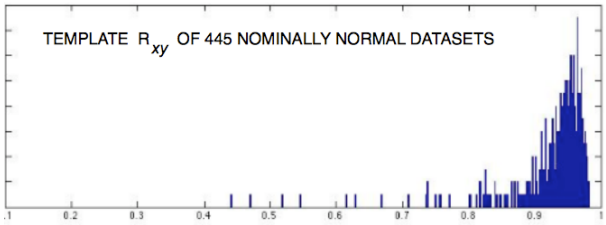


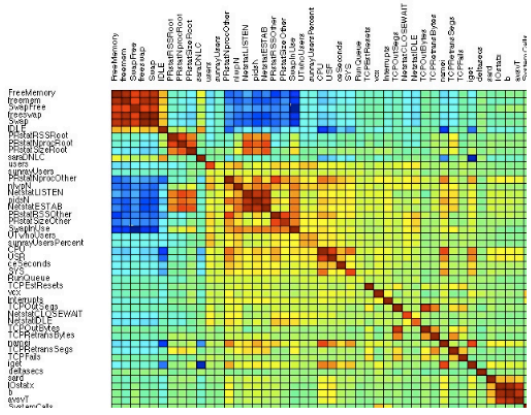
Figure 3. Distribution of Normal Correlations



Mean = 0.8827, sd = 0.0814

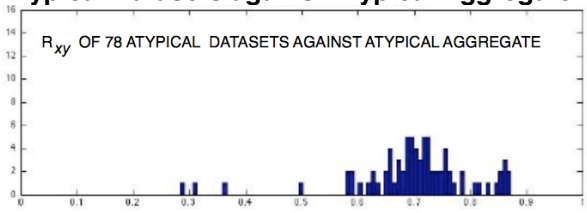
The same process was followed for server-day datasets identified as "atypical". Figure 4 shows one example of an atypical heat map. Note that the cluster structure is weak. Figure 5 shows the distribution of correlations for datasets identified as atypical, against an aggregate (formed by cell-wise averaging) of all 78 atypical datasets. Atypical datasets have low correlation with their own aggregate, meaning that they are heterogeneous in their atypicality.

Figure 4. An Atypical Heat Map



By contrast, 'normal' datasets are highly homogeneous with respect to their aggregate.

Figure 5. Distribution of Correlations for Atypical Datasets against Atypical Aggregate



Mean = 0.6998, sd = 0.1069

III. Using Supervised Learning to Classify Servers - Approach

1. Compose a training set of m datasets, in which some proportion 'p' are known to come from "healthy" servers and some proportion (1-'p') come from servers visually identified as "suspect".

2. Run multiple correlation on each set. Send the correlation matrices to a supervised machine learning algorithm (Support Vector Machine) using positive and negative examples identified from earlier steps. Generate a classification output and examine the accuracy of this algorithm in classifying the correlation outputs. Use different ratios of "healthy" to "suspect" servers in the training set, to see how this choice impacts the Test Error of the classification step. See next Section.

3. Also train an SVM on the raw data instead of the correlation outputs. It may be possible to speed up and streamline classification by skipping the computationally-intensive multiple correlation step.

IV. Training Set Composition & Test Error

Intuitively, the Training Set must contain examples of both positive and negative cases. But one can ask "how many of each?" Does the ratio of positive to negative traiing cases matter? In the absence of knowledge about the Test Set, then a 50-50 ratio makes sense. But what if we have knowledge of the ratio in the Test Set (i.e., the real world)? For example, in the case of "healthy" and "unhealthy" servers in a data center, there will always be more healthy servers than unhealthy ones, or at least the center's personnel hope so. If we want to train an optimal classifier to detect a relatively small number of unhealthy servers in a large group of healthy ones, does it make sense to train the classifier on a ratio of positive and negative cases that matches the real-world ratio? Or is 50-50 best?

There is probably a straightforward mathematical answer to this question based on the properties of the SVM algorithm. However, this author lacks the skills required to examine the problem from that angle and it is possible to test the idea empirically. Accordingly, the SVM algorithm was run under a set of varying Training Example ratios, as shown below. There were 445 normal servers and 78 atypical ones in the set, which constitutes a ratio of about 5.7 to 1. For the study, random samples were drawn from the entire set according to varying ratios of normal to atypical servers.

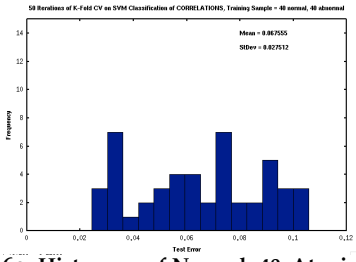
Normal-Atypical Ratios used were 1:1, 2:1, 3:1, 4:1, 5:1, and 6:1, in combinations ranging in magnitude from 20 normal:10 atypical up to 200 normal :50 atypical.

Multiple (50 or 100) runs were conducted for each ratio, with each run being a different random sample from the set, and mean and standard deviation for the ensemble of runs were computed. We started out with sets of 100 but they took a long time, so switched to runs of 50.

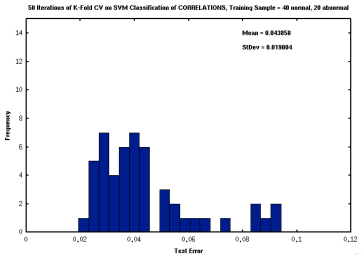
The SVM software used for this study is the smo_train.m script developed in Problem Set #2 of the cs229 course (Autumn 2009). Its default settings are 0.01 for tolerance and 10 for maximum number of passes through the training set. These are the values used for classification of correlation outputs.

Figures 6a-e show the Test Error distributions for each 50- or 100-iteration run. They are roughly gaussian, truncated at the low end and long-tailed on the high end. Medians would give lower estimates of test error, but means in this case are more conservative and for comparisons the two should be equivalent.

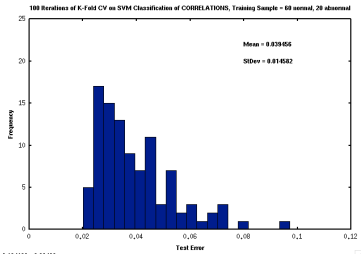
Figures 6a-e



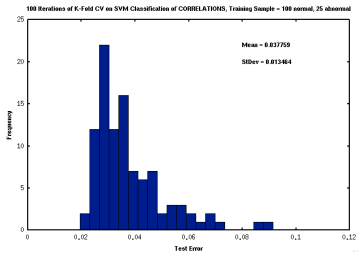
6a Histogram of Normal=40, Atypical = 40 (1:1 ratio)



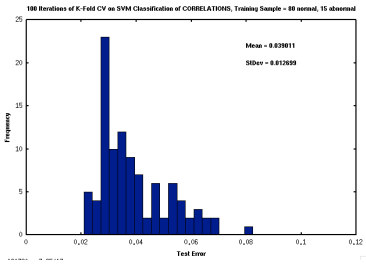
***6b* Histogram of Normal=40, Atypical = 20 (2:1 ratio)**



6c Histogram of Normal=60, Atypical = 20 (3:1 ratio)



6d Histogram of Normal=100, Atypical = 25 (4:1 ratio)



6e Histogram of Normal=80, Atypical = 15 (5:1 ratio)

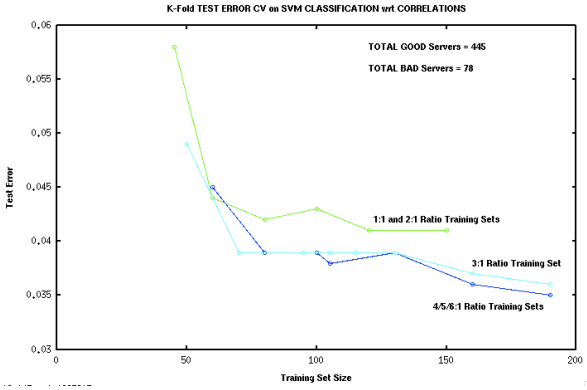
Distributions from the 1:1 & 1:2 Training Set ratios had average *stddev* of 0.021 compared to 0.012 for the 3:1-6:1 ratios, and so are more variable. Figure 7 shows mean Test Error wrt Training Set size, for various ratios. With standard deviations of 0.01-0.02, most of these differences are not statistically reliable. But the elevation of the 1:1 & 2:1 test error may be real, and taking that together with the greater spread of the distributions, better generalization performance probably comes from a Training Set that mirrors the expected Test set's ratio of "positive" to "negative" cases. NOTE - Figure 7 is an update of the plot shown at the Poster Session, and reflects additional runs of random-draw study. "Best" test error is now around 3.6%.

V. Using SVM to Classify Raw Data

So far we've done classification on the outputs of multiple correlation. As shown in Figures 3 and 5 above, the template-match distributions of Normal and Atypical servers are well separated (though not linearly separable) and our simple classifier achieved Test Error of about 3.6% under the best choice of Training Sets we could come up with. But correlation takes time because at least n(n-1)/2 correlations are needed, where n is the number of variables. It is of theoretical and practical interest to ask whether the same classification results can be obtained using the raw time series data and skipping the correlation step.

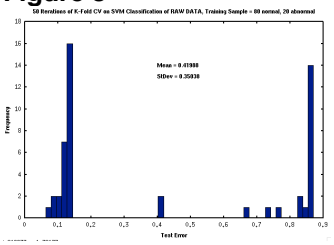
So we ran the same smo_train SVM algorithm on raw data files, by unraveling the 44-variable by 1400 interpolated-time-sample matrix for each

Figure 7



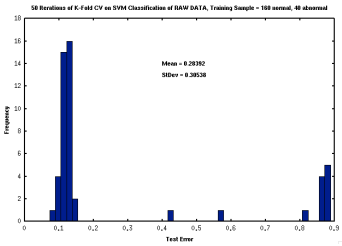
server-day into a long vector. Initial trials indicated the default settings (tolerance 0.10 and max_passes 10) were giving high and variable Test Error, so we decreased tolerance to 0.001 and increased max_passes to 40 and got lower mean error. But this greatly increased the time required to get a result, AND, upon examining the Test Error distributions, it became clear that something puzzling is going on. Figures 8 & 9 show the distributions using 4:1 Normal-Atypical Training Set ratios. They are bimodal with spikes in both the 90% error range and the 10% error range. Given a binary classification task, a random classifier would have a 50% error rate, so could the spike near 90% mean the classification is "correct" but for a sign change?

Figure 8



Histogram of Normal = 80, Atypical = 20 (4:1 ratio)

Figure 9



Histogram of Normal = 160, Atypical = 40 (4:1 ratio)

NOTE - The poster showed a plot of mean Test Error for the SVM raw data classification, but with these distributions mean error is not a good measure of what is going on. What IS going on?

VI. Fun with SVM

Linear SVM classification can also be used on datasets preprocessed with PCA or Cross-Correlation. These methods are visualizable in a 2D space. Figure 10 shows Normal & Atypical servers plotted by cross-correlation of UNIX variables **users** & **run queue**. Normal (green) have high magnitude (Y axis) and near-zero lag (X axis), and are tightly clumped in the 2-D space. Atypical (red) have low magnitude and high variance in lag. This dataset was linearly separable. (Nice ... though the plot seems to be missing at least green support vector) . In Figure 11, SVM uses radial basis function to cluster UNIX process, TCP, user, and disk.

Figure 10

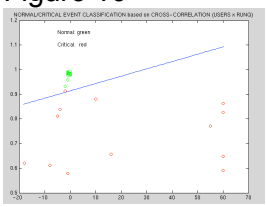
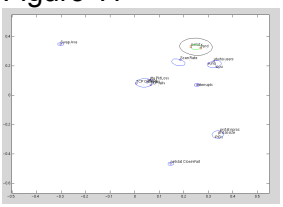


Figure 11



DISCUSSION

Normal servers have a "correlation print" that is distinctly different from atypical servers and the difference is classifiable by a simple SVM algorithm. However, SVM on the raw data seemed to reversed the classification of about 1/3 of the Test Set. Is this a property of the algorithm, the dataset, or an interaction of the two? An error of this author? If we can get raw-data SVM to work, then how should it be visualized? Is there a 2D representation?

Correlation outputs are approximately gaussian distributed, so this report *should* have included a gaussian model such as discriminant analysis or factor analysis, and compared it with SVM in terms of accuracy and efficiency. Unfortunately, time ran out.

IT'S BEEN A GREAT CLASS! THANKS!