

Predicting Failures with Hidden Markov Models

Felix Salfner

Department of Computer Science
Humboldt University Berlin
salfner@informatik.hu-berlin.de

Abstract. A key challenge for proactive handling of faults is the prediction of system failures. The main principle of the approach presented here is to identify and recognize patterns of errors that lead to failures. I propose the use of hidden Markov models (HMMs) as they have been successfully used in other pattern recognition tasks. The paper further motivates their use, explains how HMMs can be used to predict failures and describes the training procedure. An outlook to a more sophisticated treatment of time between events is also presented.

1 Introduction

Failure prediction is one of the key challenges that have to be mastered for a new arena of fault tolerance techniques: the proactive handling of faults. I propose the use of hidden Markov models (HMM) to approach it.

Failure prediction is about assessing the risk of failure for some time in the future. In my approach, failures are predicted by analysis of error events that have occurred in the system. As, of course, not all events that have occurred ever since can be processed, only events of a time interval called *embedding time* are used. Failure probabilities are computed not only for one point of time in the future, but for a time interval called *prediction interval*. See Figure 1.

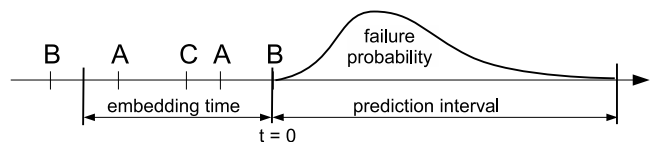


Fig. 1. The probability of upcoming failures is estimated from previous error events.

I propose the use of HMMs for failure prediction. As in many machine learning applications, the model is first built offline from previously recorded training data. After training, the model is applied to the running system in order to perform online prediction of upcoming failures.

This paper is structured as follows: First hidden Markov models (HMM) are introduced and it will be argued in Section 3 why they are suited for failure

prediction. Although prediction of failures is the second step of the machine learning approach, I will describe it first (Section 4). Having the goal of the approach in mind, the training step is explained in Section 5. Section 6 concludes the paper and provides an outlook to research challenges.

2 A Short Introduction to Hidden Markov Models

Several ways exist to define HMMs properly. This report follows the notations of Lawrence R. Rabiner as given in [1]. A hidden Markov model is an extension of a finite discrete time Markov chain (DTMC). DTMCs are defined by a set of labeled states $S = \{S_i\}$ ($1 \leq i \leq N$), a square stochastic matrix $A = [a_{ij}]$ defining transition probabilities between any pair of states, and a vector $\pi = [\pi_i]$ specifying an initial probability for each state.

Hidden Markov models extend DTMCs in that an output is produced each time the DTMC enters a state. More formally, the outputs are called observation symbols from a finite set $O = \{O_k\}$ ($1 \leq k \leq M$). The set of observation symbols is sometimes called the alphabet of the HMM. Each state S_i can output any symbol O_k based on the observation probability distribution $b_i(k)$. The output matrix $B = [b_i(k)]$ is a rectangular $N \times M$ stochastic matrix.

One may ask, why these models are called “hidden”. The name stems from the fact, that HMMs are usually used in scenarios, where only the sequence of observations is known, but not the sequence of states the model has gone through. Therefore, one can consider the states as “hidden from the observer”.

3 Why HMMs are suited for Failure Prediction

The distinction between faults, errors and failures is one of the most fundamental insights of fault tolerance research (see, e.g., [2]). From this insight results that failure prediction methods – unless they are able to perform tests inside the system – cannot operate on faults due to the faults’ property being undetected. Therefore, failure prediction can either rely on *effects* of faults or on their *detection*, which turns them into errors. Prediction methods of the first group mostly operate on continuously available measures such as memory utilization or workload to identify trends or anomalies. Methods belonging to the second group mostly only take the *time* of detection into account. The approach presented here belongs to the second group, but in addition to the time of occurrence it exploits the *type* of the error events. This might improve root cause analysis, which is necessary for automatic triggering of preventive actions against upcoming failures.

The fundamental assumption in my approach is, that the occurrence of failures can be predicted by identifying *special patterns of errors* the system is experiencing. This assumption is based on the fact, that dependencies among the components of systems exist. Due to these dependencies, an error in one component may lead to successive errors in depending components. Fault tolerant systems can handle most of these error chains but fail under some special

conditions. My assumption is, that these special conditions can be identified by *recognition* of the special error patterns.

Hidden Markov models (HMMs) have been used successfully in pattern recognition tasks such as spoken language processing or gene sequence analysis [3, 4]. They exhibit the property of being flexible and adaptive while at the same time offering structure and simplicity that allows for formal treatment as well as in-depth understanding.

The use of HMMs for failure prediction is further motivated by the following analogy: As faults are unknown and cannot be measured, they cause or “produce” an error message on their detection. This matches perfectly the notion of HMMs: There are unobservable hidden states (corresponding to faults) producing symbols corresponding to errors. To account for failures, special hidden states are introduced that do not produce error messages. See Figure 2.

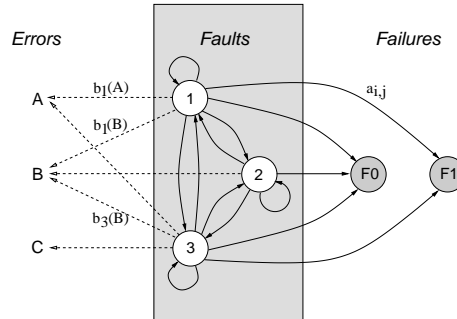


Fig. 2. Mapping of faults, errors and failures to hidden Markov models.

4 Predicting Failures with HMMs

Although failure prediction is the second phase of my overall machine learning approach, I describe it first as the preceding training step is easier to understand having the goal in sight. Training will be described in the subsequent section.

The prediction of failures comprises two steps: First the current state is estimated on the basis of the recent error events. Starting from the current state, future behavior is extrapolated and the failure probability is computed in a second step.

4.1 Estimating the current system state

The current system state is represented by $\pi^o = [\pi_i^o]$, which is the vector of probabilities π_i^o that the system is in the hidden state $S_i \in S$ after the recent error messages (observations) $O = \{O_1 O_2 \dots O_L\}$ have been encountered.

Referring to Figure 1, O would be the sequence $\{A C A B\}$. The theory of hidden Markov models provides an efficient method to compute π^o : the Viterbi algorithm. Switching to the common notation used for HMMs, the desired probabilities π_i^o at the end of the observation sequence are equivalent to:

$$\pi_i^o = \delta_L(i) = \max_{s_1, s_2, \dots, s_{L-1}} P(s_1 s_2 \cdots s_{L-1} s_L = i, O_1 O_2 \cdots O_L | \lambda) \quad (1)$$

where $\delta_L(i)$ denotes the maximum probability that during time steps $n = 1, \dots, L-1$ the model went through states s_1, s_2, \dots, s_{L-1} and that it is in state i at the last step ($s_L = i$) while observing the sequence of symbols $O_1 O_2 \cdots O_L$. Maximum probability means maximizing over all possible state sequences s_1, s_2, \dots, s_{L-1} . λ denotes the HMM parameters A, B and π . Please note that π is the initial distribution of the HMM while π^o is the probability vector of the model having observed the sequence O of error messages.

4.2 Computing failure probability

The goal of failure prediction is to compute $P_{F_i}(n)$, which is the probability that the Markov process defined by the hidden states, transition matrix A and initial probability vector π^o enters failure state S_{F_i} in at most n time steps. The probability is determined by the *first passage time distribution*. Let the random variable T be defined as $T = \min\{n \geq 0 : X_n = S_{F_i}\}$ where X_n denotes the hidden state at step n . T is called the *first passage time*. Its distribution can be computed by:

$$P_{F_i}(n) = P\{T \leq n\} \quad (2)$$

$$= \sum_{i \in S} P\{T \leq n | X_0 = i\} P\{X_0 = i\} \quad (3)$$

$P\{X_0 = i\}$ is the probability that the model is in state i at present time, which is π_i^o (as computed in the previous section). $P\{T \leq n | X_0 = i\}$ is the probability of visiting state S_{F_i} in at most n time steps starting from state i . Markov theory provides a recursive method to compute this probability (see, e.g., [5]).

The given equations provide a way to compute the cumulative probability distribution that the system evolves into a failure state S_{F_i} . Applying a dynamic programming scheme, the equations can be computed efficiently for various n . Please note that several failures can be modeled within one HMM (see Figure 2) and total failure probability is simply the sum over all P_{F_i} .

5 Training the Hidden Markov Model

HMMs are trained on the basis of a set of given observation sequences called *training sequences*. The goal of training is to find optimal HMM parameters (transition matrix A , symbol production matrix B and initial state vector π) such that the model best fits the training data. There is no known way to analytically solve the optimization problem. Therefore, methods such as gradient or

expectation-maximization techniques, e.g., the Baum-Welch algorithm [1] that iteratively improve an initialized model are used. These methods provide convergence at least to local optima.

In the case of failure prediction, observation symbols refer to error events of the system that is modeled and system failures are represented by “special” hidden states (see Figure 2). Therefore, the goal of training is to adjust the HMM parameters such that the error patterns are best represented by the model and that the model transits to a failure state each time a failure occurs in the training data.

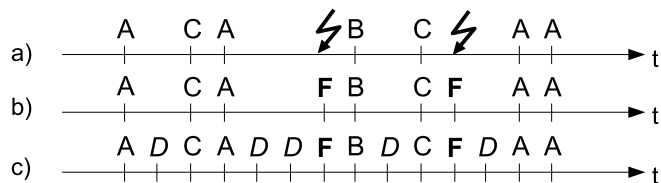


Fig. 3. Construction of training sequences.

In computer systems, training will most likely be based on logfiles containing error events (for example, error messages identified by message IDs) and on the knowledge, when failures have occurred during the logged time period. This is shown in Figure 3(a) where the occurrence of failures is indicated by flashes. Two steps are necessary to obtain training sequences for the hidden Markov model: First, failures are transformed into a special event symbol (one for each failure type) as indicated by ‘F’ in Figure 3(b). By initializing output probabilities $b_i(k)$ such that failure states are the only states that can produce their failure symbol, it can be guaranteed that the model transits to the failure state once the failure symbol occurs in the training sequence. Hence, the training algorithms do not need to be changed. The second step to obtain training sequences is devoted to the discrete time property of HMMs as presented in Section 2. Since error events do not occur equidistantly, special delay symbols ‘D’ have to be inserted to fill up the gaps between error or failure events as can be seen in Figure 3(c).

6 Conclusion and Outlook

This paper sketches how hidden Markov Models (HMMs) could be used to predict failures of computer systems. The idea is to identify suspicious patterns of error events that indicate an upcoming failure. HMMs are used as pattern recognition tool.

A machine learning approach has been proposed where the HMM is first trained offline using previously recorded logfile data, and is then used to predict failures online – while the system is running. The prediction itself comprises two steps: First the current system state is estimated from previous error events by

applying Viterbi’s algorithm. Based on the estimated state, the risk of failure is assessed by computing the first passage time distribution into a failure state.

The critical part of the approach is handling of time. The solution presented here is straightforward: Time is split into equidistant slots and every slot without error or failure is filled with a symbol indicating “silence”. A number of more sophisticated solutions has been developed. For example, output probabilities can be extended to N-dimensional continuous probability distributions. This could in the case of failure prediction be used to incorporate inter-event times as second dimension of output probabilities. Other extensions base HMMs on continuous-time Markov chains. Within this framework, inter-event durations could be incorporated directly into transition timing. I propose an additional solution that is based on the fact that events and “delays” alternate (see Figure 4). Additional to “error” nodes (E_i) that produce error event observations and to “failure” nodes (F_i), there is a set of “delay” nodes (D_i) representing inter-event times of different length. This structure offers the opportunity to stay within the efficient discrete framework presented here and the ability to represent inter-event times varying in orders of magnitude.

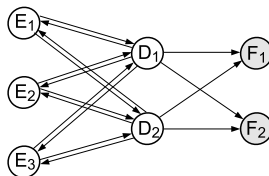


Fig. 4. Proposed model structure to account for inter-event times.

The goal of my future research is to explore the potential of the approaches. Investigations will include accuracy of failure prediction, sophisticated model initialization, root cause analysis, and model stability against changing system configurations potentially leading to online model adaption.

References

1. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE* **77** (1989) 257–286
2. Laprie, J.C., Kanoun, K.: Software reliability and system reliability. In Lyu, M.R., ed.: *Handbook of software reliability engineering*. McGraw-Hill (1996) 27–69
3. Huang, X., Acero, A., Hon, H.W.: *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Prentice Hall, Upper Saddle River, NJ, USA (2001)
4. Durbin, R., Eddy, S.R., Krogh, A., Mitchison, G.: *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge University Press, Cambridge, UK (1998)
5. Kulkarni, V.G.: *Modeling and Analysis of Stochastic Systems*. first edn. Chapman and Hall, London, UK (1995)