

Soft Failure Detection using Factorial Hidden Markov Models

Guillaume Bouchard, Jean-Marc Andreoli
Xerox Research Centre Europe

Abstract

In modern business, educational, and other settings, it is common to provide a digital network that interconnects hardware devices for shared access by the users (e.g., in an office where printers are available for use by all the office workers). In such a context, so-called “soft” failures, where a device silently starts working in degraded mode, may easily go un-noticed for a long time, resulting in potential productivity loss. It is therefore advantageous to enable system administrators to identify soft failures at an early stage. We propose here a probabilistic method using variational inference on a factorial hidden Markov model to automatically discover soft failures, based on the analysis of simple usage information which is normally logged by the network infrastructure. We propose to mine these logs in order to discover statistically significant deviations in the usage behaviour of the overall infrastructure, and we identify such deviations with soft failures, or, in any case, situations of interest to an administrator.

1. Introduction

In networked printing infrastructures, the network advantageously enables users to send a job to any selected hardware device of the network. Normally, a user will choose to print on a primary device which is located close to him/her, such as in the same building, on the same floor or in the same wing of the same floor. However, there are a number of circumstances in which the user may decide to use another device than the primary one. In particular, if the primary device of a user becomes unavailable, then that user will have to choose another one.

One task of the administrator of an infrastructure is ensuring that the devices are operational. If a device becomes unavailable due to a malfunction that causes the device to fail completely, the administrator may be made aware of the problem by complaints from inconvenienced users, or by a direct notification from the device through the network (some devices offer such self-monitoring functionality). Sometimes, however, the device does not become com-

pletely unavailable, but rather suffers a malfunction, degradation, improper configuration, or other non fatal problem, which do not cause self-monitoring equipments to raise an alert. For example, the device may begin to produce dirty, ruffled, creased, or otherwise marred sheets, or jam the sheets repeatedly, or the device may be misconfigured so as to be unable to correctly print certain fonts, or so forth. Such situations are called “soft” failures. The soft failing device still works, just not as well or as efficiently as before.

When a soft failure occurs, users may or may not abandon the device entirely: they may continue to use it for less critical jobs, or for jobs which are unaffected by the soft failure. Also, users are less likely to complain to the administrator about a soft failure, because the level of inconvenience is typically less than in the case of complete unavailability. Rather, they will tend to choose another device for those jobs that are seriously affected by the soft failure, and continue to use the failing device for jobs in which the impact of the soft failure is tolerable. As a result, the system administrator may be informed of the soft failure only after it has had a severe impact on productivity, or when the soft failure progresses to complete unavailability.

It has often been advocated that the ability to gracefully degrade service in case of failure is a prominent feature in favour of a networked service organisation. The discussion above suggests that it also has a hidden cost, due to the over-reliance of users on that feature: as the degraded form of the service becomes sustainable, users tend to adapt to the degraded mode instead of looking for ways to restore full operability, and this in turn results in an overall degradation of the productivity which may go un-noticed for some time.

Accordingly, it is advantageous to enable system administrators to identify soft failures at an early stage. We propose here a method to automatically discover soft failures, based on rudimentary usage information which is typically collected by the network infrastructure and made available to administrators in the form of logs. We propose to mine these logs in order to discover statistically significant deviations in the usage behaviour of the overall infrastructure, and we identify such deviations with soft failures, or, in any

case, situations of interest to an administrator. The purpose is not to take decisions in place of the administrator, but to assist his/her decision by synthesising relevant characteristics of the available usage information.

2. Description of the soft failure detection process

Outline of the method Typically, the devices usage log maintained by the servers that spool the jobs in the infrastructure records information about each job. This information is often collected for various purposes such as billing of the users or sizing or configuration of the infrastructure. In the sequel, we assume the following job features are recorded:

- a timestamp of when the job was submitted,
- some identification of the user who submitted the job,
- some identification of the device to which the job was sent,
- optionally other information such as the type of job (black only or colour), the type of paper or other print medium used for executing the job, etc.

Our purpose is to guess, given a log of past jobs, the present soft failure state of each device in the infrastructure. The soft failure state of a device d at time t is represented by a number $s \in [0, 1]$ and can be defined as the proportion of users who would be “satisfied” if they launched a job at time t on device d . Hence, $s = 1$ means that the users perceive device d as properly working at time t whereas $s = 0$ means that device d is unlikely to be used at all at time t . By this very definition, the soft failure state of a device cannot be observed directly. It is a hidden variable which can only be apprehended through observations (actual attempts to use the device) which depend on the soft failure state, but this dependence is far from deterministic. It is therefore natural to turn to a probabilistic approach to estimate such hidden variables. Note that since our variables are intrinsically unobservable (not just costly to observe), only fully unsupervised methods are suitable here. Our method is based on a probabilistic model and is composed of the following steps:

- *Learning*: A snapshot of several days of printing logs is used to learn the model parameters. The parameters can be re-learned at regular intervals using fresh data. Alternatively, an online learning algorithm can be used.
- *Inference*: While in operation, the probabilistic model, instantiated with the parameters learnt at the previous step, is used to estimate the soft failure state of all the

devices. If the estimated state s of a device d reaches a critical point (e.g. falls below a predefined threshold), the system generates an alert sent to the administrator.

Formal description Let N_u (resp. N_d) be the number of users (resp. devices) of the infrastructure, assumed constant over time. Thus, each user (resp. device) can be identified by an index $u \in \{1, \dots, N_u\}$ (resp. $d \in \{1, \dots, N_d\}$). A period of time in the life of an infrastructure is captured by a set of random variables.

- **Observed variables.** The log of the jobs recorded during the period is given by a triple of sequences $(\mathbf{t}, \mathbf{u}, \mathbf{d})$ of same length N (the number of recorded jobs in the log). We write $\mathbf{t} = t_{1:N}, \mathbf{u} = u_{1:N}, \mathbf{d} = d_{1:N}$. For each $i = 1, \dots, N$, the i -th job in the log is described by its timestamp t_i (ranging over real numbers), the index of its user u_i (ranging over $\{1, \dots, N_u\}$) and the index of its device d_i (ranging over $\{1, \dots, N_d\}$). By construction, sequence \mathbf{t} is assumed to be non-decreasing (in fact strictly increasing, as we assume that two jobs never occur exactly at the same instant).
- **Unobserved variables.** For each job $i = 1, \dots, N$ and device index $d \in \{1, \dots, N_d\}$, the unobserved soft failure state of device d at time t_i is denoted s_{di} . The state of the whole infrastructure at the time of the i -th job is therefore represented by the vector $\mathbf{s}_i = (s_{di})_{d \in \{1, \dots, N_d\}}$. The sequence of these vectors $\mathbf{s}_{1:N}$ is denoted \mathbf{s} .

The stochastic dependency between the unobserved \mathbf{s} and the observed $(\mathbf{t}, \mathbf{u}, \mathbf{d})$ is assumed to be entirely captured by a parameter θ and the form of the probability distribution $\mathbf{p}(\mathbf{s}|\mathbf{t}, \mathbf{u}, \mathbf{d}; \theta)$ is assumed to be known. We give in the next section an example of such parameterisation. In a learning phase, we are given a reference log $(\mathbf{t}^{(o)}, \mathbf{u}^{(o)}, \mathbf{d}^{(o)})$ from which we estimate the parameter $\hat{\theta}$. In an inference phase, we are given a running log $(\mathbf{t}, \mathbf{u}, \mathbf{d})$ of length i and we compute the current state distribution $\mathbf{p}(s_{di}|\mathbf{t}, \mathbf{u}, \mathbf{d}; \hat{\theta})$ for each device d . Administrator alerts can be based on this distribution, for example by taking its mode.

3. A Factorial Hidden Markov Model

The proposed parameterisation of our problem is best represented using the graphical model of figure 1, which is an instance of a Factorial Hidden Markov Model (FHMM)[3]. The underlying assumptions are the following: the state dynamics of any two devices are independent (1); each individual device state dynamic is Markovian (2); at each job, the choice of the device depends only on the user and the current state of the infrastructure (3). Formally

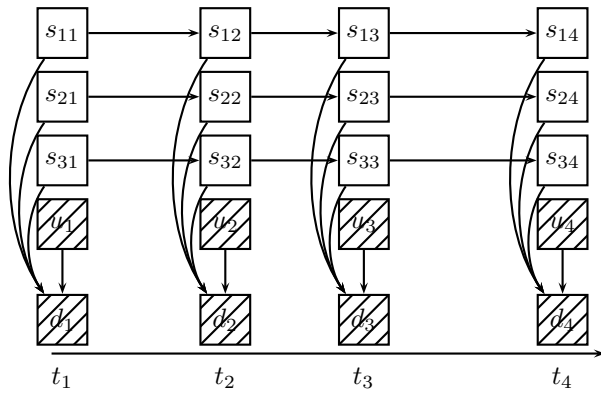


Figure 1. The Dynamical Bayesian Network of the full model with $N_d = 3$ devices.

(omitting the variables $t_{1:i}$ and parameter θ):

$$\mathbf{p}(s_{i+1}|s_{1:i}) = \prod_{d=1}^{N_d} \mathbf{p}(s_{d(i+1)}|s_{d(1:i)}) \quad (1)$$

$$\mathbf{p}(s_{d(i+1)}|s_{d(1:i)}) = \mathbf{p}(s_{d(i+1)}|s_{di}) \quad (2)$$

$$\mathbf{p}(d_i|s_{1:i}, u_{1:i}) = \mathbf{p}(d_i|s_i, u_i) \quad (3)$$

Hence, to complete the model, we need to specify the time series distributions $\mathbf{p}(s_{d(i+1)}|s_{di}, \theta)$ and the user device choice distributions $\mathbf{p}(d_i|s_i, u_i, \theta)$. There are many possible alternatives for these distributions. In the following we present one which appears to work well in practice, captured by equations (4) and (6) below.

3.1. Time series model

To simplify, we discretise the device state space: the state variables s_{di} are constrained to take their value in a finite space $\Sigma = \{\sigma_1, \dots, \sigma_L\}$ where $1 = \sigma_1 > \dots > \sigma_L \geq 0$ for some integer value L (the size of the state space). In this case, one can interchangeably work with the state s_{di} or its index S_{di} ranging over $\{1, \dots, L\}$ and defined such that $s_{di} = \sigma_{S_{di}}$. This allows the use of the traditional transition matrix notation. The $L \times L$ transition matrix of the Markov chain of a device state between any instants t_a and t_b is assumed to be independent of the device and given by

$$T_{t_a, t_b} = \exp(-Q(t_b - t_a))$$

where Q is a parameter matrix whose rows sum to zero. This model comes from the continuous time Markov process theory, where matrix Q is called the infinitesimal generator¹. In our case, we obtain

$$\mathbf{p}(s_{d(i+1)}|s_{di}) = (T_{t_i, t_{i+1}})_{S_{di}, S_{d(i+1)}} \quad (4)$$

¹Since, here, the state space is discretised, the infinitesimal generator is a matrix.

For the specific case $L = 2$ (binary discretisation), the state dynamic has a clear interpretation in terms of breakdown/repair operations. The infinitesimal generator is $Q = \begin{bmatrix} -\rho & \rho \\ \tau & -\tau \end{bmatrix}$ where τ is the device failure rate and ρ is the repair rate. The transition matrix has the following interesting form (where $\mu = \rho + \tau$):

$$T_{\Delta t} = \exp -Q\Delta t = \frac{1}{\mu} \begin{bmatrix} \tau + \rho e^{-\mu\Delta t} & \rho - \rho e^{-\mu\Delta t} \\ \tau - \tau e^{-\mu\Delta t} & \rho + \tau e^{-\mu\Delta t} \end{bmatrix} \quad (5)$$

We see that the stationary state of the Markov chain is $(\frac{\tau}{\rho+\tau}, \frac{\rho}{\rho+\tau})$, which means that under the stationary distribution, the proportion of working devices is $\frac{\rho}{\rho+\tau}$.

3.2. User device choice model

We consider that a given user chooses a device randomly according to a user-specific distribution which does not directly depend on time. The probability of user u choosing device d depends only on the habits of user u (a nominal profile) and the soft failure states of the devices: if the primary device d (highest ranked in the nominal profile) is not working properly ($s_{di} < 1$), another device d' will be chosen with a higher probability — we say that the user *redirects* his/her job from d to d' . Sometimes the user does not know that device d is not in a satisfactory state, or the state is not sufficiently degraded to justify a redirection, so that the probability of the event $\{d_i = d\}$ given $s_{di} < 1$ should not be 0. We propose to use the following conditional probability, which is assumed constant over time (hence index i can be removed):

$$\mathbf{p}(d|s, u) = \frac{\pi_{ud}s_d}{\sum_{d'=1}^{N_d} \pi_{ud'}s_{d'}} \quad (6)$$

This formula can be interpreted as follows: when all the devices are working ($s_d = 1$ for all device d), user u selects the device according to his/her nominal profile given by the proportions π_{ud} . If a device d is perceived as not working ($s_d < 1$), the proportion π_{ud} is multiplied by s_d , so that the user has less chance to print on d . These modified proportions are then normalized so as to represent probabilities. It is here assumed that the smallest possible discretised state σ_L is non null, so that this normalisation is always possible.

3.3. Inference and learning

Inference The inference task consists in evaluating, for every new job i , the most probable state of the infrastructure, i.e. the mode of the distribution $p^*(s_i) = \mathbf{p}(s_i|t_{1:i}, \mathbf{u}_{1:i}, \mathbf{d}_{1:i}; \theta)$. With HMM, this filtering is solved in an exact way by the “forward” pass of the forward-backward algorithm [5]. It is not feasible with our FHMM

model due to the dimensionality of the state space. The state space for each of the components s_{di} is of cardinality L , so the state space for the whole vector is of cardinality L^{N_d} and the complexity of the algorithm, which is quadratic in the size of the state space, is thus exponential in the number of devices N_d .

We therefore need to use an approximation scheme. One possibility is to use a particle filtering algorithm, which provides an approximate solution to the exact problem; alternatively, one can use a variational approach [4] which provides an exact solution to an approximate reformulation of the problem. In both cases, the computational complexity becomes linear in the number of devices.

In the variational approach, distribution p^* is approximated by its “projection” \hat{p} in a tractable subspace \mathcal{Q} of the space of probability distributions over the whole state space. This projection is defined by

$$\hat{p} = \underset{q \in \mathcal{Q}}{\operatorname{argmin}} KL(q||p^*) \quad (7)$$

where KL denotes the Kullback-Leibler divergence between probability distributions. If \mathcal{Q} were the whole space of possible distributions, then, obviously, we would have $\hat{p} = p^*$, and the problem of finding its mode would be just as complex as above. Instead, \mathcal{Q} is chosen so that the computation of the mode of \hat{p} becomes tractable. Essentially, \mathcal{Q} is set to contain only the distributions which can be factorised, i.e. distributions q of the form

$$q((s_d)_{d=1}^{N_d}) = \prod_{d=1}^{N_d} q_d(s_d) \quad (8)$$

where each q_d is itself a probability distribution over a smaller space (of cardinality L). The mode of such a factorised distribution is obviously obtained by taking the mode of each of its components, so that the only difficulty resides in the tractability of the optimisation problem (7). Solutions to this problem are known in the case where the observations are continuous variables following Gaussian distributions: see e.g. [1] for a detailed discussion of the algorithm in that case. However, in our FHMM model, the observations are discrete and follow multinomial distributions. We developed the algorithm adapted to that case [2].

Note that the constraint (8) enforces a full factorisation in each of the devices. Less stringent constraints can be enforced by clustering together some of the components. The state space of such clustered variables is larger, but the factorisation has less components. When all the components are thus grouped into a single cluster, the optimisation problem (7) becomes unconstrained, and we are back to the untractable case where $\hat{p} = p^*$.

Learning The learning of the parameters can be done by the variational EM algorithm, which includes variational in-

ference (described above) as a sub-task. Alternatively, we propose here a heuristic which gives good results in practice, based on the interpretation of the parameters τ , ρ , π :

- We observed that the results of the inference are weakly sensitive to variations in the parameters τ and ρ . Therefore, we propose to set them manually to reasonable values, according to the administrator experience. In the example of printer devices, we can expect to observe 2 failures per year, and we can set the repair delay to say 8 days, so that $\tau \approx \frac{2}{365}$ and $\rho \approx \frac{1}{8}$. For higher discretisation levels $L > 2$, the full matrix Q has to be set by the administrator.
- The user profiles π_{ud} can be evaluated on the training dataset. We observed that the empirical proportions are rarely sufficient to get precise estimates, especially due to zeros in the data matrix (when a user u has never used device d , still the proportion π_{ud} should not be zero). To smooth the data, many methods exist, including Laplace smoothing.
- In the binary discretisation case ($L = 2$), the actual state space Σ can be set to $\{1.0, 0.1\}$, meaning that the probability to print on a malfunctioning device is approximately the original proportion divided by 10.

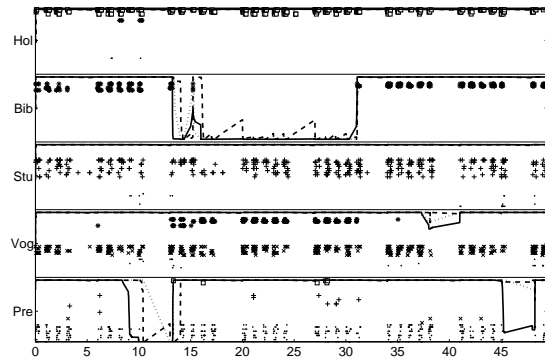


Figure 2. Inference with a binary discretisation. The points represent the jobs. The surimposed curves give $p(s_{di} = 1)$ for each printer at each time, using three different methods: plain line for exact inference; dashed line for variational inference; dotted line for exact inference with smoothing.

An illustrative example We illustrate our method on a small infrastructure (a subset of a real one) involving 30

printer and failure date	baseline		FHMM	
	delay(min)	warning(%)	delay(min)	warning(%)
Lep(07/19/2006 08:44:07)	131.2	2.0	142.3	1.5
Lep(07/19/2006 16:23:02)	76.9	2.2	70.8	1.4
Stu(08/23/2006 09:12:16)	NaN	0.4	94.4	0.1
Stu(01/17/2007 10:29:10)	54.3	0.6	42.8	0.1
Stu(01/25/2007 09:40:26)	200.4	3.3	31.4	2.8
Stu(03/30/2007 14:20:35)	331.1	0.1	90.9	0.1
Foc(08/24/2006 14:38:57)	NaN	1.1	45.9	0.1
Foc(09/15/2006 10:02:40)	NaN	0.1	339.0	4.2
dau(09/29/2006 14:51:03)	NaN	2.1	260.9	0.2
dau(02/22/2007 09:17:03)	72.8	4.2	66.5	0.6
mean	1.96	2.41	1.07	1.23
standard deviation	1.57	1.76	1.08	0.66

Table 1. FHMM vs baseline on real data. Average and standard deviation are computed on the 6 failures detected by the baseline.

users and 5 printers (Pre, Vog, Stu, Bib and Hol) in the same building, and we assume a binary discretisation of the states of the devices ($L = 2$). The results of the inference are plotted on figure 2. Each point corresponds to a job. Its shape is determined by the primary printer of the involved user. Users have been ordered so that: Pre is the primary device of users 1-10 (• shape), Vog for users 11-15 (× shape), Stu for users 16-23 (+ shape), Bib for users 24-28 (* shape) and Hol for users 29-30 (□ shape). It is thus easy to see when a redirection occurs. Given the size of the problem, exact inference was possible; two versions are given: one uses filtering (i.e. the state of a device at time t_i is estimated knowing only the observations before t_i) and the other smoothing (i.e. the device state is estimated knowing the whole history).

Printer Bib appears out-of-order between days 10 and 31. This was confirmed by the infrastructure administrator. The model correctly identified the problem because all the regular users of Bib printed on Vog (on the same floor) during that period. Printer Pre seems to have a problem on day 10: among the regular users of Pre, four decided to print on Vog instead (one floor below) and three on Stu (one floor above). The printers Vog and Stu do not have specific features that are not available on Pre, so this is likely to correspond to a real problem, although this could not be verified.

Filtering corresponds to a real situation where the algorithm is used for on-the-fly detection. Smoothing has no real application, but is given for comparison purpose: compared to it, filtering introduces a small delay in the detection of soft failures and is more noisy. This is to be expected as smoothing takes into account the future (i) to ignore some redirections which, in the filtering case, may appear as the premises of a soft failure, and (ii) conversely, to stress some redirections which, in the filtering case, need subsequent

confirmations to decide on a soft failure.

4. Experimental validation

Although the notion of soft failure is a tangible reality in a print infrastructure, there is no realistic way to systematically measure the soft failure state of a device over time. Therefore, we cannot expect a clean labeled dataset against which to evaluate our method. On the other hand, unlabeled datasets are cheap.

4.1. Methodology

Data sample We picked a real dataset of $N = 54824$ jobs recorded over a year in a print infrastructure. It defines the triplet of sequences $(\mathbf{t}, \mathbf{u}, \mathbf{d})$, each of length N , involving $N_u = 165$ users and $N_d = 17$ devices. From this data, we were able to compute the nominal profile $(\hat{\pi}_{ud})_{d=1, \dots, N_d}$ for each user u , defining $\hat{\pi}_{ud}$ as the empirical proportion of jobs associated with user u for which the device is d . In fact, at any time T within the recorded period, we compute these proportions on a subsequence of the log spanning over a pre-determined period of time before T (e.g. 4 weeks). In this way, we account for potential (slow) variations of the profiles, the only assumption being that the profiles remain quasi-constant over the chosen period instead of the whole record.

Baseline method We now define a baseline method against which to compare our method. We want to decide whether a device d is failed at a time T . We look at the users who have d as primary device. For the last $\nu = 20$ jobs before T by this group of users, we compute the weighted proportion of time they print on d (for jobs ordered by time,

the weights are $1, 2, \dots, \nu$, respectively, so as to penalise older jobs). When this proportion is above a threshold, we declare d as failed. This threshold is tuned depending on the proportion of false positive detection vs. proportion of missed failures that can be tolerated. In the experiment, we chose the threshold that maximizes the $F(2)$ -score, i.e. we weight more the false alarms compared to missed detection.

Comparison on labeled data In fact, our test dataset is not completely unlabeled: a few hard failures (ca. 10) were recorded with approximate timestamps by the infrastructure administrators, and, looking at the data on these few cases, we were able to reconstruct quite precisely when the failure occurred in each case. Table 1 shows the comparison between our method and the baseline method on 6 of these cases where the baseline method succeeded in detecting the failure. What is compared is the delay between the actual occurrence of a failure and its detection by each of the methods. The warning percentage, on the other hand, corresponds to the proportion of jobs within one month before a failure at which the device is classified as failed. The goal is to have a small delay without generating too many warnings.

4.2. Simulation

Simulated data sample Given the extreme scarcity of the labeled data, we relied on simulation to further validate our method. First, we simulated failures/repair dates for each device $d = 1, \dots, N_d$ using according to a 2-state continuous-time Markov process with infinitesimal generator $\begin{bmatrix} -\tau_0 & \tau_0 \\ \rho_0 & -\rho_0 \end{bmatrix}$ where $\tau_0 = 1/30$ (failure rate) and $\rho_0 = 1/3$ (repair rate).

Then, keeping the job time stamps $(t_i)_{i=1, \dots, N}$ and corresponding user indices $(u_i)_{i=1, \dots, N}$ of the previous sample, we generated the device indices $(d_i)_{i=1, \dots, N}$ taking into account the simulated failures. For each device d , an ordered list L_d of potential redirections from d is fixed. Then, for every job $i = 1, \dots, N$, the user u_i chooses the target device \tilde{d}_i which is the first one to be in a non failed state in the list L_d corresponding to the primary device d of u_i . The final simulated dataset is then $(t_i, u_i, \tilde{d}_i)_{i=1}^N$. We used different values for $\tau = 2/365$ and $\rho_0 = 1/14$ to illustrate the robustness of our approach on miss-specified models. For each date, the profile parameters π_{uk} were estimated on the three months before the failures.

Results A total of 160 different failures was generated using the previous method. We tested the failure prediction algorithms at 1000 different dates uniformly spaced (skipping the first month to have enough historical data). Results are shown on Table 2. We see that the method based on

method	AUC (%)	precision	recall	$F(2)$
baseline	71.2	75.0	51.3	60.9
FHMM	96.0	84.3	63.8	72.7

Table 2. FHMM vs baseline on simulated data. The AUC is the Area Under the ROC Curve.

the Factorial Hidden Markov Model is much more accurate than the baseline method. The computational cost of the proposed algorithms is similar and is mainly dominated by the computation of the user profiles.

5. Conclusion

In this study, we introduced a novel application that aims at detecting soft device failures based on the usage data in an infrastructure. This tool is of particular importance for infrastructure administrators since it can detect problems that are not detectable by the devices themselves and are often not reported by users. A version of this tool is currently integrated in the Xerox infrastructure management suite.

The probabilistic approach proved to be a flexible and powerful tool to isolate relevant information from noisy data. The use of Factorial HMMs allowed us to nicely capture our main assumption that device dynamics in the infrastructure are mutually independent. Although we are interested in (soft) failure analysis, our work departs substantially from traditional HMM-based failure analysis of machines, since our data is usage data rather than sensor data as is usually the case. This had direct consequences on our choices of the probability distributions. Sensor data from one machine teaches nothing about another machine, while here, it is the usage data of the whole infrastructure which provides information on its individual components.

References

- [1] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [2] G. Bouchard and J.-M. Andreoli. Factorial hidden markov model with discrete observations, may 2007. Patent Application filed at USPTO.
- [3] Z. Ghahramani and M. I. Jordan. Factorial hidden markov models. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 472–478. The MIT Press, 1996.
- [4] T. Jaakkola and M. Jordan. A variational approach to bayesian logistic regression problems and their extensions. In *Proceedings of the Sixth International Workshop on Artificial Intelligence and Statistics.*, 1996.
- [5] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.