HAL
open science

Fault monitoring with sequential matrix factorization

Dawei Feng, Cecile Germain

To cite this version:

Dawei Feng, Cecile Germain. Fault monitoring with sequential matrix factorization. ACM Transactions on Autonomous and Adaptive Systems, 2015, 10 (3), pp.20:1–20:25. 10.1145/2797141 . hal-01176013

HAL Id: hal-01176013

https://inria.hal.science/hal-01176013v1

Submitted on 2 Oct 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Fault monitoring with sequential matrix factorization

Dawei Feng[a], Cécile Germain[b]

[a] National University of Defense Technology
[b] TAO team, INRIA & LRI, CNRS & University Paris-Sud

Preliminary version

Preamble

For real-world distributed systems, the knowledge component at the core of the MAPE-K loop has to be inferred, as it cannot be realistically assumed to be defined a priori. Accordingly, this paper considers fault monitoring as a latent factors discovery problem. In the context of end-to-end probing, the goal is to devise an efficient sampling policy that makes the best use of a constrained sampling budget.

Previous work addresses fault monitoring in a Collaborative Prediction framework, where the information is a snapshot of the probes outcomes. Here, we take into account the fact that the system dynamically evolves at various time scales. We propose and evaluate Sequential Matrix Factorization (SMF) that exploits both the recent advances in matrix factorization for the instantaneous information and a new sampling heuristics based on historical information. The effectiveness of the SMF approach is exemplified on datasets of increasing difficulty and compared with state of the art history-based or snapshot-based methods. In all cases, strong adaptivity under the specific flavor of active learning is required to unleash the full potential of coupling the most confident and the most uncertain sampling heuristics, which is the cornerstone of SMF.

1 Introduction

We depend on computer systems that are not dependable: large scale distributed systems pervade real-world Information Technology infrastructures and usage; and, decades ago, Lamport characterized such systems as those where "*the failure of a computer you didn't even know existed can render your own computer unusable*".

Computer Science research has worked on large scale fault management since long, with two main directions: discovering faults, and/or coping with them. With the advent of truly massively distributed systems with complex structures, a key change now occurred: rich monitoring information becomes available. Complete knowledge, and the very concept, of the state of a distributed system remain unreachable for fundamental reasons [17]. But, with the availability of equally massive information, estimating elements of the system state becomes a realistic goal. Specifically, some fault management goals can be re-casted as *fault inference* problems. This work targets a specific aspect of fault monitoring, fault discovery. In the following, the inference approach to fault discovery will be called *fault prediction*, because it fits with the relevant contexts in Machine Learning (supervised learning at large, and more specifically Collaborative Prediction), although fault estimation would more accurately describe the approach.

Predicting faults improves system availability and reliability by providing useful information for the next task of coping with them, as the systems are normally highly redundant and heavily supervised. Often, alternatives to the faulty services can be proposed [41, 43]; in these cases, a well organized fault management system will conceal the hardware and software dysfunctions and will provide a transparent service that is a crucial ingredient of Quality of Experience. On the other hand, irrecuperable faults must be signaled as fast as possible to the human of automatic supervision. Overall, this amounts to re-evaluate the role of monitoring in fault management, and to consider fault prediction as an *inference in the space-time domain*.

Autonomous Computing (AC) provides a conceptual framework for designing fault management for these monitoring-equipped systems. Its so-called MAPE-K loop is organized around a Knowledge component. To set up fault prediction as a realistic objective in an AC approach, the first question is which kind of knowledge is actually reachable. In a fault diagnosis approach, the knowledge component includes a detailed internal model of the system that can be exploited to pinpoint the faulty components. The root causes of the faults can be revealed through various techniques [46] like statistical inference [36, 4, 9], log-based causality analysis [35, 14, 26] or deterministic replay[16, 27, 20]. Fault diagnosis can be seen as the process of recognizing the most likely explanation for the symptoms based on some causal and effect models among the propositions of interest in the problem domain.

While diagnosis maximizes the usefulness of monitoring data, it faces some potentially significant practical limitations. The first one is simply scalability: diagnosis is NP-hard [36]. More profoundly, assuming knowledge of a decent model of the system internals might prove unrealistic (the "*computer you didn't even know existed*" nicely summarizes this feature). As a consequence, this work formulates the fault prediction problem in the context of end-to-end monitoring. The overall infrastructure is a black box, with no a priori knowledge of its structure. End-to-end probes are designed to test a functional property of this blackbox. Then, fault prediction involves a classification problem: from a selection of the probes (the training set), infer the outcomes of the other probes.

In many cases, the probes can be meaningfully replicated in the system. For instance, in the example that will be further described in section 5, the functionality is related to file access, and the probes are launched from the computing nodes to the storage nodes. Then, the replication takes a *matrix* form: the endpoints are the row- or column- entities, and the probes outcomes are the entries in this matrix. Formally, fault prediction becomes a *matrix completion* problem. The benefit with respect to a fully unstructured setting is that the matrix hypothesis grounds a Collaborative Filtering technique that is more powerful than behavioral clustering [32], [45]: in a nutshell, although the exact causes of failures might remain elusive, causality can be precisely modeled as the rank of the matrix, to be inferred, while clustering is bound to phenomenology and a-posteriori analysis of exemplars or centroids.

Our previous work [13], inspired by [37], addressed fault prediction in a classical Collaborative Prediction framework, as a purely spatial problem where the matrix is assumed to be a snapshot of the probes outcomes. Here, we take into account the fact that the system dynamically evolves at various time scales. Addressing this issue brings us closer to a model consistent with the practitioners expectations, but turns out to be significantly more difficult than the previous and more idealized setting. Not surprisingly, we cope with this difficulty by exploiting the dynamic setting for enriching the snapshot with time-related information: *sequential monitoring* deals with a sequence of partially observed matrices and makes prediction using information both from the current and previous time windows.

Within this framework, in order to be realistic, inference has to address two specific difficulties. Firstly, strongly imbalanced distributions must be assumed, as faults are hopefully much less represented that nominal behavior; this belongs to the spatial aspect of inference. Second, in the time domain, one cannot assume that measurements could be kept fully up-to-date, as these systems are highly dynamic environments.

Fortunately, the same adaptivity strategy proved successful in various contexts to address both imbalanced distributions and noisy information: active learning iteratively selects most-informative samples in order to best improve the prediction accuracy. On the other hand, and always with realism in mind, active learning has the drawbacks to slow down the fault discovery process, as it requires to build the input incrementally, and to make it more complicated, thus more fault-prone itself. A transversal goal of this work is thus to evaluate the specific contribution of the active learning ingredient in the fault inference methods that we propose.

Our experimental validation dataset comes from the European Grid Initiative (EGI). Grids tend to be regarded as somehow outdated, thus a few words about the relevance of the dataset might be necessary. The specific grid technologies of the 2000's have of course been superseded by cloud-related ones. However, the essential paradigm of grid is organized sharing: safely and fairly federating hardware, software and data resources from multiple independent providers. Thus grids exemplify both the physical problems of worldwide scale systems, and the additional and major issues associated with a multi-owned multi-operated system, that are equally present in federated clouds.

The main contributions of this paper are twofold.

- A detailed analysis of the algorithmic alternatives, that contributes to disentangle the components of performance and substantiate the claim that sequential patterns should be exploited.
- The **SMF** (Sequential Matrix Factorization) algorithm, and its active learning version, **SMFA** (Sequential Matrix Factorization with active learning), that efficiently combines the spatial and temporal information sources. Its major strength is to balance exploration and exploitation in a way that formalizes and exploits the multi-scale intuition of the practitioners.

The paper is organized as follows. Section 2 presents the empirical motivation and the formal description of the problem. For clarity, we have structured the rest of the paper by presenting first all the algorithms that are considered for evaluation, including our adaptations of classical ones as well as the proposed SMF. Thus, the motivation for SMF will become more evident only with the experimental sections. Section 3 walks through the various ways to organize spatial and temporal information. Section 4 describes SMF and its active learning version, SMFA. The next three sections present a detailed experimental evaluation, where the vanilla algorithms are combined in various ways with agnostic optimizations (smoothing) and information-oriented ones (strategies for active learning). Section 5 describes the experimental setting, and sections 6 and 7 the results on the EGI dataset, before the usual conclusion.

2 Contexts and motivations

2.1 Matrix factorization for fault prediction

Consider the simple setting where a partial snapshot of the system is available through end-to-end probes (e.g. a *ping*). Assume that we have M sources and N targets. A *probe selection* method defines which ones of the possible MN probes are actually launched. Each individual result is binary: *positive* means that the probe *failed*, i.e. a fault occurred; *negative*, the probe succeeded. Let X be the sparse $M \times N$ binary matrix of the outcomes of the selected probes.

Then, the inference task falls into the general category of *matrix completion*: given a sparse matrix X, find a full, matrix Y of the same size that approximates X, i.e. that minimizes some measure of discrepancy between X and Y. When Y is required to be equal to X on the known entries, the problem is termed *exact completion*, and *approximate* otherwise.

With such a general setting, the problem is hopelessly ill-defined: in order to guess the missing entries, some assumptions have to be made about the matrix to recover Y. A natural one is to look for *low-rank* matrices, amounting to assume that a small number of hidden and partially shared factors (*latent* factors) affect the matrix entries.

The existence and unicity of a solution of exact matrix completion is a complicated problem (see e.g. [8]). Anyway, this formulation does not look not very helpful, as rank minimization for completion is NP-hard and not feasible practically even for small sizes. However, it has paved the way for efficient algorithms, both for exact [8, 33] and approximate [39] completion. The main insight is to replace the rank by the trace (or nuclear) norm in the regularization term.

This heuristic grounds the theories of both exact completion of real-value matrices [8, 33] and approximate completion of binary ones [39]. Although they apparently address the same issue, they do not coincide in their assumptions and their goals (for an in-depth discussion of the differences, see e.g. [10]). The real-valued approach is related to compressed sensing; its goal is to provide sufficient conditions for exact recovery (with high probability). Informally the sample size is related to the "spikiness" of the target matrix: at the extreme, a matrix with only one positive entry needs on average to be fully sampled. Non-spikiness can be formally quantified by the incoherence indicator [7], an intrinsic (isometry-invariant) quantity related to the spread of the target values; the less coherence in rows or columns, the less entries must be sampled.

Approximate completion for binary matrices adopts the point of view of statistical learning theory. Hinge loss as the discrepancy measure yields the **Maximum Margin Matrix Factorization (MMMF)** algorithm [39] with the objective function:

$$||Y||_{\Sigma} + C\mathcal{L}_h(X(S), Y(S)), \tag{1}$$

where S is the set of known entries, $||.||_{\Sigma}$ is the nuclear norm and $\mathcal{L}_h(A, B)$ is the hinge loss between A and B. As the minimization procedure of equation (1) produces a real-valued matrix, a decision threshold (the classification common practice) gives the final binary matrix. This is a model-free approach, where the goal is to bound the generalization error. More precisely, [39] bounds the average error across all matrix entries in terms of the margin error on the observed entries: if the recovered matrix Y has low nuclear norm and matches the observed entries by a significant margin, then the error on unobserved entries is guaranteed to be small.

As previously mentioned, the optimization problem (1) is convex, thus is guaranteed to define a global optimal solution. More generally, as the nuclear norm is a convex function, and the matrices of bounded nuclear norm are a convex set, any convex loss function provides a convex optimization problem. However, each observed entry acts as a constraint during the minimization process, and the computational complexity increases drastically with the number of involved constraints. Therefore, in the fault prediction case, the external cost of each probe as a monitoring overhead doubles as an internal computational practical limit.

The soft-margin approach of (1) is more adapted to the problem at hand than exact matrix completion. Firstly, there is significant empirical evidence [10] that the standard approach for exact matrix completion behaves poorly in the discrete case, due to hard scale bounds resulting

of constraining to the exact values; the thresholding of the continuous discriminant function learned in the soft-margin approach might be more informative. Secondly, the fault matrices can be expected to present isolated faults on a large background of healthy entries, in other words to be spiky. Thus, the theoretical evaluations of the sample size derived from the incoherence indicator would not provide any effective guideline, as they would be quite large.

In classical collaborative filtering, X is given (e.g. consumer ratings); matrix completion theory assumes a priori defined uniform sampling schemes, which make theoretical analysis tractable. The goal of this paper is very different: the contribution of SMF and SMFA is on the focus on smarter, data-driven sampling schemes for the case where historical information is available.

2.2 Motivation

[37] proposed a method to handle fault inference based on a collaborative prediction approach, further analyzed in [13]. Although this method significantly reduces the number of required probes for acquiring an accurate view of the system, it is somehow static. More exactly, its only input is a snapshot from (assumed) simultaneous probes.

This setting has two drawbacks. Firstly, fault behavior is multi scale in time: beyond the stable system components with consistent status over time, there are other ones which status may fluctuate intensely at peak time and remain stable at off-peak time. Second, transient faults are systematically observed: transients are faults that get on and off at high frequency and should be considered as noise; practitioners do not have a clear explanation for them, and they might as well be produced by flaws in the monitoring software itself. Of course, the problem is to disentangle them from real, but short-lived faults.

A further motivation is to explore the possibility of getting rid of adaptivity. [37] integrated active learning with MMMF (min-margin heuristic) and [13] have shown that active learning was a required ingredient in the most difficult and realistic case on a real-world fault prediction example. On the other hand, active learning is somehow inconvenient: because the probe selection is adaptive, it requires a feedback loop, an interface between online analysis and monitoring, and thus a more complicated software than with a pre-determined setting. At grid or cloud scale, any unnecessary source of complexity should be eliminated. Thus, we explore the hypothesis that the past could statically provide information equivalent to the one obtained by adaptively querying the present.

2.3 Categorization

To address these issues, this paper reconsiders the collaborative monitoring task with a series of time-based inputs and exerts the collaborative prediction sequentially. Then, two types of inputs become available: *spatial* and *temporal*. For a given monitoring component, the spatial information is the current status of other components in the system, while the temporal information is its own historical information from the past. Here the components are restricted to end-to-end probes.

Depending on which information a method uses as input, we further divide the methods into three categories.

- The *pure spatial* methods only use information available from the current timestamp; they will be exemplified by **MMMF** presented in section 2.1.
- The *pure temporal* methods where the inputs are the entry-wise historical sequences considered independently; they predict entry-wise too, based on time series methods, e.g. Moving Average. They will be exemplified by the **EWM** method presented in section 3.1.
- The *integrated* methods, which exploit both informations as inputs. In the classical Collaborative Filtering framework, the so-called *Tensor* method exemplifies this approach and

is discussed in section 3.3. Our algorithms, **SMF** and **SMFA** (section 4), fall into this category. The *collapsed* methods are a subcase. They straightforwardly transfer those employed in static matrix based prediction to the sequential situation and are the simplest approach to integration: first, the temporal information is exploited to build a summary, then a purely spatial method uncovers its hidden structure. They will be exemplified by the **SSVD** method presented in section 3.2.

2.4 Problem statement

We now introduce the notations and formalize the previous concepts. Let

$$X_t \in B^{M\times N} \text{ be the partially observed matrix at time } t,$$
$$\hat{Y}_t \in R^{M\times N} \text{ be the result of a prediction algorithm,}$$
$$Y_t \in B^{M\times N} \text{ be the thresholded binary version of } \hat{Y}_t.$$

With threshold ρ, $Y_T(i,j)$ is defined by:

$$Y_T(i,j) = \begin{cases} -1 \;\; if \quad \hat{Y}_T(i,j) \leq \rho, \\ 1 \;\; \text{otherwise.} \end{cases}$$

With $B = \{-1, 1\}$, the binarization threshold ρ is set to 0.

We define the task of sequential matrix prediction as: given a series of partially observed matrices $(X_1, \ldots, X_t)$, predict the fully estimated matrix Y_t. Figure 1 illustrates the sequential process: at each time step t, a matrix Y_t is estimated from the observation sequence $(X_1, \ldots, X_t)$ and the sequence of the past estimates $(Y_1, \ldots, Y_{t-1})$.

With this formalism, temporal methods, that only rely on the entry-wise historical information, try to complete the prediction of $Y_T(i,j)$ only based on information of entry (i,j), which could be only the $X_t(i,j)$, $t = 1, \ldots, T$ or include $Y_{t-1}(i,j)$, $t = 2, \ldots, T$.

In spatial methods, where only information of the current time window is available, the prediction of $Y_T(i,j)$ is completed only using knowledge captured in X_T.

Here, it must be clear how much *prediction* is a misnomer, as there is no concept of history. *Inference* would convey the correct semantics, but usage has settled for prediction.

The integrated methods utilize both spatial and temporal information to make a prediction of Y_T. Precisely, to produce a prediction of $Y_T(i,j)$, all X_t for $t = 1, \ldots, T$ and Y_{t-1} for $t = 2, \ldots, T$ are exploited.

3 Background

This section first discusses the options for temporal methods, in relation with their intended application, fault prediction at large scale; then we present a collapsed approach, and we briefly survey the tensor one.

3.1 Temporal methods

Three classical methods for embedding sequential information are auto regressive moving average (ARMA), Hidden Markov Models (HMM) and exponential weighted moving average (EWM). In our case, ARMA or HMM would need to learn a set of model parameters and update them for each entry in the target (monitored) matrix. The total cost for ARMA is $O(m^3LMN)$ where L is the length of the time series sample, $m = \max(p, q+1)$ with p the window size of AR and q the window size of MA. The total cost for HMM is $O(S^2L^2MN)$, where S is the number of HMM states. For EWM the cost is $O(LMN)$. Besides selecting a baseline, the reason to consider EWM instead of ARMA is to evaluate the actual performance of a method with really fast execution time.

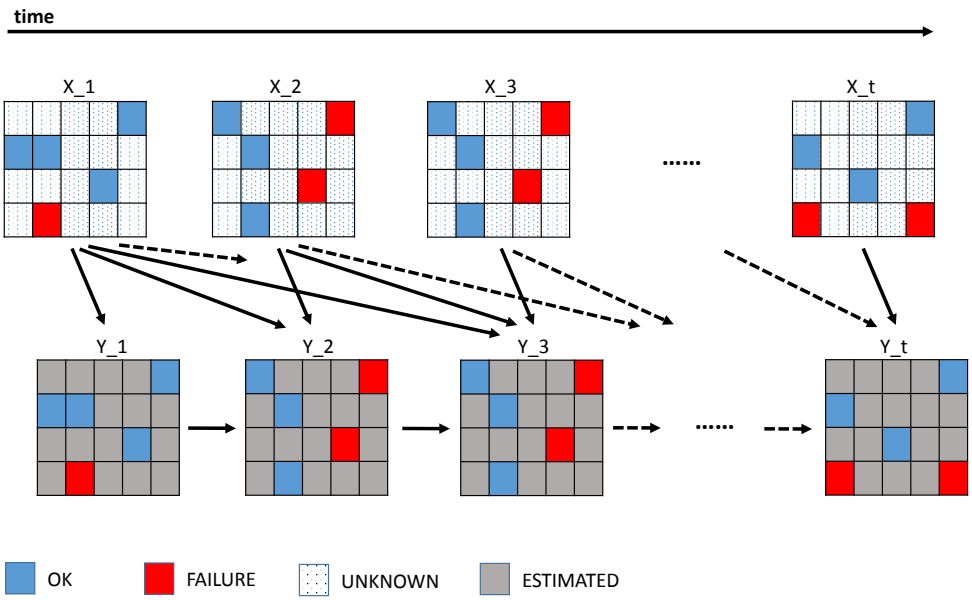


Figure 1: Problem description

Unlike simple moving average where equal weights are assigned to the past observations, exponential weighted moving average gives exponentially decreasing weights to data over time.

In the sequential monitoring task, at a time step T we want to predict for a given matrix its current status based on a series of historical observations. This can be completed directly using the EWM in an entry-wise manner:

$$\hat{Y}_T(i,j) = \begin{cases} X_1(i,j), & T = 1, \\ \theta X_T(i,j) + (1-\theta)\hat{Y}_{T-1}(i,j), & T > 1 \end{cases} . \tag{2}$$

Note that in equation 2, $\hat{Y}_T(i,j)$ is estimated based on information only from the entry's past observation sequence $X_t(i,j), t = 1, ...T$, making it a pure temporal method. For simplicity, in the following we keep the EWM name for the point wise application of EWM to the matrix entries.

Steps of using EWM to estimate Y_T are given in algorithm 1. The last L samples $X_{T-L+1}, ..., X_T$ are used as input for EWM. However, it is well possible that all of these are missing, that is no probe was launched on this particular (i,j) pair. In this case, the missing entry in $\hat{Y}_T$ is filled with the corresponding entry in $\hat{Y}_{T-1}$, as the best available estimate. For ensuring a starting point, at the initial stage, Y_{init} is computed as the exponential weighted moving average of a sequence of fully observed matrix with length L, i.e. $X_1, X_2, ..., X_L$. Because $X_1, X_2, ..., X_L$ are fully observed matrices, no missing entry exists in Y_{init}.

3.2 Collapsed methods

The idea behind collapsed methods is to exploit a full matrix built by a temporal method: predictions are produced based on a full matrix X'_t. Going back to the essential intuition that we

Algorithm 1: EWM, exponential weighted moving average.

Input: N, number of random samples;
Y_{init}, initial value for each entry;
ρ, threshold for binarization;
$X_{l,l=t-L,\ldots,t-1}$, history sample sequence;
Output: Full binary-valued matrix Y_t
Initialize: $X_t \leftarrow 0$, init random sample heuristic h_2

```
S_r ← Sample(h_2, N) /*Select N random sample indexes*/;
X_t(S_r) ← QueryLabels(S_r) /*Query the true labels for entries in S_r*/;
Ŷ_t ← θX_t + (1 − θ)Ŷ_{t−1} ;
I ← findMissingEntries(Ŷ_t) ;
for i ∈ I do
    Ŷ_t(i) ← Y_init(i)
Y_t ← binarization(Ŷ_t, ρ) /*Turn the real-valued Y_t into Binary matrix*/;
return Y_t
```

are looking for low-rank approximation, dimension reduction methods are then employed on X'_t for matrix factorization. Thus, in this section, we consider the adaptation of Singular Value Decomposition to the sequential case under the name of sequential singular value decomposition, **SSVD**.

The rank-R SVD approximation of a matrix X is given by:

$$X \approx U_R \Sigma_R V_R^T \approx \sum_{k=1}^{R} \sigma_k u_k v_k^T, \tag{3}$$

where V^T means the transpose of V.

Algorithm-2 describes how **SSVD** predict Y_t. In the first step, the missing values in X_t are imputed using *ImputeMatrix* (algorithm 3): an EWM imputation is implemented in *ImputeMatrix* by replacing the missing entries with an exponential weighted moving average of its past values. The second step of SSVD employs a SVD decomposition on the imputed matrix; then the top R-rank approximation is binarized and the result returned as the SSVD estimated matrix.

Another possible collapsed method would build a partial matrix X'_t from the past, but including only the actually observed past entries, and then perform matrix completion through e.g. MMMF on X'_t. However, X'_t would be not so sparse, and the computational complexity would become prohibitive as explained in 2.1.

3.3 Tensor factorization

A number of approaches have been proposed for the sequential matrix completion problem based on tensor factorization. For the recommendation application, [24] separates the sequential data into several coarse time domains. It then assumes a static group-level rating distribution and a slightly drifting individual user interests across the time domains. A cross-domain CF framework is used to share the static group-level rating matrix, together with a Bayesian latent factor model for capturing the drifting behavior of individual user. The inference model is learned using Gibbs sampling. This method is suitable for modeling relative long term (coarse time domains) user interests, however, not appropriate for capturing system transients. A user's interest will certainly last for a relative long time, but a component's status in a complex system may fluctuate frequently.

[25] extends the low-rank matrix completion to the tensor case by proposing the trace norm for tensors. As in the matrix completion case, the tensor completion is formulated as a convex optimization problem, and solved by three heuristic methods proposed by the user. A recent

Algorithm 2: SSVD, sequential R-rank SVD approximation

Input: N, number of random samples;
R, # of top singular components to select;
ρ, threshold for binarization;
l, # of past observations used for imputing missing entries;
Output: Full binary-valued matrix Y'_t
Initialize: $X_t \leftarrow 0$

$S_r \leftarrow Sample(h_2, N)$ /*Select N random sample indexes*/;
$X_t(S_r) \leftarrow QueryLabels(S_r)$ /*Query the true labels for entries in S_r*/;
$X'_t \leftarrow ImputeMatrix(X_t, X_{t-l,\ldots,t-1})$ /*Impute missing entries in X_t;
$[U, \Sigma, V^t] = SVD(X'_t)$ /*SVD decomposition of X'_T*/ ;
$Y_t \leftarrow U_R \Sigma_R V_R^t$ /*Top R-rank approximation*/ ;
$Y'_t \leftarrow binarization(Y_t, \rho)$ /*Turn the real-valued Y_t into Binary matrix*/;
return Y'_t

Algorithm 3: ImputeMatrix, impute missing entries in a matrix

Input: X_t, matrix with missing entries ;
$X_{t-l,\ldots,t-1}$, matrix sequence for imputing missing entries;
Output: X'_t, imputed matrix

$X' \leftarrow X$;
$I \leftarrow$ findMissingEntries(X_t) /*find missing entries in X_t*/ ;
for $i \in I$ **do**
  $X'(i) \leftarrow \sum_{k=1}^{l} (1-\theta)^{l-k} X(i)$ /*Impute missing entries via EWM*/
return X'

work [22] concerning sequential active matrix and tensor completion employs adaptive sampling schemes to obtain guarantee of strong performance for the low-rank matrix and tensor completion problem. Entries which are informative for learning the column space of the matrix are identified (tensor) through an adaptivity exploitation. Theoretical results of the sufficient number of adaptively selected samples for an exact recovery are given both for the matrix and tensor case.

Despite these extensive alternatives, here we test more generic tensor factorization methods since they are easy to implement and can serve as a baseline for the tensor based approaches. In the following, we first go through the basics of tensor factorization, and then discuss its applicability for our problem.

A tensor is a multidimensional array. A N-way (or Nth-order) tensor can be described as the product of N vector spaces. This decomposition can be used to reveal underlying linear structures in the data, and has applications like noise reduction or data compression. Generally speaking, two particular tensor decompositions are widely discussed: CANDECOMP/PARAFAC (CAPA for short in the following) and Tucker [21]. CAPA decomposes a tensor as a sum of rank-one tensors, and Tucker decomposes a tensor into a set of matrices and one small core tensor.

The benefits brought by not collapsing data into a flat matrix but keeping its natural high dimensional structure are twofold: firstly, the underlying patterns in multi-way datasets are preserved, as collapsing along any dimension loses information in that dimension. Secondly, CAPA yields a highly interpretable factorization that includes a time dimension, and patterns in the time dimension can be extracted out directly. Unlike matrix based prediction which is limited to predict for a single time step, CAPA can be used in both single step and periodic temporal prediction problems.

Compared to Tucker decomposition, the CAPA model is reputed to be more advantageous in terms of interpretability, uniqueness of solution and determination of parameters [6]. A CAPA

mode-3 decomposition can be expressed as either a sum of rank-one tensors (each of which an outer product of vectors a_r, b_r, c_r and a weight λ_r) or factor matrices:

$$Z \approx \sum_{r=1}^{R} \lambda_r (a_r \circ b_r \circ c_r) \equiv [\lambda; A, B, C]$$

where Z represents the raw data tensor and R specifies the number of rank-one components. $a \circ b$ means the outer product of a and b. For sequential monitoring, we have a third-order tensor $Z_t = (X_1, \ldots, X_t),\ Z \in R^{I,J,K}$ at time t, and the CAPA tensor factorization directly decomposes Z_t into a set of rank-one components. Then the missing entries in Z_t can be recovered by making an outer product of the first R rank-one components.

In principle, tensor factorization could be able to deal with sequential matrix factorization in a promising way. Because information along the temporal dimension is processed directly without any collapsing, the temporal transition can be preserved in the factorization. However, tensor factorization without regularization can be seen as a simple linear regression along each dimension, thus only those principally important factors are kept in the result. Moreover, entries are equally weighted, and in a dynamic setting, we care more about the most recent information than the far past one.

4 Sequential matrix factorization

4.1 The SMF algorithm

As mentioned before, there are two types of information available for sequential monitoring: spatial and temporal information. The spatial information can be thoroughly exploited by a collaborative prediction method like MMMF, while on the other hand, the temporal information which concerns the dynamics of the entries provides extra opportunity for improving algorithm performance. At each time step t we have a sequence of history predictions $Y_1, ..., Y_{t-1}$. The confidence in these predictions can be expressed by the distance of each predicted value to the separation hyper-plane. Thus two types of predictions emerge: those predictions close to the separation plane and those far from the separation plane. We call the former ones the *most uncertain* prediction set and the latter ones the *most confident* prediction set. From the system point of view, the most uncertain predictions are related to those components with short term status like the transient faults and the most confident predictions are related to those components with relatively long term stable status.

In this section, we propose an algorithm, i.e. sequential matrix factorization (SMF), to capture both the long term and short term status behavior by utilizing the spatial information as in MMMF, and exploring the most uncertain and most confident heuristic concealed in the temporal information meanwhile. In the following, S_u, S_c and S_r are index sets of matrix X, for denoting the most uncertain prediction set, most confident prediction set and a random sample set, respectively. The observed set (labels queried at time t) is $S_u \cup S_r$. All three sets depend on t, but we dropped the unnecessary supplementary indices.

With squared-error loss, the objective function of MMMF would translate to:

$$\arg\min_{Y} ||Y_t||_\Sigma + C||Y_t(S_r) - X_t(S_r)||_2^2 \tag{4}$$

where C is a regularization term. The objective function is composed of two terms, where the first one is the trace norm of the estimated matrix Y_t and the second term is the discrepancy between estimation and observation. In the following we will develop the objective function of SMF by adding the *most uncertain* and the *most confident* information to equation 4.

First we consider the most uncertain information. The most uncertain prediction set S_u (entries with small margin to the classification hyper-plane) can be derived from Y_{t-1} and their labels at time t can be queried from the system. Hence, the ground truth of those most uncertain predictions in Y_{t-1} is available in the sample set X_t. We denote this as $X_t(S_u)$.

The second information, the *most confident* predictions is concealed in the history estimation. For these most confident entries, instead of sampling their true labels at time t, their previous predictions can be used directly in the next run. More exactly, in SMF we choose those most confident predictions from Y_{t-1} and assume their states remain unchanged at time t with a confidence level γ.

We compute γ in terms of the overall difference between Y_{t-1} and X_t (i.e. the difference between last estimation and current observation) on the *observed* entries in X_t. Any classification criteria like accuracy, true positive rate (TPR) or FSCORE can be used for measuring this discrepancy. Since in X_t the observed set is $S_u \cup S_r$, we therefore compute γ as the difference between $X_t(S_u \cup S_r)$ and $Y_{t-1}(S_u \cup S_r)$ as follows:

$$\gamma = \text{TPR}(Y_{t-1}(S_u \cup S_r), X_t(S_u \cup S_r)), \tag{5}$$

where $\text{TPR}(A, B)$ is the true positive rate of A according to the ground truth set B. In the prediction, γ is used as an adaptive cost ratio which adjusts the weight (penalty) of the heuristic information in the objective function (similar to the coefficient C in equation 4). The reason we choose *TPR* as the penalty lies in the fact that in distributed system monitoring successfully discovering a failure comes more important than alerting one incorrectly (see section 5.3).

In addition to the most uncertain set S_u and most confident set S_c, we keep the random set S_r in the objective function of SMF. The random sample set S_r serves as a term for avoiding over-fitting the history information, as sudden change between past estimation and current observation might occur.

To sum up, SMF has the following objective function:

$$||Y_t||_\Sigma + C||Y_t(S) - X_t(S)||_2^2 + C\gamma||Y_t(S_c) - Y_{t-1}(S_c)||_2^2 \tag{6}$$

where $S = S_u \cup S_r$ is the sample set which labels are queried at time t and S_c is the most confident prediction set that we borrow from $t-1$. Thus the difference between equation 4 and 6 is exhibited by the selection of S_u and the presence of S_c. Like in equation 4 this function is convex, thus can be directly minimized, with a a SDP solver. The complexity issue is the same as explained in section 2.1: the number of samples is the controlling factor.

Figure 2 illustrates the selection process. The most uncertain and most confident predictions are selected from Y_{t-1}, where labels of the former set are further queried at time t, and labels of the latter set are the corresponding estimation values in the last run.

Algorithm 4 describes the pseudocode of SMF. At the beginning, the sample set S of X_t is generated by a combination of selecting most uncertain predictions from Y_{t-1} and a random sampling (line 1 to 3). Then the true labels of S are queried from the system and are used as ground truth for measuring the discrepancy between Y_{t-1} and X_t (line 4, 5). The most confident predictions in Y_{t-1} are selected in the following step and used as input for the estimation. In the final step Y_t is derived by finding an estimation which minimizes equation 6.

4.2 Sequential matrix factorization with active sampling, SMFA

In active matrix factorization [37, 13], the prediction performance is improved by selecting the sample entries in X_t actively and iteratively, using the most uncertain heuristic from the very last prediction until the maximum allowed number of samples is reached. The key idea is that, with the progress of each iteration, confidence in the estimation increases simultaneously.

In SMF, sample entries are selected under three policies: random, most uncertain and most confident. The latter two strategies rely on information from the last prediction Y_{t-1}. The selection of active samples is complete all at once in SMF and no further actions can be taken given its first estimation of Y. **Sequential matrix factorization with active sampling (SMFA)** builds a sequence of estimators $Y_t^i, i = 2, 3, ...$ that iteratively benefits from the estimation process to refine the definition of both the most uncertain and most confident predictions.

Algorithm 5 describes the steps of SMFA. We denote the estimation matrix at the ith iteration of time t as Y_t^i. At the beginning, SMF is used to give an initial estimation Y_t^0 from Y_{t-1} (line

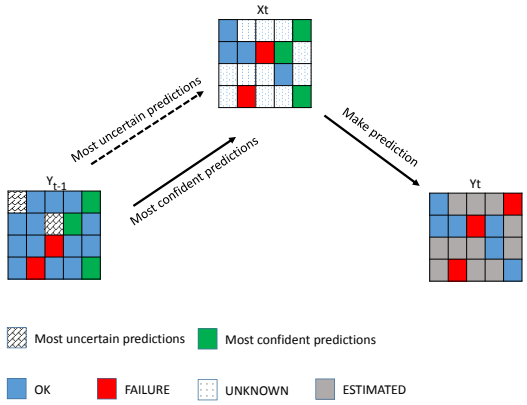


Figure 2: Illustration of heuristics in SMF

Algorithm 4: SMF, Sequential Matrix Factorization

Input: $\mathbf{Y_{t-1}}$, last prediction;
$\mathbf{N_u}$, number of most uncertain samples from Y_{t-1};
$\mathbf{N_c}$, number of most confident samples from Y_{t-1};
$\mathbf{N_r}$, number of random samples;
$\mathbf{C}$, slack penalty.
Output: Full real-valued matrix $\mathbf{Y_t}$
Initialize: $Inith_1, h_2, h_3$, /*Initialize the most uncertain, most confident and random sampling heuristic, respectively*/;

1 $S_u \leftarrow Sample(h_1, N_u, Y_{t-1})$ /*select N_u most uncertain sample indexes from Y_{t-1}*/;
2 $S_r \leftarrow Sample(h_2, N_r)$, /*select N_r random sample indexes*/;
3 $S \leftarrow S_u \cup S_r$;
4 $X_t(S) \leftarrow QueryLabels(S)$, /*query the true label for entries in S*/ ;
5 $\gamma \leftarrow Precision(X_t(S), Y_{t-1}(S))$ /*given $X_t(S)$ (true labels for entries in S), compute the precision of $Y_{t-1}(S)$*/;
6 $S_c \leftarrow Sample(h_3, N_c, Y_{t-1})$, /*select N_c most confident samples from Y_{t-1}*/;
7 $Y_t \leftarrow \arg\min_Y ||Y_t||_\Sigma + C||Y_t(S) - X_t(S)||_2^2 + C\gamma||Y_t(S_c) - Y_{t-1}(S_c)||_2^2$ /*find an estimation that minimizes the objective function*/;
8 **return** Y_t

4), then an iterative estimation is employed on the prediction sequence $Y_t^i, i = 1, 2, ...$ until the maximum number of samples is reached (line 5 to 9). Active sample selection is engaged each time the SMF algorithm selects the most uncertain and most confident predictions from the last estimation.

Formally, neither the SMF nor the SMFA algorithms can be considered as bandit methods: they exploit only the previous estimate and the current sample, thus there is no concept corresponding to the empirical mean of the reward that would be used for selecting an arm; in our case, an arm would be a matrix entry or a subset of matrix entries. There is a good reason not to consider a straightforward bandit framework, with the above-defined arms: the corresponding action space would be exceedingly large with respect to the hypothesis of a stable distribution

Algorithm 5: SMFA, Sequential Matrix Factorization with active sampling

Input: N, max # of new samples;
Y_{t-1}, last prediction;
P_0, initial sample rate for the 1st prediction;
P_a, active sample rate at each iteration;
ρ, ratio of random samples and most uncertain samples for P_a;
C, slack penalty.
Output: Full real-valued matrix Y_t
initialize: $Init(N_c)$. /*Initialize the number of most confident samples to select in each iteration*/;

```
i = 0 /*current iteration index*/ ;
n = N × P_0 /*current number of new samples*/ ;
[N_u, N_r] ← computeSampleSize(n, ρ) /*Get random and most uncertain sample size for the initial prediction*/;
Y_t^i ← SMF(Y_{t-1}, N_u, N_c, N_r, C);
while (n < N) do
    [N_u, N_r] ← computeSampleSize(N × P_a, ρ) /*Get random and most uncertain sample size according to ρ and P_a*/;
    Y_t^{i+1} ← SMF(Y_{t_i}, N_u, N_c, N_r, C);
    n = n + N_u + N_c + N_r ;
    i = i + 1 ;
Y_t = Y_t^i ;
return Y_t
```

(see section 6.1 for an example). However, the self-calibration of the γ parameter that controls the respective weights of the past information and newly acquired one in the objective function has some analogy with the adaptation of the parameter of the bandit ϵ-greedy strategy described in [40]. An approach for a more aggressive sample balancing is presented in the conclusion.

4.3 Smoothing the results

Although one of the key features in SMF or SMFA is to preserve the continuity of predictions between consecutive time windows, extra smoothing of the outputs can be considered. Smoothing the prediction sequence with e.g. EWM works as follows:

$$Y'_t(i,j) = \begin{cases} Y_k(i,j), & k = t-l+1, \\ \theta Y_t(i,j) + (1-\theta)\hat{Y}_{k-1}(i,j), & k = t-l+2, ..., t \end{cases} . \tag{7}$$

where $\theta \in (0,1)$ is an user defined damping factor, and l is the lag window length.

4.4 Methods summary

Table 1 summarizes the methods introduced in the two previous sections, with their inputs, outputs and related parameters. For a given method H, its smoothed version is noted H^* (e.g. the smoothed version of SMF is noted SMF^* in later section).

5 Experimental setting

5.1 The source

The European Grid Infrastructure (EGI) enables access to computing resources for European researchers from all fields of science, including high energy physics, humanities, biology and more.

Table 1: Methods summary

	Input Data	Output Data	Parameters
EWM	$X_{t-L+1}, ..., X_{t-1}, X_t$	Y_t	N, # of samples; θ, damping factor; L, lag window length.
SSVD	X_t	Y_t	N, # of samples; R, rank of SVD approximation; L, lag window length for imputation.
MMMF	X_t	Y_t	N, # of samples; C, coefficient for slack penalty;
SMF	X_t, Y_{t-1}	Y_t	N_r, # of random samples; N_c, # of most confident samples. N_u, # of most uncertain samples; C, slack penalty.
SMFA	X_t, Y_{t-1}	Y_t	N, # of total samples; C, slack penalty; P_0, initial sample rate; P_a, active sample rate at each iteration; ρ, ratio of random sample and most uncertain sample for P_a;
TENSOR	$X_{t-L+1}, ..., X_{t-1}, X_t$	Y_t	N, # of samples in X_t; R, # of rank-1 components; λ, $R \times 1$ vector, with each one be the weight of an outer product of a sub-dimension.
MMMFA	X_t	Y_t	N, # of total samples; C, slack penalty; P_0, initial sample rate; P_a, active sample rate at each iteration; ρ, ratio of random sample and most uncertain sample for P_a;
H^*	$Y_{t-L+1}, ..., Y_t$	Y'_t	L, lag window length; θ, damping factor for smoothing.

The infrastructure federates some 350 sites world-wide, gathering more than 250,000 cores, which makes it the largest non-profit distributed system worldwide. Hardware and software failures are intrinsic to such large-scale systems. Middleware e.g. gLite [23], Globus [15] or ARC [11] cannot handle this without substantial human intervention. Access rights to EGI are primarily organized along the concept of Virtual Organization (VO), and each of the 200 VOs has to be specifically configured on its supporting sites, which adds complexity and introduces extra failures. User communities exploit two strategies to cope with faults: overlay middleware e.g. DIRAC [42], DIANE [30], AliEn [3] and PaNDA [28] implements specific fault-tolerance strategies to isolate users from the vagaries of the infrastructure; and monitoring identifies problems and quantifies performance w.r.t. quality of service agreements.

The data source for this study is the Biomed VO. Biomed has access to 256 Computing Elements (CEs) and 121 Storage Elements (SEs). CEs are shares of computing resources, implemented as queues of each site manager (e.g. PBS), and SEs are shares of storage resources; the formal definition is part of the Glue Information model [2]. File access remains one the major issues, for numerous causes spanning from hardware breakdowns to entanglements in the complicated verification of access rights, and including the bizarre transients reported by operations managers.

5.2 Data description

The dataset[1] was collected on EGI by submitting a series of jobs to 212 Biomed CEs every two hours between *Mon Nov 12 15:52 CET 2012* and *Sat Nov 24 09:54 CET 2012*, for about 282 hours in total. Each job tested the service availability between its CE and each of 96 Biomed SEs, by launching the *lcg-cp* probe. lcg-cp copies a file (like Unix cp), thus is a relatively high-level probe

[1]The dataset is publicly available of the Grid Observatory web site (www.grid-observatory.org)

Table 2: Illustration of duration length for OK and Failure

sequence	1	1	0	1	1	1	-1	0	-1	-1
	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
duration	2	1		3	2	1	1		2	1

that tests core access (network path), availability of the access control services as well as reading and writing capacities. The CEs and SEs have been preselected as relatively reliable, in order to eliminate trivially discoverable faults.

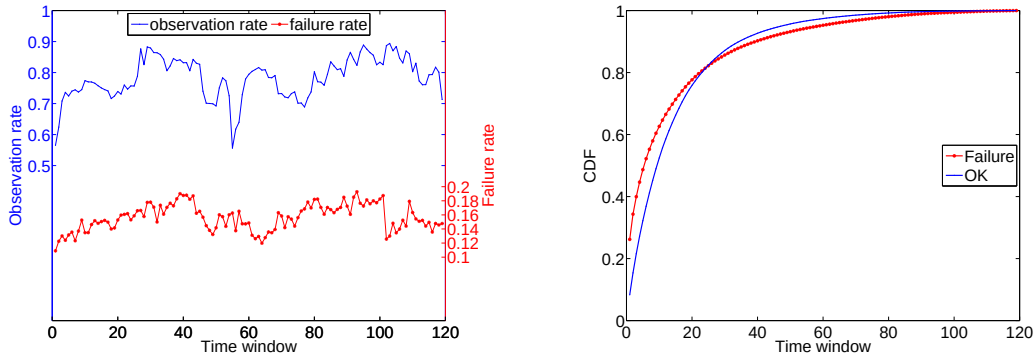
Of course, our test jobs were fully protected again the consequences of a *probe* failure (the job successful termination does not depend on the outcome of the probe), and the procedure has been designed so that the resources involved in running the jobs are as disjoint as possible from those required by the probes. However, our test jobs were no more immune to *middleware* faults than any other user job, and a significant part of them did fail, thus reporting no information at all. In order to get a consistent sample, we deleted the data from those CEs with less than 7000 observed entries and also from those time windows with less than 50% data observed. This results in a data cube of size $79 \times 96 \times 119$, with each dimension corresponding to CE, SE, and time window respectively. The goal of our experiment is to predict whether the jth SE is accessible from the ith CE at a given time window t, and this data cube is the *ground truth* for the prediction algorithms. We use 0 for representing a missing observation, 1 for a *Failed* probe (job succeeded and *lcg-cp* failed), and -1 for an OK probe. This notation is in accordance with the general meaning of positive (abnormal) and negative (normal) in statistics.

Let M be the total number of CEs, W be the total number of SEs, and $t_{k,k=1,2,\ldots,T}$ be the time window sequence, we further note N_{t_k} as the number of observed entries at t_k and $N^+_{t_k}$ be the number of positive entries (failures) at t_k, then the observation rate and test failure rate at t_k are defined as $r_{t_k} = N_{t_k}/MW$ and $f_{t_k} = N^+_{t_k}/MW$, respectively. Figure 3(a) illustrates the *observation rate* r_{t_k} and *failure rate* f_{t_k} of the dataset. Most of the observation rates stay above 70% and the failure rates are less than 20%. A high observation rate ensures a more reliable result for performance evaluation, since we have more ground truth information at hand. A relatively stable failure rate indicates a consistent system status in consecutive time windows. The failure rate presents some breakpoints, e.g. the sharp drop from 18.74% to 12.57% at the 101st time window. Their impact on prediction performance is discussed in sections 6 and 7.

Another interesting aspect of the data is the *duration length* of each status (i.e. OK or Failed). The duration length of a status is defined as the number of time windows the status spans until a different status is observed in the sequence; table 2 gives an example, and figure 3(b) shows the cumulative distributions. The high proportion of duration length 1 for Failed (about 25%) illustrates the phenomenon of transients, while the about 20% entries for both success and failure with a duration length at least 26 illustrate relative stable behavior. These distributions sustain our considerations about a multi-scale in time behavior.

5.3 Criteria

With the availability of the ground truth, the classical performance indicators for binary classification can be measured. Accuracy (the ratio of correctly predicted entries over the total number of entries) is of limited interest: as the data set is not balanced, overly optimistic algorithms that favor OK predictions will exhibit satisfactory accuracy. The indicators associated with the risks (confusion matrix) are more informative: *sensitivity* (True Positive Rate), the proportion of actual positives that are correctly predicted; *specificity* (True Negative Rate), the proportion of actual negatives that are correctly predicted; *precision*, the ratio of true positives over all predicted positives.



(a) Observation rate and failure rate over time

(b) CDF of duration of OK and Failure

Figure 3: Statistics of dataset.

They all make sense for operational needs, but sensitivity is the most important one: an undetected failure (bad sensitivity) results in a failed user job and a dissatisfied user, while a false negative might go unnoticed; specifically when the services are redundant (e.g. from replicated data), the failed request can be transparently rerouted to another server. Of course, specificity is nonetheless an important criterion, and compound criteria such as precision (PPV in the tables of sections 6 and 7, or Fscore measure various tradeoffs between them.

The last important indicator is the Matthews Correlation Coefficient (MCC), a correlation coefficient between the observed and predicted binary classifications that is relatively insensitive to unbalanced classes. Its interest comes from the fact that MCC is a proxy for the Area Under ROC (Receiver Operating Characteristic) Curve (AUC), which summarizes the intrinsic quality of a binary classifier independent of the decision threshold. Moreover, MCC does not assume that the classification error is a reasonable estimation of the prediction error [19], thus is a more robust indicator with respect to the objective functions involved in the optimization step of the algorithms. Other related indicators such as those involved in Neymann-Pearson learning [38] have not been reported, as they have not still gained widespread acceptance.

5.4 Details

For all non-active algorithms, 10% of the total entries are selected as training set (i.e. $N = 10\% \times W \times M$), and for SMF and SMFA another 10% of the most confident entries with values from Y_{t-1} is added as in algorithm-4. The adaptive weight for the most confident entries is computed according to line 5 in algorithm-4. Table 3 lists the concrete parameter settings for the algorithms of table 1. Parameters marked with a '+' are selected via training and validation on the first 20 time windows.

All experiments are performed 10 times. The results are presented in two ways: tabulated averages with standard deviation for precise numerical values, and notched box plots. Although not a formal test, if two boxes notches do not overlap there is strong evidence (95% confidence) that the corresponding medians differ [29].

For MMMF and SMF, and their active variants, we used CSDP [5] as the SDP solver. For Tensor we used the tensor toolbox [1] as the tensor completion solver, and for SSVD we used the Matlab internal *svd* function. All experiments were conducted on a standard laptop (quad-core Intel Core i7 processor with 8GB memory). Table 4 gives the empirical runtime of different algorithms for one matrix prediction.

Table 3: Summary of parameter values

	Parameters
EWM	$N = 10\%$ of random samples in X_t; $\theta = 0.5$, damping factor; $L = 20$, input window length.
SSVD	$N = 10\%$ of random samples in X_t; $R^+ = 10$, rank of SVD approximation; $L = 20$, lag window length for imputation.
TENSOR	$N = 10\%$ of random samples in X_t; $R^+ = 10$, # of rank-1 components; λ=ones(10,1), equal weight on each sub-dimension.
MMMF	$N = 10\%$ of random samples in X_t; $C^+ = 10$, coefficient for slack penalty; $\Sigma^+ =$'*max norm*'.
SMF	$N_r = 5\%$ of random samples in X_t; $N_u = 5\%$ of most uncertain samples in X_t; $N_c = 10\%$ of most confident samples from Y_{t-1}; $C^+ = 10$, slack penalty.
MMMFA	$P_0 = 5\%$, initial sample rate; $P_a = 1\%$, active sample rate at each iteration; $\rho^+ = 0.5$, equal size of random samples and most uncertain samples at each active iteration; $C^+ = 10$, slack penalty.
SMFA	$P_0 = 5\%$, initial sample rate; $P_a = 1\%$, active sample rate at each iteration; N_c, 10% of most confident prediction from Y_{t_i}; $\rho^+ = 0.5$, equal size of random samples and most uncertain samples at each active iteration; $C^+ = 10$, slack penalty.
H^*	$L = 20$, lag window length; $\theta = 0.5$, damping factor.

Table 4: Empirical runtime for the main algorithms

	MMMF	MMMFA	SMF	SMFA	SSVD	TENSOR
Time (Secs)	1.29	16.37	15.70	208.54	0.135	24.79

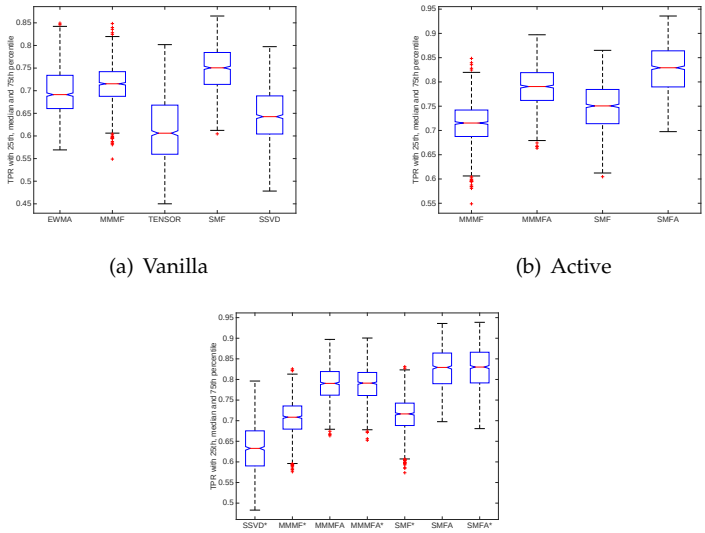
6 Experimental results - Non-curated dataset

Four approaches have been presented: pure spatial with MMMF, pure temporal with EMW, collapsed with SSVD, and integrated, with TENSOR and our algorithm, SMF. Some of these methods are amenable to active learning, and smoothing can be applied to either the vanilla methods, or their active counterpart, or both. Exhaustive exploration of the experimental landscape would only confuse the interpretation. Thus, we first analyze the vanilla algorithms, then evaluate the impact of active learning and of smoothing. Some supplementary material is available in [12].

6.1 Vanilla methods

The results of table 5 and figure 4(a) can be analyzed along different paths. Firstly, while all algorithms reach fairly good specificity, sensitivity exhibits only acceptable performance: 25-30% of the actual failures would not be predicted. This is a natural, but nonetheless problematic effect of the imbalanced dataset.

Going further illustrates the difficulty of comparisons: the ranking of SSVD and Tensor on the one hand, and of the pure temporal method (EWM) on the other hand is reversed for the sensitivity and specificity criteria. For an imbalanced matrix with negative as the majority, the principal factors preserved after a singular value decomposition mainly reflect the negative population.



(a) Vanilla

(b) Active

(c) Smoothed

Figure 4: Descriptive statistics of methods on the non-curated dataset. Pairwise comparison shows significant difference at 95% confidence level, except for the impact of smoothing.

Table 5: Average performance comparison for vanilla algorithms

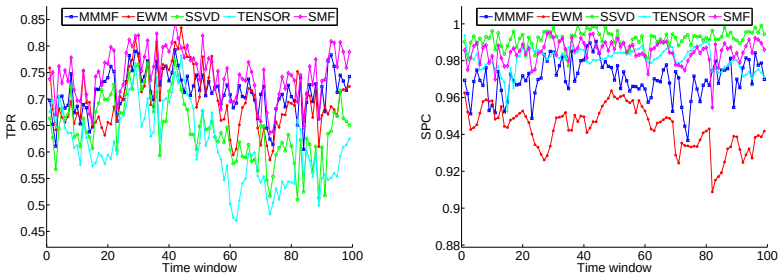
	TPR	SPC	PPV	MCC	FSCORE
MMMF	0.713±0.040	0.970±0.010	0.824±0.045	0.725±0.051	0.764±0.041
EWM	0.699±0.054	0.944±0.011	0.703±0.052	0.643±0.046	0.699±0.037
SSVD	0.645±0.060	**0.992±0.004**	**0.941±0.028**	0.747±0.047	0.763±0.046
TENSOR	0.613±0.071	0.981±0.006	0.859±0.043	0.684±0.055	0.713±0.053
SMF	**0.747±0.047**	0.985±0.006	0.901±0.038	**0.791±0.046**	**0.816±0.040**

SVD sees the isolated positive entries as noise, which the top rank eigenvector reconstruction is likely to remove. Thus SSVD tends to have excellent average specificity (0.992 ± 0.004) but poor sensitivity (0.635 ± 0.0645). In contrast, EWM tends to be more agnostic. The effectiveness of EWM on sensitivity must be stressed, as its simplicity might be appealing for problems with strong real-time constraints.

Like matrix based methods, TENSOR also shows a clear superiority to EWM on most criteria. However, when looking at sensitivity, TENSOR performs the worst. This poor performance might stem from the fact that tensor factorization performs a single least square estimation for the observed entries, without any regularization.

Finally, SMF shows a significant advantage on sensitivity, which is our primary performance indicator (c.f. section 5.3) and ranks only second for specificity (0.985 ± 0.006) and precision, translating into an altogether clear advantage on the compound indicators. Integrating the most confident information from the last predictionY_{t-1} brings some of the advantages of SSVD, although with a completely different technique: given the large proportion of negative entries in X, the majority part of most confident entry set are negative values.

The time series presented in figures 5(a) and 5(b) provide some insight on the factors of performance. MMMF and SMF behave essentially in lockstep on sensitivity, showing that matrix factorization provides a decisive contribution; the most important gains of SMF over MMMF occur in the relatively stable intervals (e.g. 40-55) where the knowledge of the past matters. This is also true for specificity, except at the sharp drop of SMF at the 81st time window (recall that



(a) Sensitivity (b) Specificity

Figure 5: Time series of sensitivity and specificity for the vanilla algorithms. Windows are numbered from the first one where prediction starts.

we use 20 time windows as initial input, so this is the 101st time window in the original data). As mentioned in 5.2, this is caused by a drop in real failures between the observations in the two adjacent time windows. In this case the historical information does not help, but instead hinders performance improvement, biasing the algorithm towards false positives. EWM highlights the phenomenon, with a similar drop of specificity. As can be expected, SSVD is largely unaffected.

6.2 Active methods

Table 6: Average performance comparison for MMMF, SMF, MMMFA

	TPR	SPC	PPV	MCC	FSCORE
MMMF	0.713±0.040	0.970±0.010	0.824±0.045	0.725±0.051	0.764±0.041
MMMFA	0.789±0.037	0.959±0.013	0.800±0.048	0.752±0.052	0.793±0.041
SMF	0.747±0.047	**0.985±0.006**	0.901±0.038	0.791±0.046	0.816±0.040
SMFA	**0.826±0.047**	0.983±0.007	**0.907±0.033**	**0.840±0.046**	**0.864±0.038**

As explained in section 4.2, active learning is a candidate for improving on sensitivity. Table 6 and and figure 4(b) compares MMMF and SMF with their active versions. In both cases, sensitivity improves by 11%, while the decrease in specificity is negligible (respectively 1% and 0.2%). Moreover, SMFA outperforms MMMFA, but active learning is powerful enough to make MMMFA outperform SMF on sensitivity by 6%. In other words, the good selection of current information allows to forget the past: for selecting the most uncertain prediction, which is likely to be on the positive entries, active sampling on the current data does a better job than passive history.

6.3 Smoothing

Because the simple smoothing of EWM was relatively successful for sensitivity, one can wonder whether it would not be competitive with the active approach, or improve on it. In fact, smoothing actually often degrades sensitivity even with respect to the vanilla algorithm (e.g. MMMF goes from 0.713 down to 0.700): the smoothing process over-corrects the false positive predictions.

Table 7 and figure 4(c) compares the active versus smoothing approach (e.g. SMFA vs SMF*); clearly smoothing is not competitive with active learning on sensitivity, although it always improves specificity and sometimes the compound criteria. On the other hand, combining smoothing and active learning has contrasted results, degrading MMMFA sensitivity, but marginally

Table 7: Average performance of SSVD*, MF*, SMFA*, TENSOR, SMF*, and MMMFA*

	TPR	SPC	PPV	MCC	FSCORE
SSVD*	0.635±0.063	**0.997±0.004**	**0.974±0.034**	0.757±0.050	0.767±0.050
MMMF*	0.700±0.045	0.990±0.004	0.933±0.031	0.778±0.041	0.799±0.037
MMMFA	0.789±0.037	0.959±0.013	0.800±0.048	0.752±0.052	0.793±0.041
MMMFA*	0.788±0.039	0.987±0.005	0.924±0.029	0.826±0.038	0.850±0.032
SMF*	0.716±0.071	0.993±0.005	0.947±0.042	0.797±0.053	0.813±0.051
SMFA	0.826±0.047	0.983±0.007	0.907±0.033	0.840±0.046	0.864±0.038
SMFA*	**0.827±0.047**	0.991±0.005	0.950±0.028	**0.865±0.041**	**0.884±0.036**

improving on SMFA. Finally, SSVD* dominates all the other algorithms on specificity, but with unacceptably low sensitivity (SMFA is 30% better), indicating its tendency to favor negative predictions.

For completeness (results not shown), we experimented smoothing the TENSOR method. As can be expected, there is no significant difference between TENSOR and TENSOR*, as TENSOR employs a regression on the time dimension directly, and continuity in estimation sequence is already captured, leaving no room for a smoothing-based enhancement.

6.4 Summary

The above analysis firstly emphasizes the effectiveness of the sequential matrix factorization approach, and specifically of the proposed SMF algorithm and its variants: a properly synthesized use of spatial and temporal information significantly outperforms purely spatial, temporal, or collapsed methods, at equal levels of method complexity (e.g. SMF vs MMMF or SSVD).

Then, active learning is consistently and significantly beneficial: the positive entries are the minority part of the whole population, it is therefore difficult to uncover them by using any conventional method with equal cost on positive and negative entries. However, with the aid of active sampling it is possible to unveil those *difficult to predict* entries, since they are more likely to be exposed and labeled during the active sampling process.

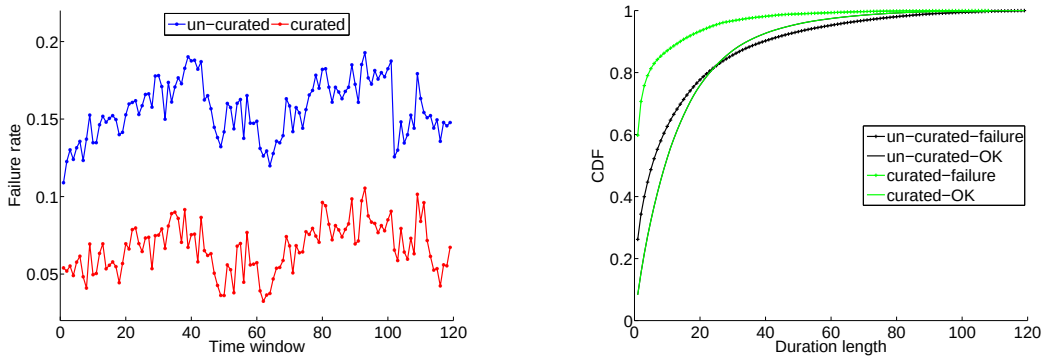
Finally, simple smoothing cannot compete with the active approach, and should be considered useful only combined with active learning, and only when false positives are a major concern.

7 Experimental results with the curated dataset

7.1 The curated dataset

It could be argued that our benchmark is too easy: the tail of the distribution of the failure duration lengths corresponds to long-lasting errors, that basic monitoring tools (e.g. heartbeats) would report anyway, and the prediction methods should be applied only to more elusive causes of errors. While this is disputable (remember that all probes succeed as jobs, thus a significant part of the services are up and running), it is worth assessing the performance of the methods when these systematic errors are eliminated. Therefore, we designed a second set of experiments, with *curated* matrices as the reference fault structure.

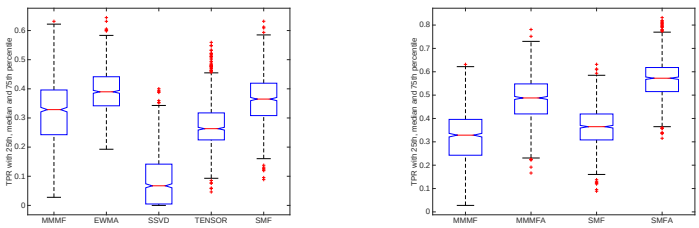
The curated dataset is derived by removing those lines and columns with at least 98% failed entries in the reference matrices. Figure 6(a) shows the magnitude of the decrease in the failure rate, approximately from 15% to 5% on average. Moreover, the CDF of failure duration length also experiences a sharp change (figure 6(b)). The percentage of length-one durations increases from 25% to about 60%, and the percentage of duration lengths less than 20 grows from 75% to approximately 92%. In other words, after the elimination of systematic failures, the proportion of short term failures increases significantly.



(a) Timeseries of failure rates

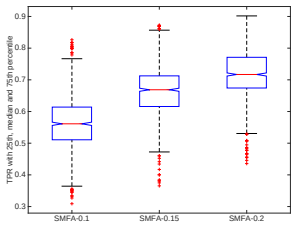
(b) CDF of duration lengths

Figure 6: Comparison of curated and un-curated datasets.



(a) Vanilla

(b) Active



(c) Sampling rate

Figure 7: Descriptive statistics of methods on the curated dataset. Pairwise comparison shows significant difference at 95% confidence level.

7.2 Vanilla methods

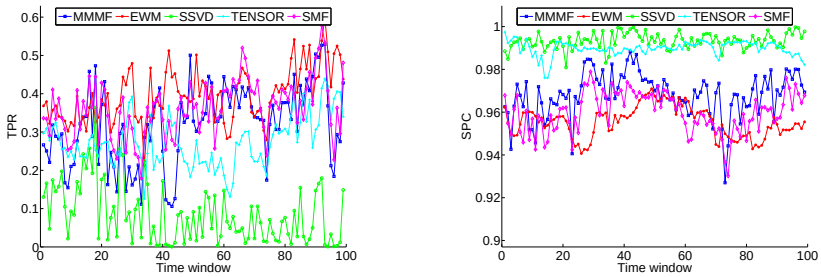
The very adverse curated dataset produces quite interesting results (table 8 and figure 7(a)) concerning sensitivity. Most importantly, all vanilla method perform poorly: at best (EWM), more than 60% of the faults will get unpredicted. The extreme imbalance of the data obviously highlights the limits of accuracy-based methods (MMMF).

The real surprise is that the basic EWM gives the best (or the least bad) performance. Our SMF is very close (1% difference), while EWM is 21% better than MMMF. This confirms that using only the snapshot information (MMMF) cannot suffice in such a sparse setting: historical information is critical (EWM, SMF), but must be properly used (SSVD behaves poorly).

The catastrophic results of SSVD on sensitivity might seem at odds with the relatively good ones of EWM, as SSVD is the singular value decomposition of the EWM-collapsed matrix. The

Table 8: Average performance comparison of MMMF, SMF and MMMFA on curated dataset

	TPR	SPC	PPV	MCC	FSCORE
MMMF	0.319±0.102	0.968±0.011	0.427±0.110	0.328±0.106	0.361±0.107
EWM	**0.396±0.069**	0.955±0.007	0.392±0.066	0.347±0.053	**0.389±0.051**
SSVD	0.084±0.076	**0.993±0.004**	0.398±0.211	0.150±0.124	0.130±0.112
TENSOR	0.275±0.065	0.990±0.003	**0.661±0.094**	**0.397±0.068**	0.381±0.069
SMF	0.362±0.074	0.960±0.009	0.374±0.078	0.326±0.075	0.365±0.074



(a) Sensitivity (b) Specificity

Figure 8: Curated dataset: time series of sensitivity and specificity for the vanilla algorithms. Windows are numbered from the first one where prediction starts.

problem stems from the fact that with so few not too transient failures, the EWM-collapsed matrix is highly unbalanced. Without interpretation (EWM algorithm), it may preserve some of them, while the Singular Value Decomposition followed by binarization squashes them, for the same reasons that have been identified in the analysis of the non-curated case, but with much more impact.

In all cases, specificity is more than acceptable, even if, as previously EWM shows the worst one. With such sensitivity/specificity imbalance, the compound criteria do not make much sense, as confirmed by the spread of the results.

Figure 8 give some insights about the issues encountered by SMF. For sensitivity, contrary to the non-curated case, SMF and MMMF are not in lockstep. More precisely, there are intervals where EWM and MMMF behave in opposite directions, e.g. at time 40-45; there, taking into account the past somehow helps SMF to limit its loss, but the weight of the catastrophic behavior of MMMF is still to high.

7.3 Active learning

Table 9 and figure 7(b) show the results for the active versions of static (MMMFA) and sequential (SMFA) matrix factorization. As can be expected, active learning procures a decisive improvement, in the order of more than 50% for both. SMFA has a clear advantage: it is capable of discovering more than 50% of the faults while maintaining good specificity and acceptable precision: only 25% of the alarms are spurious.

For lack of space, the results of smoothing are not given. Their trend is very similar to the one of section 6.3.

7.4 Higher sampling rate

So far, the sampling rate was limited to 10%. Table 10 and figure 7(c) illustrate the result of increased sample rates, i.e. 10%, 15% and 20%, of SMFA* on the curated dataset. Results are

Table 9: Average performance comparison of MMMF, SMF and MMMFA on curated dataset

	TPR	SPC	PPV	MCC	FSCORE
MMMF	0.319±0.102	0.968±0.011	0.427±0.110	0.328±0.106	0.361±0.107
MMMFA	0.482±0.081	0.959±0.014	0.471±0.080	0.436±0.080	0.471±0.077
SMF	0.362±0.074	0.960±0.009	0.374±0.078	0.326±0.075	0.365±0.074
SMFA	**0.569±0.076**	**0.986±0.006**	**0.743±0.079**	**0.628±0.080**	**0.642±0.076**

Table 10: Average performance of SMFA* with different sampling rate, curated dataset

	Sensitivity	Specificity	Precision	MCC	FSCORE
SMFA*-0.1	0.562±0.078	0.993±0.003	0.853±0.053	0.675±0.069	0.675±0.070
SMFA*-0.15	0.664±0.076	0.997±0.002	0.947±0.031	0.780±0.057	0.778±0.059
SMFA*-0.2	**0.720±0.074**	**0.998±0.002**	**0.970±0.033**	**0.825±0.051**	**0.825±0.054**

averaged on a 5-run experiment. A steady and notable improvement is exhibited. For example, at 20%, SMFA* finds out 72% of the faults while keeping a balanced MCC value of 0.825. The good news is that the prediction performance increases steadily with the sample rate even in the curated situation. However, the sampling rate drives two costs: the monitoring overhead per se, and the computational cost, as discussed in section 2.1. To control the computational cost, accelerating methods like Fast MMMF [34] might be required.

7.5 Summary

The curated dataset is highly imbalanced, with only about 5% positive entries. Then static accuracy-based classification becomes awkward and exhibits poor performance. Active learning is one of the few approaches that can contribute to alleviating this issue. Our question in this section was its impact within already "smart" - heuristic based - methods. The results show that active learning carries a consistent improvement, with a very similar factor, for the two relatively different heuristics involved with SMF and MMMF; in other words, active learning is robust in this context. To be complete, analogous techniques for SSVD, such as weighted synthetic oversampling [18] could be explored, as well as improvements over the base SVD [31]; however, the very poor results of the baseline and its intrinsic over-denoising characteristic make the solution not attractive.

8 Conclusion

Efficient monitoring of production grids and clouds at acceptable manpower cost cannot assume exhaustive a priori knowledge of their software and hardware infrastructures. In this context, and with end-to-end probing as sole data acquisition strategy, fault discovery can be re-casted as an inference task, and can borrow methods from collaborative prediction. The challenge as well as the opportunity brought by switching from a static, snapshot-oriented, view of the monitoring to considering the time dimension reside in the sequential correlation between consecutive data points. We have shown that these sequential patterns, if exploited properly, can play an important role in improving prediction performance.

This paper explores various methods that combine time and space information with increasing complexity. It proposes and evaluates SMF, a fully integrated method that exploits both the recent advances in matrix factorization for the spatial information and a new heuristics based on historical information. The effectiveness of the SMF approach has been exemplified on datasets of increasing difficulty. In all cases, active learning unleashed the full potential of coupling the most confident and the most uncertain heuristics, which is the cornerstone of SMF. To our question about the need of adaptive, and thus more complicated and fault-prone algorithms, the answer

is unambiguous: a method versatile enough for accommodating various levels of difficulty on both the sensitivity and specificity criteria must include adaptivity through active learning.

Future work will go further in the adaptivity direction. The first perspective considers self-calibrating the SMFA parameters in an auto-learning approach. The samples are selected with two strategies: the most uncertain predictions in the last run guide the selection of samples to enhance the current prediction confidence, while the random sampling strategy avoids over-fitting the past.The current sample ratio between the two strategies is fixed and set to $1:1$. However, a straightforward extension is to address this problem under the sequential decision optimization framework. The hybrid optimization indicator of [44] can be considered to efficiently balance the exploitation and exploration trade-off.

We have seen that performance was limited by the occurrence of abrupt changes, where the advantage of taking into account the past turns into a liability. The second perspective considers a semi-supervised online change detection framework that has already proved to be efficient at the level of the individual timeserie [12]. The next step would be to extend it towards the full matrix data.

References

[1] Evrim Acar, Daniel M. Dunlavy, Tamara G. Kolda, and Morten Mørup. Scalable tensor factorizations for incomplete data. *Chemometrics and Intelligent Laboratory Systems*, 106(1):41–56, March 2011.

[2] S. Andreozzi and al. Glue Schema Specification, V.2.0. Technical report, Open Grid Forum, 2009.

[3] S Bagnasco and al. Alien: Alice environment on the grid. *Journal of Physics: Conference Series*, 119(6):062012, 2008.

[4] Paul Barham, Austin Donnelly, Rebecca Isaacs, and Richard Mortier. Using magpie for request extraction and workload modelling. In *OSDI*, volume 4, pages 18–18, 2004.

[5] Brian Borchers. Csdp, ac library for semidefinite programming. *Optimization methods and Software*, 11(1-4):613–623, 1999.

[6] Rasmus Bro. Parafac. tutorial and applications. *Chemometrics and intelligent laboratory systems*, 38(2):149–171, 1997.

[7] Emmanuel J. Candes and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

[8] Emmanuel J. Candès and Terence Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Trans. Inf. Theor.*, 56(5):2053–2080, 2010.

[9] Mike Y Chen, Emre Kiciman, Eugene Fratkin, Armando Fox, and Eric Brewer. Pinpoint: Problem determination in large, dynamic internet services. In *Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on*, pages 595–604. IEEE, 2002.

[10] Mark A. Davenport, Yaniv Plan, Ewout van den Berg, and Mary Wootters. 1-bit Matrix Completion. *Information and Inference*, 3(3):189–223, 2014.

[11] M. Ellert and al. Advanced resource connector middleware for lightweight computational grids. *Future Generation Computer Systems*, 23(2):219 – 240, 2007.

[12] Dawei Feng. *Efficient End-to-End Monitoring for Fault Management in Distributed Systems*. PhD thesis, Universite Paris Sud, 2014.

[13] Dawei Feng, Cecile Germain-Renaud, and Tristan Glatard. Efficient distributed monitoring with active collaborative prediction. *Future Generation Computer Systems*, 29(8):2272–2283, 2013.

[14] Rodrigo Fonseca, George Porter, Randy H Katz, Scott Shenker, and Ion Stoica. X-trace: A pervasive network tracing framework. In *Proceedings of the 4th USENIX conference on Networked systems design & implementation*, pages 20–20. USENIX Association, 2007.

[15] I. Foster. The globus toolkit for grid computing. In *IEEE Int. Symp. on Cluster Computing and the Grid*, 2001.

[16] Dennis Geels, Gautam Altekar, Petros Maniatis, Timothy Roscoe, and Ion Stoica. Friday: Global comprehension for distributed replay. In *NSDI*, volume 7, pages 285–298, 2007.

[17] Seth Gilbert and Nancy Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2):51–59, 2002.

[18] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 21(9):1263–1284, 2009.

[19] T. Joachims. A support vector method for multivariate performance measures. In *International Conference on Machine Learning (ICML)*, pages 377–384, 2005.

[20] Charles Killian, James W Anderson, Ranjit Jhala, and Amin Vahdat. Life, death, and the critical transition: Finding liveness bugs in systems code. *NSDI 07: Networked Systems Design and Implementation*, pages 243–256, 2007.

[21] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[22] Akshay Krishnamurthy and Aarti Singh. Low-rank matrix and tensor completion via adaptive sampling. In *Advances in Neural Information Processing Systems*, pages 836–844, 2013.

[23] E. Laure and al. Programming the Grid with gLite. In *Computational Methods in Science and Technology*, volume 12, pages 33–45, 2006.

[24] Bin Li, Xingquan Zhu, Ruijiang Li, Chengqi Zhang, Xiangyang Xue, and Xindong Wu. Cross-domain collaborative filtering over time. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*, pages 2293–2298. AAAI Press, 2011.

[25] Ji Liu, Przemyslaw Musialski, Peter Wonka, and Jieping Ye. Tensor completion for estimating missing values in visual data. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2114–2121. IEEE, 2009.

[26] Xuezheng Liu, Zhenyu Guo, Xi Wang, Feibo Chen, Xiaochen Lian, Jian Tang, Ming Wu, M Frans Kaashoek, and Zheng Zhang. D3s: Debugging deployed distributed systems. In *NSDI*, volume 8, pages 423–437, 2008.

[27] Xuezheng Liu, Wei Lin, Aimin Pan, and Zheng Zhang. Wids checker: Combating bugs in distributed systems. In *NSDI*, 2007.

[28] T Maeno. Panda: distributed production and distributed analysis system for atlas. *Journal of Physics: Conference Series*, 119(6):062036, 2008.

[29] Robert McGill, John W. Tukey, and Wayne A. Larsen. Variations of box plots. *The American Statistician*, 32(1):12–16, February 1978.

[30] J. T. Moscicki. Diane - distributed analysis environment for grid-enabled simulation and analysis of physics data. In *Nuclear Science Symposium Conference Record, 2003 IEEE*, volume 3, pages 1617–1620 Vol.3, 2003.

[31] Chengbin Peng, Ka-Chun Wong, Alyn Rockwood, Xiangliang Zhang, Jinling Jiang, and David Keyes. Multiplicative algorithms for constrained non-negative matrix factorization. In *ICDM*, pages 1068–1073. IEEE Computer Society, 2012.

[32] Andres Quiroz, Manish Parashar, Nathan Gnanasambandam, and Naveen Sharma. Design and evaluation of decentralized online clustering. *ACM Trans. Auton. Adapt. Syst.*, 7(3):34:1–34:31, 2012.

[33] Benjamin Recht. A simpler approach to matrix completion. *J. Mach. Learn. Res.*, 12:3413–3430, 2011.

[34] Jasson DM Rennie and Nathan Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd international conference on Machine learning*, pages 713–719. ACM, 2005.

[35] Patrick Reynolds, Charles Edwin Killian, Janet L Wiener, Jeffrey C Mogul, Mehul A Shah, and Amin Vahdat. Pip: Detecting the unexpected in distributed systems. In *NSDI*, volume 6, pages 115–128, 2006.

[36] Irina Rish, Mark Brodie, Sheng Ma, Natalia Odintsova, Alina Beygelzimer, Genady Grabarnik, and Karina Hernandez. Adaptive diagnosis in distributed systems. *IEEE Trans. Neural Networks*, 16(5):1088 – 1109, 2005.

[37] Irina Rish and Gerald Tesauro. Estimating end-to-end performance by collaborative prediction with active sampling. In *Integrated Network Management*, pages 294–303, 2007.

[38] Clayton Scott. Performance measures for neyman-pearson classification. *IEEE Trans. Inf. Theor.*, 53(8):2852–2863, August 2007.

[39] Nathan Srebro, Jason D. M. Rennie, and Tommi S. Jaakola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems 17*, pages 1329–1336, 2005.

[40] Michel Tokic. Adaptive epsilon-greedy Exploration in Reinforcement Learning Based on Value Differences. In *33rd Annual German Conference on Advances in Artificial Intelligence*, LNCS 6359, pages 203–210, Berlin, Heidelberg, 2010. Springer-Verlag.

[41] Erik Torres, German Molto, Damia Segrelles, and Ignacio Blanquer. A replicated information system to enable dynamic collaborations in the grid. *Concurr. Comput. : Pract. Exper.*, 24(14):1668–1683, 2012.

[42] A Tsaregorodtsev and al. DIRAC3 . The New Generation of the LHCb Grid Software. *Journal of Physics: Conference Series*, 219(6):062029, 2009.

[43] Srikumar Venugopal, Rajkumar Buyya, and Kotagiri Ramamohanarao. A taxonomy of data grids for distributed data sharing, management, and processing. *ACM Comput. Surv.*, 38(1), June 2006.

[44] Weijia Wang and Michèle Sebag. Hypervolume indicator and dominance reward based multi-objective Monte-Carlo Tree Search. *Machine Learning*, 92(2-3):403–429, May 2013.

[45] Xiangliang Zhang, Cyril Furtlehner, Cecile Germain-Renaud, and Michele Sebag. Data stream clustering with affinity propagation. *IEEE Transactions on Knowledge and Data Engineering*, 26(7), 2014.

[46] Wenchao Zhou. Fault management in distributed systems. Technical Report MS-CIS-10-03, University of Pennsylvania Department of Computer and Information Science, 2010.