BLR-D: Applying Bilinear Logistic Regression to Factored Diagnosis Problems

Sumit Basu
Microsoft Research
One Microsoft Way
Redmond, WA
+1 425 706-7971
sumitb
@microsoft.com

John Dunagan
Microsoft
One Microsoft Way
Redmond, WA
+1 425 705-8577
jdunagan
@microsoft.com

Kevin Duh
NTT Labs
2-4 Hikaridai, Seika-cho
Kyoto, Japan
+81 774 93-5315
kevin.duh
@lab.ntt.co.jp

Kiran-Kumar
Muniswamy-Reddy
Harvard University
Cambridge, MA 02138
+1 617 699-4384
kiran
@eecs.harvard.edu

ABSTRACT

In this paper, we address a pattern of diagnosis problems in which each of *J* entities produces the same *K* features, yet we are only informed of overall faults from the ensemble. Furthermore, we suspect that only certain entities and certain features are leading to the problem. The task, then, is to reliably identify which entities and which features are at fault. Such problems are particularly prevalent in the world of computer systems, in which a datacenter with hundreds of machines, each with the same performance counters, occasionally produces overall faults. In this paper, we present a means of using a constrained form of bilinear logistic regression for diagnosis in such problems. The bilinear treatment allows us to represent the scenarios with *J*+*K* instead of *JK* parameters, resulting in more easily interpretable results and far fewer false positives compared to treating the parameters independently. We develop statistical tests to determine which features and entities, if any, may be responsible for the labeled faults, and use false discovery rate (FDR) analysis to ensure that our values are meaningful. We show results in comparison to ordinary logistic regression (with L1 regularization) on two scenarios: a synthetic dataset based on a model of faults in a datacenter, and a real problem of finding problematic processes/features based on user-reported hangs.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning – *Parameter learning;* I.5.1 [**Pattern Recognition**]: Models – *Statistical;* G.3 [**Probability and Statistics**]: Nonparametric statistics; C.4 [**Computer Systems Organization**] Performance of Systems – *Reliability.*

General Terms

Algorithms, Reliability, Measurement, Performance, Experimentation.

Keywords

Bilinear Models, Logistic Regression, Bilinear Logistic Regression, Diagnosis Problems, Factored diagnosis, Non-Parametric Statistics, False Discovery Rate Analysis.

1. INTRODUCTION

Diagnosis problems appear everywhere in society: determining what disease an individual might have, determining who is responsible for a crime, and determining what may be causing a complex datacenter to fail are all important scenarios. In some instances, such as the datacenter, there are *J* instances of entities (computers), each of which produce a distinct set of the same *K* features (performance counters), yet faults are only labeled in terms of the entire ensemble (i.e., the datacenter failed to respond), and occur only rarely. In other words, the *JK* available features can be factored into *J* entities with *K* features each, while the labels are on the ensemble.

Such problems abound in the computer systems world: not only are there a variety of problems involving multiple computers as above, there is the more local problem of a single computer with hundreds of processes, each of which has dozens of performance counters associated with it. Furthermore, as in many diagnosis scenarios, each suspected cause can be very expensive to pursue. Missing a true cause is also undesirable, but if we find and eliminate other problems that may be masking it we can expect such missed causes to show up in later measurements. As such, while we still wish to achieve high detection rates, we are particularly concerned with minimizing the number of type I errors (false positives); this coupled with the small number of available labels makes such diagnosis tasks rather challenging.

Fortunately, the faults in such scenarios often also have a factored nature. For instance, in a datacenter, we expect that while only certain machines may be failing, it is often the case that they are failing for the same reasons. Such machines may be more prone to failure due to a common manufacturing defect (insufficient fan speed, improper disk mount, etc.), which is exacerbated when a certain set of features (CPU load, disk I/O) reach a certain level. Similarly, for the single machine case, while there may be several rogue processes, they are likely causing the system to hang in a similar (but unknown) manner – by using too much memory, CPU, etc., which the user's particular machine may be especially sensitive to.

Without the factored structure, a sensible approach would be to apply logistic regression with *JK* parameters, i.e., one for each entity-feature combination, using the labels as a target, and look for high-magnitude values amongst the resulting parameters. Of course, given the large number of parameters, a great deal of data would be necessary to avoid overfitting. In our approach, we still begin with logistic regression, but in bilinear form (as introduced in [9]) with only J+K parameters: one α for each entity and one β for each feature. To further improve the interpretability of the parameter and reduce symmetries in the solution space, we constrain the entity weights α to be positive. With many fewer parameters, it follows that we should be able to find meaningful parameter values (and hence far fewer false alarms) with far less data. Later, in Section 7, we develop an extension to the method that can model multiple modes of failure (multiple sets of α and β values).

We develop a test for finding causes from amongst the estimated parameters by examining the distributions of parameter values over multiple rounds of true and false labels. We then use false discovery rate (FDR) analysis [21] to ensure an acceptably low rate of false positives. We call the entirety of the procedure BLR-D, for bilinear logistic regression applied to diagnosis. We show empirical results on two data scenarios: synthetic data based on a model of a data center and real data from perceived hangs on an individual machine. We also compare our results against ordinary logistic regression (with L1 regularization and augmented with the same statistical tests) and show that our method can correctly identify causes with far fewer false alarms.

2. RELATED WORK

There is a long history of research in diagnosis problems in machine learning: the most prominent is the work on medical diagnosis as with the QMR-DT scenario (see, for instance [19, 13]). This and many other classically studied problems, though, tend to have a variety of causes (diseases) that lead to a set of common symptoms where the conditional probabilities of the symptoms given the diseases are known or can be estimated. The problem structure is thus inherently a graph and the natural approach is to apply inference or approximate inference in the hopes of determining the posterior over possible causes. The problems we are concerned with in this work do not have this structure, both in the sense that each entity produces an independent set of features, and in that we receive multiple instances of the data with an overall label as to the state of the ensemble. Another related area in this vein is multi-task learning (MTL) [3], in which the same observations (symptoms) are used to develop classifiers for multiple tasks (causes) that share a common representation; however, MTL requires separate labels for the individual tasks (causes), whereas we only have labels on the ensemble (the presence or absence of an overall fault).

There has also been much past work in using logistic regression for diagnosis problems that are not of a graphical form; this is particularly common in the medical domain. In many of these cases, researchers have interpreted regression parameters to identify causes as in [14], in which the authors are investigating the causes of spinal problems. To determine the statistical significance of these parameters, Wald's test is typically used, which computes a statistic on a single maximum likelihood estimate of the parameters from the data.

In terms of our methods, we are not the first to explore the use of bilinear structure in logistic regression problems. Dyrholm et al. [9] worked out the bilinear form for classification problems involving brain signals; they referred to it as bilinear discriminant analysis since their goal was classification. Each trial would contain J sensors producing K timepoints of data; their goal was to correctly classify whether the trial contained a stimulus or not. Our work, on the other hand, applies bilinear logistic regression to factored diagnosis problems, where the goal is not accurate classification but instead to determine the parameters that are likely to be responsible for the ensemble-level faults. In order to make the results more interpretable and to reduce symmetries amongst the possible solutions, we have adjusted the formulation so that the entity weights are constrained to be positive. We also construct a statistical test to determine when a parameter should be considered meaningful, and finally apply FDR analysis to determine appropriate significance levels for that test. Our contribution, then, is this overall technique, which we call BLR-D, the application of constrained bilinear logistic regression to diagnosis problems.

There has also been a variety of work in the systems world on diagnosing problems based on fault reports. In addition to algorithmic innovations, such work has introduced techniques for generating more useful features [17, 22, 23] and enabling hypotheses to be cheaply tested through sandboxing techniques [20]. The algorithmic techniques in this prior work have included hand-coded heuristics [11], hierarchical clustering [5], signal processing techniques [1], metric attribution [6], Bayesian techniques [24], factor graphs [15] and others. Some of the problems considered in this prior work can be modeled as a mapping of potential causes to features, e.g., determining which configuration setting is responsible for a certain application failure [24]. However, we are not aware of any prior technique that takes advantage of factorable data scenarios as we do here.

3. BILINEAR LOGISTIC REGRESSION

Since we have J entities (each with its associated α_j) that each produce K features (each with its associated β_k), we have a total of JK features f_{ijk} for each i of N samples; we begin by normalizing all features to have unit variance and zero mean. We also have N labels $y_1 \ldots y_N$ which we wish to explain with the model. As in ordinary logistic regression (see, for instance, [2]), we define a probability model for the labels as follows:

$$P(y_i) = \frac{1}{1+e^{-z_i}} = \sigma(z_i)$$

In the bilinear formulation, the weights for each feature contain both an entity specific (α) and a feature specific (β) weight:

$$z_i = \sum_j \sum_k \alpha_j \beta_k f_{ijk} + \delta$$

where δ is a bias term. In order to constrain the entity parameters α_j to be positive, we parameterize them as the square of a corresponding γ_j, i.e., $\alpha_j = \gamma_j^2$. While it is more traditional to do such a parameterization via the monotonic $\exp(\gamma_j)$ function, since there is then a unique mapping between the two parameters, we found empirically that this approach makes it difficult for the α_j to reach zero as it requires γ_j to go to $-\infty$. Our approach remedies this problem; furthermore, while there are now two possible values for γ_j for a given α_j, we are only interested in the uniqueness of the latter – it is the α_j that we wish to interpret.

We can express the total likelihood of all the observed data as

$$P(Y) = \prod_i \left(\sigma(z_i)\right)^{y_i} \left(1-\sigma(z_i)\right)^{1-y_i}$$

and the negative log likehood (NLL) as

$$-\,log\,P(Y) = -\sum_i y_i\,log\,\sigma(z_i) - \sum_i (1-y_i)\,log(1-\sigma(z_i))$$

Our goal is now to minimize the NLL; to do this, we need the gradients with respect to each of the parameters, α_j, β_k, and δ.

We first note that the gradient with respect to any parameter, e.g., γ_j, can be written via the chain rule:

$$\frac{\partial \mathrm{NLL}}{\partial \gamma_j} = \sum_i \frac{\partial \mathrm{NLL}}{\partial z_i} \frac{\partial z_i}{\partial \alpha_j} \frac{\partial \alpha_j}{\partial \gamma_j} = \sum_i (\sigma(z_i) - y_i) \frac{\partial z_i}{\partial \alpha_j} \frac{\partial \alpha_j}{\partial \gamma_j}$$

We thus only need to find the partials of z_i with respect to the parameters, which are as follows:

$$\frac{\partial z_i}{\partial \alpha_j} = \sum_k \beta_k f_{ijk} \qquad \frac{\partial \alpha_j}{\partial \gamma_j} = 2\gamma_j$$

$$\frac{\partial z_i}{\partial \beta_k} = \sum_j \alpha_j f_{ijk} \qquad \frac{\partial z_i}{\partial \delta} = 1$$

Given these partials, we perform the minimization of the NLL with respect to the *J+K+1* parameters using the L-BFGS method for quasi-Newton optimization [18].

Notice that the bilinear form is no longer strictly convex even with our positive constraint for the α_j: there are now a continuum of solutions which provide alternate and equivalent NLL values. For instance, we could reduce all α values by half and double all the β values to achieve the same solution. However, this will not adversely affect our gradient descent approach, as moving along such a continuum would always result in a zero gradient, and if the smallest possible gradient were zero L-BFGS would terminate. Thus we expect to land at some point along the optimal continuum; to achieve consistency between trials, we normalize the final α parameters by the sum of their values and scale the β values accordingly.

4. TESTING PARAMETERS FOR SIGNIFICANCE

Once we have optimized the parameters, there still remains the question of whether a particular α_j or β_k is significant. Because we wish to operate in regimes with small amounts of data with respect to the number of features, we expect some degree of overfitting, and as such a simple threshold would be a poor option. As a result, we seek a statistical test to distinguish meaningful parameters from noise.

For this purpose, we look to the work of Friedman et al. [12] on estimating the significance of proposed links in a learned Bayesian network. In their work, the authors used Efron's Bootstrap [10], i.e., they resampled their data with replacement, in order to get a distribution over whether a given link was deemed present or absent; they then made the decision to keep a link when the probability of the link occurring under this method was above a threshold.

In our case, the algorithm produces not a binary answer but a continuous parameter; comparing it to zero is not inherently meaningful. Instead, we draw a first population from the distribution over each estimated parameter value with the true labels using the Bootstrap, as well as a second population from the distribution over the values with sets of random labels (i.e., null distributions). We then compare the two populations to see with what probability they are coming from different distributions. If the populations are indeed different with sufficient statistical significance and the mean parameter value from the true labels is larger (in absolute value), we declare that the candidate parameter is among the causes of the observed faults. Because we do not know that the two populations will have normal distributions or even the same variances, it is not appropriate to use the standard Student's t-test. Instead, we use the non-parametric Mann-Whitney U-test [7], which makes no assumptions about the underlying distributions.

Such statistical tests come at substantial computational cost: by making multiple estimates of the parameters under true and false labels, we incur 40 times the cost in our experiments (20 rounds of each). The reader may wonder whether one could not simply apply a threshold to a single round and achieve nearly the same results. To show the advantages of the statistical method, we trace out the ROC curves for this approach (by varying the significance level) vs. choosing a simple threshold (by varying the threshold) on the baseline conditions from our datacenter experiments in Figure 1 below. While expensive, it is clear that there is a substantial advantage to using the statistical approach, both in terms of detecting causes and in reducing false alarms.

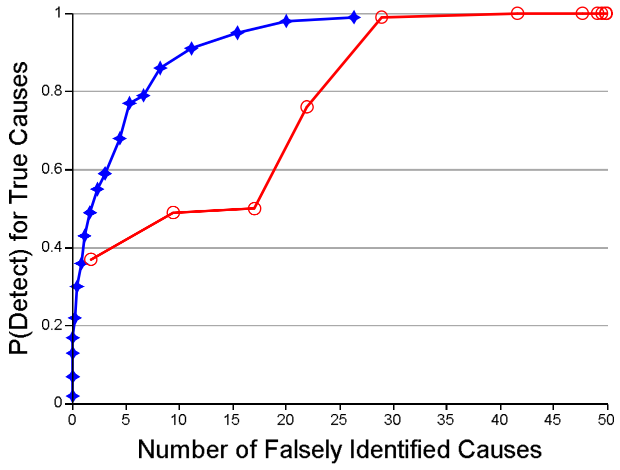

Figure 1. ROC for BLR method using proposed statistical tests (BLR-D, blue +) and a simple threshold (BLR-threshold, red o) under the baseline condition averaged over 10 trials.

5. COMPUTING THE FALSE DISCOVERY RATE

For our experiments with synthetic data, we know the true causes, and as such can report on the detection rate and false alarms from the algorithm. In general, though, we need another means to measure performance when we do not know the causes *a priori.* Fortunately, our problem coincides well with the recent work in biostatistics on estimating the false discovery rate (FDR) for an algorithm [21,16]. As in their work, we have the challenge of reporting significance on a large number of parameters as well as the concerns over the expense of investigating a large number of false alarms. They describe their approach in terms of finding $FDR(a)$ for algorithm a, where a reports whether parameters are significant. This FDR quantity is the average number of parameters falsely reported as significant $F(a)$ under null distributions D^q (i.e., sets of false labels) divided by the average number of parameters reported with true labels, $S(a)$. The denominator is commonly approximated by a single value on the dataset of interest, $N(D,a)$, which is the number of parameters reported by a as being significant. The distributions D^q are then created by assigning random labels to the data. Following [22], we have:

$$FDR(a) = E\left[\frac{F(a)}{S(a)}\right] \cong \frac{E[F(a)]}{E[S(a)]} \cong \frac{\sum_{q=1}^{Q} \frac{N(D^q,a)}{Q}}{N(D,a)}$$

The resulting quantity tells us what fraction of the parameters reported as significant that we can expect to be false alarms. Ideally we want this quantity to be close to zero as possible. In practice, a user of FDR will tune the parameters of the algorithm (in our case, the U level of the significance test) in the hopes of achieving a minimal FDR while still producing a non-empty set of results for the true labels, i.e., $N(D, a)$>0. If a low FDR is not achievable, it is likely that the algorithm a cannot produce reliable results for the given dataset.

6. EXPERIMENTS

To illustrate our technique, we perform two sets of experiments: the first set is on synthetic data generated by a model of a datacenter, in which we can measure true detection rates and false positives; the second is from real data on process behavior on a personal computer.

6.1 Machines in a Datacenter

Datacenters have recently received enormous attention in the software industry for their ability to deliver compelling services. Inside the datacenters, these services run on large sets of machines with homogeneous software, though they do not always have homogeneous roles: a few machines may provide some form of coordination function for other machines in the pool. Some of the most well-known examples of such software are BigTable [4] and Dynamo [8], storage services that runs the same piece of software across a large number of machines. On each of the J machines, there are K features measuring such aspects as CPU, network, and disk utilization. To an external observer, the success or failure of one of these systems to meet a Service Level Agreement (SLA) will manifest itself as a label on the entire ensemble.

We developed a synthetic model that captures the characteristics of such datacenter systems. We have used this synthetic model for this first set of results so that we can manipulate individual parameters and report accuracies with respect to the true causes. This also gives us an opportunity to make the data public, so that future researchers can test their own models against it. The datasets and an explanation of their format are available at:

http://research.microsoft.com/~sumitb/blrd

In the baseline system, there are 30 machines with 30 features each; this means there are 900 features for each timestep. For each feature, we generate zero-mean, normally distributed values; the variance for each feature is separately drawn from a uniform distribution for each dataset. We select (as a baseline) five machines to be "fault-prone" and we select five features to be "fault-causing." We then generate all features for all machines for each timestep; if any of the "fault-prone" machines in the ensemble has a "fault-causing" feature that is greater than two standard deviations from zero, then with probability p_{fault} (0.8 in the baseline) an SLA violation is generated for the datacenter. The baseline dataset contains 1000 labeled samples of data with balanced numbers of positive and negative examples.

In each of the three experiments below, we varied one of the parameters from the baseline. For each unique parameter setting we produced 10 synthetic trials. All values displayed in the figures below are averaged over the ten corresponding trials to smooth the effects of individual variations. For each setting, we then computed the distributions of all parameters under false and true labels values as described in Section 4 over 20 rounds each with a Bootstrap fraction of 0.8; we then used the Mann-Whitney test on each parameter with a U value of 0.01 (for BLR-D) and 0.001 (LR-D). We chose these particular values as they tended to give commensurate detection performance for both methods.

In all experiments, the true j fault-prone machines and k fault-causing features result in j+k significant features for BLR-D to detect, whereas for LR-D there are jk significant features. The detection rates reported are in terms of these total numbers. We also report the *number* of false positives, i.e., the number of parameters incorrectly identified as significant, as opposed to the *probability* of false alarms, to illustrate the true cost to the analyst of investigating these false leads. For comparison, we show a traditional ROC for the baseline parameters in Figure 2 below. BLR-D still performs better in terms of probabilities, but the real costs of LR-D become clear in Figures 3-5.

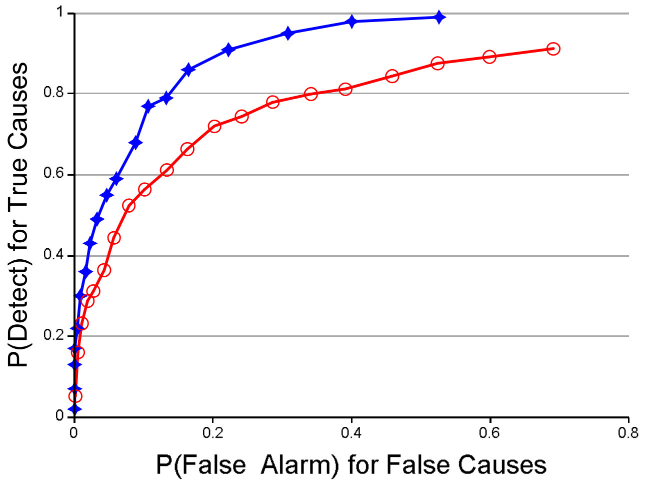

Figure 2. Traditional ROC comparing our method, BLR-D (blue +), to LR-D (red o) showing *probabilities* of false alarms for the baseline parameter values; later figures show the *number* of false alarms to illustrate the true cost of investigating false leads.

6.1.1 Varying the Number of Samples

The first experiment involved varying the number of samples for the dataset; the results are shown in Figure 3. In this case, we took these samples from a larger set of generated samples so that we could have a balanced number of positive and negative examples. As we would expect, increasing the number of samples results in a higher probability of detecting the true culprits for both methods. However, our method achieves similar levels of detection to LR-D with far fewer numbers of false alarms.

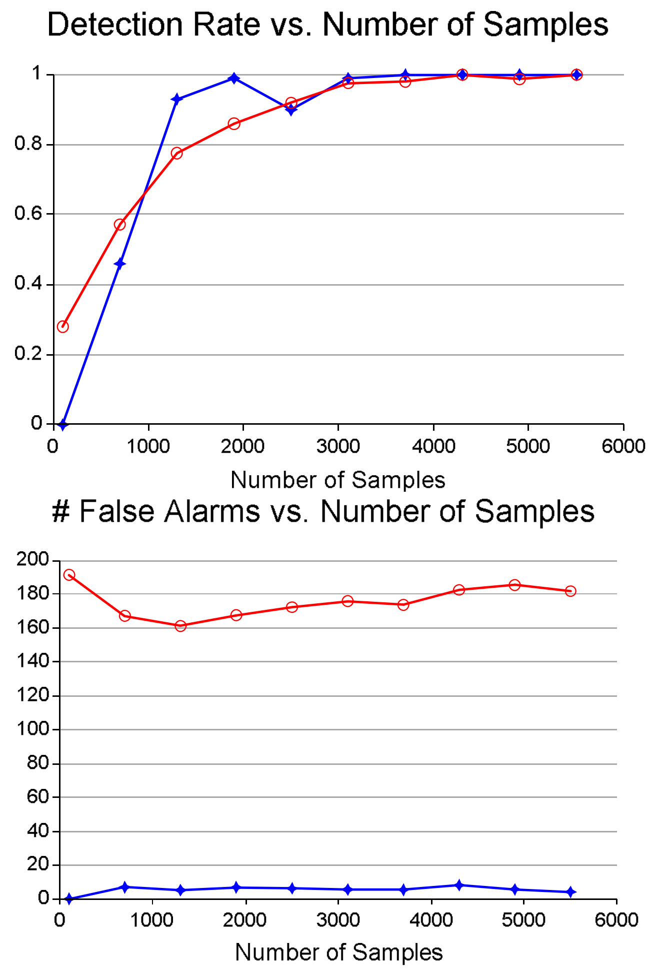

Figure 3. Detection rate for the true causes of datacenter faults (left) and number of falsely reported machines and features (right) for BLR-D (blue +) and LR-D (red o) vs. a varying number of labeled samples.

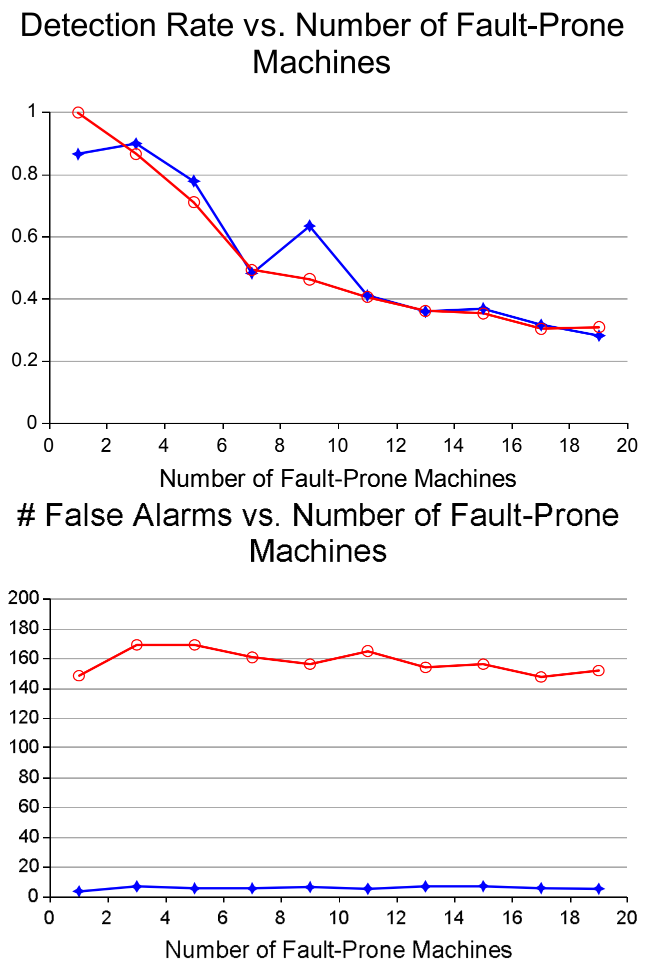

Figure 4. Detection rate for the true causes of datacenter faults (left) and number of falsely reported machines and features (right) for BLR-D (blue +) and LR-D (red o) vs. a varying number of fault-prone machines.

6.1.2 Varying the Number of Fault-Prone Machines

The second experiment involved changing how many of the 30 machines in the baseline datacenter were prone to having faults while maintaining a fixed number of (balanced) samples. This problem becomes more difficult as more machines become fault-prone, as there are more possible (true) causes that each fault can be attributed to, and thus we see fewer faults for each individual cause. In Figure 4, we see an expected decline in detection performance for both methods as the number of fault-prone machines increases, but again our method has a much smaller number of falsely identified causes.

6.1.3 Varying the Number of Machines in the Datacenter

In this experiment, we increased the number of machines in the datacenter while keeping the number of (balanced) labeled samples constant. This made things increasingly difficult as there were more opportunities to overfit to the data. As the number of machines increases, the number of parameters for our method increases as $J+K$ and as JK for LR-D. As a result, with 95 machines in the datacenter, our method had 125 possible causes to contend with, while LR-D had all 2850 to sort out. This resulted in a linear increase in false alarms for LR-D with the number of machines (see Figure 5).

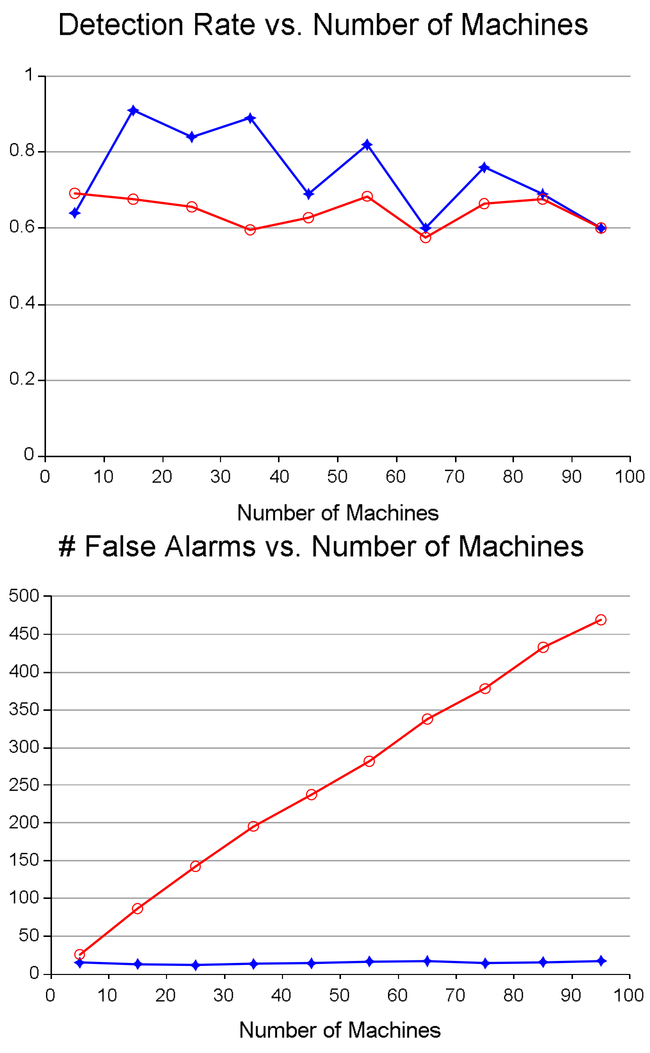

Figure 5. Detection rate for the true causes of datacenter faults (left) and number of falsely reported machines and features (right) for BLR-D (blue +) and LR-D (red o) vs. a varying number of machines in the datacenter.

For this experiment, we chose a higher point in the ROC curve for BLR-D (U=0.1) to show that we can achieve a higher accuracy than LR-D while still producing many fewer false alarms. At our standard setting of U=0.01 for BLR-D on this experiment, the detection rate falls by about 10% (absolute) on average, but the average number of false alarms drops by 66%, from 15 to 5.

6.2 Processes on a Machine

The second scenario we considered was diagnosing user-reported machine hangs. Based on well-known causes of unresponsiveness problems (such as swapping), we created a process called WhySlowFrustrator that occasionally and intentionally caused such unresponsiveness by increasing CPU, memory, and IO loads. The user generated a label by pressing a hotkey whenever the machine became unresponsive, which brought up a labeling UI (the WhySlow process). Our dataset (from one user) is particularly difficult in that labels are few and far between: over two months and 86,400 samples, there were only 63 positive labels, yet there were 104 active processes, each with 28 features, for a total of 2,912 features per timestep. To add to the difficulty, the user did not press the key for all instances (the user wasn't there, didn't notice, etc.); there were also cases when he did press the key but WhySlowFrustrator was not active (i.e., he was frustrated by some other process).

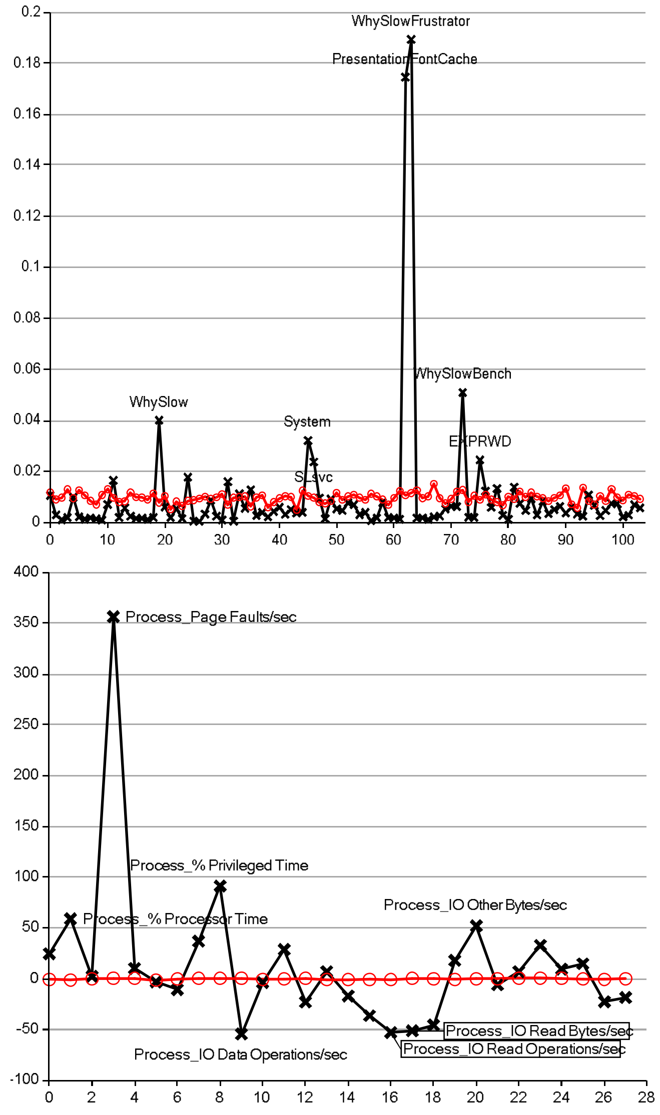

Figure 6. Learned entity weights (left) and feature weights (right) for processes on a machine under true labels (black x) and false labels (red o) of system hangs.

We sampled 100 negative examples per positive example, resulting in 6,363 samples. We then performed BLR-D over 20 rounds (with a bootstrap fraction of 0.99 given the sparsity of positive labels) for the true labels as well as the false labels. Computing the FDR then meant doing the same procedure, now for multiple rounds of D^Q, i.e., data with false labels. We then adjusted the threshold on the U value from the Mann-Whitney test until we achieved the smallest possible FDR while maintaining a non-empty set of identified causes for the true labels. In our data, we were able to reduce the FDR to zero with a U value of 0.02 while still identifying three entities as causes under the true labels.

The mean estimated entity weights and feature weights are shown in Figure 6. Only *WhySlowFrustrator*, *Presentation-FontCache*, and *WhySlow* were identified as causes at the required significance level (U=0.02), as well as nine features, all related to memory, CPU, and IO utilization. We know that *WhySlowFrustrator* is a true cause, and the selected features correspond well to its modes of behavior. *WhySlow* is a classic case of correlation vs. causation (GUI appearing when the hotkey was pressed). *PresentationFontCache* was unexpected: this is a Windows process which often takes up a significant amount of memory. It is possible that the large amount of swapping caused by *WhySlowFrustrator* may have resulted in this process being consistently swapped out; it is also possible that it was an independent source of frustration. Even if we consider it to be an error, though, this experiment shows that our method can identify true causes of problems in real-world scenarios with low numbers of false alarms.

7. EXTENDING BLR-D TO MULTIPLE MODES

As we discussed earlier, BLR-D works best for situations where there are multiple entities at fault that are failing *in the same way*. But what happens if there are multiple modes of failures for different subsets of entities? Consider a case where one set of machines has a problem when the network I/O is too high and disk I/O too low, and another has a problem when the disk I/O and CPU usage are too high. In our representation above, both conditions would have to be represented in the same set of α's and β's; since no assignment of values could cover both, the best "compromise" fit might be achieved by snapping to one of these modes, treating the other as unexplained noise, or worse yet by settling on other (erroneous) parameter values. Fortunately, a small extension to our method can cover such situations, at the expense of additional free parameters and identifiability issues.

Remember that the core of our formulation is simple logistic regression but with a parametric form for the JK weights: if we were to index the parameters as a single list, the weight for feature f_{jK+k} would be $\alpha_j\beta_k$. Imagine now that we put all of the weights into a J by K matrix W, where $w_{jk} = \alpha_j\beta_k$. Representing the vectors of parameters now as α and β, we can express this as

$$W = \alpha\beta^T$$

We are thus forming a rank-1 model of the underlying W. This gives us the greatest parsimony of description, but we can easily extend this to higher rank models:

$$W = \alpha_0\beta_0^T + \alpha_1\beta_1^T + \cdots$$

Or, more generally,

$$W = \sum_l \alpha_l\beta_l^T = \mathrm{AB}^\mathrm{T}$$

Where the lth column of the rank-L matrices A and B are the vectors of parameters α_l and β_l respectively. This expands our overall model to be

$$p(y_i) = \sigma(z_i)$$

$$z_i = \sum_l \sum_j \sum_k \alpha_{lj}\beta_{lk}f_{ijk} + \delta$$

With each new mode l, we increase the number of parameters by J+K, but we also allow for an additional mode of fault behavior. In this way, we can increase the number of parameters from J+K to JK by steps of J+K, depending on our modeling needs. The partials remain mostly the same, though there are now l times as many:

$$\frac{\partial z_i}{\partial \alpha_{lj}} = \sum_k \beta_{lk}f_{ijk} \qquad \frac{\partial \alpha_{lj}}{\partial \gamma_{lj}} = 2\gamma_{lj}$$

$$\frac{\partial z_i}{\partial \beta_{lk}} = \sum_j \alpha_{lj}f_{ijk} \qquad \frac{\partial z_i}{\partial \delta} = 1$$

We can apply the same optimization approach as in Section 4 to estimate the parameters and then normalize each column of A by its sum (and multiply the corresponding column of B by that sum). However, there is a substantially increased challenge for identifiability due to the multiple modes: if we perform the multiple bootstrap runs required by the BLR-D procedure, we cannot assume the same modes will be assigned to the same index l each time.

There are a variety of approaches by which we could attempt to find the correspondence between the modes $\alpha_l\beta_l^T$; the simplest would be to choose assignments by minimizing distance measures between them. Another approach, in analogy to the singular value decomposition, would be to order the modes in terms of their explanatory power (their ability to reduce the NLL). We could thus greedily select modes in decreasing order of additional explanatory power. Missing a single mode in a scheme like this, however, could wreak havoc on the remaining correspondences.

Alternatively, since A is constrained to be positive, if a given entity j is responsible for *some* faults, on each run there should be *some* columns in row j of A that would have a value significantly greater than 0 (though these could be different columns on different runs). One approach would then be to take a sum over the columns of A ($\alpha_S = \mathrm{A} * 1$), and then perform the statistical tests described in Section 4 on the elements of the resulting vector. This approach would only allow us to identify fault-causing entities and not features, but the underlying model would be able to account for the multiple modes and not be forced into compromise solutions as described above.

8. DISCUSSION

For certain applications, such as the systems scenarios presented in the Experiments section, a factored diagnosis model can be quite natural. For those cases, our method of using a bilinear form of logistic regression paired with (1) constraining the entity weights to be positive, (2) testing for the significance of individual parameters, and (3) using the FDR to find appropriate thresholds of significance proved to be an effective means of providing interpretable results and diagnosing causes with very low rates of false positives.

In our future work, we plan to gather machine data from datacenters to see how well our method works under real-world conditions. We also wish to explore the behavior of the multimodal extension of BLR-D, particularly in terms of resolving its identifiability issues, and considering the tradeoff of its greater explanatory power with its greater complexity.

9. REFERENCES

[1] Aguilera, M., Mogul, J., Wiener, J., Reynolds, P., and Muthitacharoen, A. "Performance Debugging for Distributed Systems of Black Boxes." In *Symp. on Operating Sys. Principles (SOSP),* 2003.

[2] Bishop, C.M. *Pattern Recognition and Machine Learning.* New York: Springer, 2006.

[3] Caruana, R. "Multitask Learning." *Machine Learning* 28: 41-76. 1997.

[4] Chang, F., Dean, J., Ghemawat, S., Hsieh, W., Wallach, D., Burrows, M., Chandra, T., Fikes, A., and Gruber, R. "Bigtable: A Distributed Storage System for Structured Data." In *Operating Sys. Design and Implementation (OSDI)*, 2006.

[5] Chen, M., Kiciman, E., Fratkin, E., Fox, A., and Brewer, E. "Pinpoint: Problem Determination in Large, Dynamic Internet Services." In *Dependable Sys. and Networks (DSN)*, 2002.

[6] Cohen, I., Zhang, S., Goldszmidt, M., Symons, J., Kelly, T., and Fox, A. "Capturing, Indexing, Clustering, and Retrieving System History." In *Symp. on Operating Sys. Principles (SOSP),* 2005.

[7] Conover, W. J. *Practical Nonparametric Statistics (3rd Ed.).* New York: Wiley, 1980.

[8] DeCandia, G., Hastorun, D., Jampani, H., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., and Vogels, W. "Dynamo: Amazon's Highly Available Key-value Store." In *Symposium on Operating Systems Principles (SOSP),* 2007.

[9] Dyrholm, M., Christoforou, C., and Parra, L. C. "Bilinear Discriminant Component Analysis." *JMLR* 8: 1097-1111. 2007.

[10] Efron, B. and Tibshirani, R. *An Introduction to the Bootstrap.* New York: Chapman & Hall, 1993.

[11] Engler, D. and Ashcraft, K. "RacerX: Effective, Static Detection of Race Conditions and Deadlocks." In *Symposium on Operating Systems Principles (SOSP),* 2003.

[12] Friedman, N., Goldszmidt, M., and Wyner, A. "On the Application of the Bootstrap for Computing Confidence Features of Induced Bayesian Networks." In *Proceedings of Artificial Intelligence and Statistics*, 1999.

[13] Jaakkola, T. and Jordan, M. I. "Variational Probabilistic Inference and the QMR-DT Network." *Journal of AI Research* 10, pp. 291-322. 1999.

[14] Konno, S., Hayashino, Y., Fukuhara, S., Kikuchi, S., Kaneda, K., Seichi, A., Chiba, K., Satomi, K., Nagata, K., and Kawai, S. "Development of a Clinical Diagnosis Support Tool to Identify Patients with Lumbar Spinal Stenosis." *Eur. Spine Journal* 16 (11): 1951-1957. Nov. 2007.

[15] Kremenek, T., Twohey, P., Back, G., Ng, A., and Engler, D. "From Uncertainty to Belief: Inferring the Specification Within." In *Operating Sys. Design and Implementation (OSDI)*, 2006.

[16] Listgarten, J. and Heckerman, D. "Determining the Number of Non-Spurious Arcs in a Learned DAG Model." In *Proeedings of UAI,* 2007.

[17] Lu, S., Park, S., Hu, C., Ma, X., Jiang, W., Li, Z., Popa, R., and Zhou, Y. "MUVI: Automatically Inferring Multi-variable Access Correlations and Detecting Related Semantic and Concurrency Bugs." In *Symposium on Operating Systems Principles (SOSP),* 2007.

[18] Nocedal, J. "Updating Quasi-Newton Matrices with Limited Storage." *Mathematics of Computation* 35: 773-782. 1980.

[19] Shwe, M., Middleton, B., Heckerman, D., Henrion, M., Horvitz, E., Lehmann, H., and Cooper, G. "Probabilistic Diagnosis Using a Reformulation of the INTERNIST-1/QMR Knowledge Base." *Methods of Information in Medicine* (30): 241-255. 1991.

[20] Srinivasan, S., Kandula, S., Andrews, C., and Zhou, Y. "Flashback: A Lightweight Extension for Rollback and Deterministic Replay for Software Debugging." In *USENIX Annual Technical Conference*, 2004.

[21] Storey, J. D. and Tibshirani, R. "Statistical Significance for Genomewide Studies." In *Proc Natl Acad Sci USA*, 100(16):9440–9445, August 2003.

[22] Tan, L., Yuan, D., Krishna, G., and Zhou, Y. /*iComment: Bugs or Bad Comments?*/. In *Symposium on Operating Systems Principles (SOSP),* 2007.

[23] Verbowski, C., Kiciman, E., Kumar, A., Daniels, B., Lu, S., Lee, J., Wang, Y., and Roussev, R. "Flight Data Recorder: Monitoring Persistent-state Interactions to Improve Systems Management." In *Operating Systems Design and Implementation (OSDI)*, 2006.

[24] Wang, H., Platt, J., Chen, Y., Zhang, R., and Wang, Y. "Automatic Misconfiguration Troubleshooting with PeerPressure." In *Operating Sys. Design and Implementation (OSDI)*, 2004.