Efficient Incremental Model Learning on Data Streams

by

Xilun Chen

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved April 2019 by the
Graduate Supervisory Committee:

K. Selçuk Candan, Chair
Hasan Davulcu
Giulia Pedrielli
Maria Luisa Sapino
Hanghang Tong

ARIZONA STATE UNIVERSITY

August 2019

ABSTRACT

With the development of modern technological infrastructures, such as social networks or the Internet of Things (IoT), data is being generated at a speed that is never before seen. Analyzing the content of this data helps us further understand underlying patterns and discover relationships among different subsets of data, enabling intelligent decision making. In this thesis, I first introduce the Low-rank, Windowed, Incremental Singular Value Decomposition (SVD) framework to inclemently maintain SVD factors over streaming data. Then, I present the Group Incremental Non-Negative Matrix Factorization framework to leverage redundancies in the data to speed up incremental processing. They primarily tackle the challenges of using factorization models in the scenarios with streaming textual data. In order to tackle the challenges in improving the effectiveness and efficiency of generative models in this streaming environment, I introduce the Incremental Dynamic Multiscale Topic Model framework, which identifies multi-scale patterns and their evolutions within streaming datasets. While the latent factor models assume the linear independence in the latent factors, the generative models assume the observation is generated from a set of latent variables with various distributions. Furthermore, some models may not be accessible or their underlying structures are too complex to understand, such as simulation ensembles, where there may be thousands of parameters with a huge parameter space, the only way to learn information from it is to execute real simulations. When performing knowledge discovery and decision making through data- and model-driven simulation ensembles, it is expensive to operate these ensembles continuously at large scale, due to the high computational. Consequently, given a relatively small simulation budget, it is desirable to identify a sparse ensemble that includes the most-informative simulations, while still permitting effective exploration of the input parameter space. Therefore, I present Complexity-Guided Parameter

Space Sampling framework, which is an intelligent, top-down sampling scheme to select the most salient simulation parameters to execute, given a limited computational budget. Moreover, I also present a Pivot-Guided Parameter Space Sampling framework, which incrementally maintains a diverse ensemble of models of the simulation ensemble space and uses a pivot guided mechanism for future sample selection.

DEDICATION

To my wife, and

my parents

ACKNOWLEDGMENTS

I would like to first thank my advisor, Dr. K. Selçuk Candan for his endless support on my work during this journey. His insightful comments, suggestions and guidance significantly helped me achieving my final goal. I'm always very grateful for having him as my advisor.

I would also like to wholeheartedly appreciate all my committee members, Dr. Hasan Davulcu, Dr. Giulia Pedrielli, Dr. Maria Luisa Sapino and Dr. Hanghang Tong for their valuable comments on my research. I really appreciate their help in my proposal defense and thesis defense.

I would like to thank Stephen Foldes, Brian Appavu, Austin Jacobson from Phoenix Children Hospital for a great research collaboration opportunity, I would also like to thank Mehran Kafai and Kave Eshghi for their help with my internship at HP Labs and Dr. Youngchoon Park for his help with my internship at Johnson Controls. I want to thank the graduate advisors at CIDSE and the technical staffs for always helping me on my requests.

I am very thankful for Xinsheng Li, Shengyu Huang, Jung Hyun Kim, Parth Nagarkar, Hans Behrens, Maolin Li, Sicong Liu, Yash Garg, Silvestro Poccia who helped me a lot during my study and life at ASU.

Lastly, I am very grateful for my parents and my wife for always being there whenever I needed them.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

CHAPTER Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Chapter 1

INTRODUCTION

With the development of modern communication infrastructures, such as IoT and social networks, data is generating at the speed which we would never envision before, analyzing the content of the data helps us further understand the underlying patterns and discover relationships among different sets of data, which enable intelligent decision making. For example, one would like to examine the interests for a group of users in a social network, such as Facebook, finding the latent semantics for these users enables accurate content recommendation and friendship suggestion. Moreover, users may registered in multiple social sites, aligning different user groups in various networks also help identify the user clustering information, and hence better understand the user interests and preferences.

To find the patterns and relationships in the data, usually a model is learned from a set of training data, which contains labels indicating ground truth information. There are three major approaches that are widely used to analyze data: (a) probabilistic generative models [63], (b) latent factor models [3] and (c) black box models [22]. Traditional probabilistic models must explicitly declare the number of latent clusters at the beginning. Bayesian nonparametric mixture models, such as [67], infer the number of latent clusters from the data instead of setting the number of clusters explicitly. Consequently, when the new data comes, the number of clusters can adaptively grow or shrink, and thus enable better inference of the data. Latent factor models, such as [37], on the other hand, decompose observed data into a linear combination of latent factors. While the latent factor models do not assume any domain and application knowledge of the data except for the linear independence in

it, the probabilistic generative models, on the other hand, assume the observation is generated from a set of latent variables with different distributions. Furthermore, in the blackbox models, one cannot directly infer the parameters from the observations since the underlying model is not accessible or too complex and thus sampling has to be involved when exploring them. Figure 1.1 shows some specific methods used in the three methods.

The main difference between probabilistic model and factor model is that, factor model assumes that every observation is of a degree of membership to a cluster, instead of assigning clusters explicitly in the probabilistic model. Typical factor models include singular value decomposition [37], non-negative matrix factorization [53], and tensor decomposition [52]. These techniques are usually used for dimensionality reduction and clustering by identifying the principal components given the observations and cluster the data based on degree of membership, and usually this process does not require any domain knowledge of the data. One advantage of these techniques is that the number of clusters is adjustable according to the data, and when the data comes in a streaming fashion, the number of clusters can be increased or decreased. The probabilistic model usually assume that the observation is generated from a set of latent variables with different distributions and sampling methods can be applied to maximize the posterior probability given this assumption and thus a probabilistic model can be learned from the data. These methods usually require some prior knowledge of the data. On the other hand, the simulation ensemble approach learns the model by exploring the parameter space resulting from running the actual simulations, instead of finding latent variables associate in the data, the simulation ensemble learns to find the most informative parameter settings that help

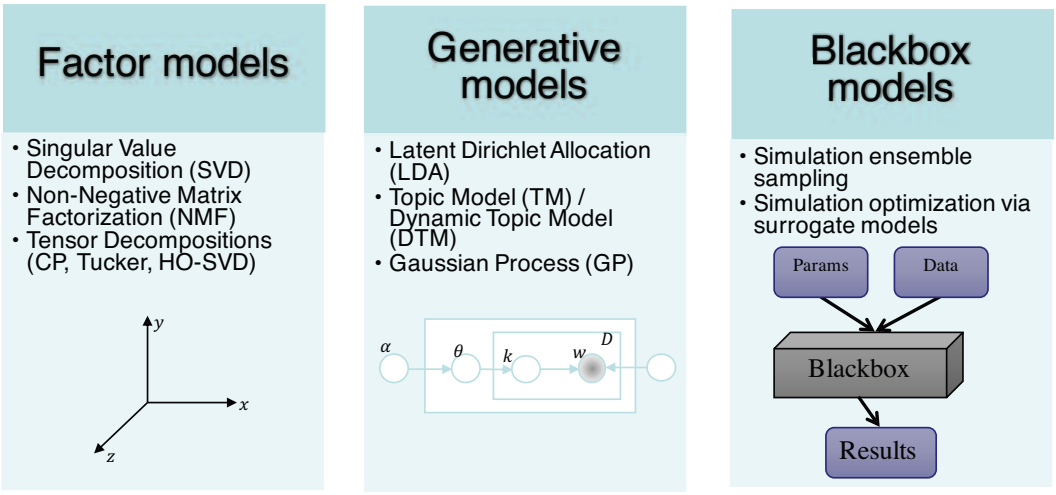


Figure 1.1: Model learning methods

with knowledge discovery tasks.

1.1 Model Learning with Latent Factor Models

In many applications, data have a multitude of features that can be used for indexing and retrieval. In practice, however, using more features (or having more feature dimensions to represent the data) is not always an effective way for managing data. This is commonly known as the *dimensionality curse* and has been shown to negatively affect many key data management tasks, from similarity searches to the analysis of data for patterns [6, 17]. In order to tackle this challenge, usually feature selection and dimensionality reduction are used, these techniques [91] usually involve some (often linear) transformation of the vector space containing the data, for example, the singular value decomposition (SVD [29]), to help focus on a few features (or combinations of features) that best discriminate the data in a given corpus.

However, usually obtaining the decomposition involves several matrix operations that need to be performed, SVD is computationally costly and therefore a naive implementation does not match the real-time needs of scenarios where data evolve

continuously: decomposition of an $n \times m$ matrix requires $O(n \times m \times min(n, m))$ time. To address this challenge, I introduce **LWI-SVD: Low-rank, Windowed, Incremental Singular Value Decompositions** algorithm [24] to improve the efficiency of repeatedly performing SVD in a streaming data scenario, more specially,

- LWI-SVD leverages efficient and accurate *low-rank approximations* to speed up incremental SVD updates
- LWI-SVD also uses a *window-based* approach to aggregate multiple incoming updates (insertion or deletions) and, thus reduces on-line costs

In order to reduce the error accumulated in the continuous decomposition, I also develop LWI2-SVD algorithm, which leverages a novel partial reconstruction based change detection technique to support timely refreshing of the decompositions to prevent accumulation of errors.

Another important approach in the latent factor model family is Non-negative matrix factorization (NMF), which is also a well known method for obtaining low rank approximations of data sets, that can then be used for efficient indexing, classification, and retrieval. The non-negativity constraints enable probabilistic interpretation of the results and discovery of generative models. One key disadvantage of the NMF, however, is that it is also costly to obtain because of the matrix operations that have to be performed, and this makes it difficult to apply NMF in applications where data is dynamic. Decomposing an $n \times m$ matrix requires $\#iterations \times O(n \times m \times r)$ time, where r is the rank of latent factors and $\#iterations$ is the number of iterations for the decomposition process to converge. In order to make it work in a dynamic scenario, I design **GI-NMF: Group Incremental Non-Negative Matrix Factorization** algorithm [23], which leverages the temporal redundancies in the data for reducing

the cost for characterization of each factorization (G-NMF), and also incrementally maintain the NMF in the scenario of the data stream (GI-NMF), more specifically,

- both G-NMF (Section 4.3) and GI-NMF (Section 4.4) algorithms solve the multiplicative update rule (MUR) subproblems by partitioning the original data matrices into submatrices (representing data shared by multiple queries) and combining the subfactors into the final factors. This method reduces the computation cost significantly by reusing the subfactors corresponding to data shared by different queries, and

- moreover, the GI-NMF algorithm (Section 4.4) maintains each partition's subfactors incrementally as the matrices change over the time and quickly identifies the updated factors for the next time instance, in order to scale to the needs of time-evolving data.

Both G-NMF and GI-NMF algorithms rely on novel update rules which simultaneously leverage (a) the incremental nature of the MUR based solutions to the NMF problems and (b) the special group structure of the error terms minimized during the MUR process.

1.2 Model Learning with Probabilistic Generative Model

Besides the factor models, probabilistic models are also widely used for analyzing data to find the latent topics, especially for web data. Web and social media data evolve over time reflecting events and trending topics in the real world: a topic may remain active for a long time or may end abruptly after a short and intense activity. Consequently, understanding the temporal scales of these topics and leveraging this information for inference can potentially help make better decisions and recommendations based on web data. Recently, probabilistic models for discovering latent

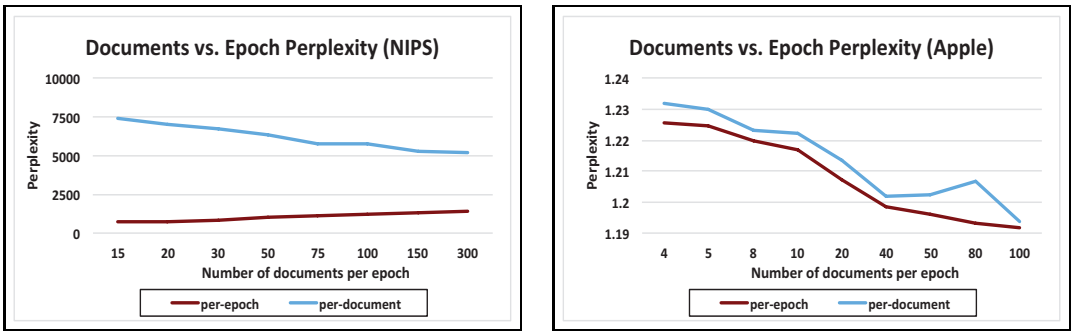
patterns in data have drawn significant attention due to the attractiveness of the underlying theory and the practical effectiveness of the probabilistic approaches to data analysis [4, 21, 87, 50]. Topic models [12] (TM) are a good example for the successful application of probabilistic techniques for discovery of latent patterns, including in scientific analysis [21], image analysis [87], and web social media data [4] analysis. A *topic model* is a hierarchical probabilistic model, where each object (e.g. document) is explained by a mixture of latent patterns (e.g. topics), and each pattern is further explained by a probability distribution of features (e.g. words). The time complexity for inferring hyperparameter of such topic model is usually linear in the size of objects and linear in the number of latent patterns one would seek for, and the number of iterations for the inference process to converge is highly depending on the data. The basic topic model does not consider time; i.e., it assumes that the data corpus is fixed; therefore, *dynamic topic model*s (DTMs) extend the idea by allowing the data corpus to evolve in *epochs* and by chaining topics from consecutive epochs together to track their evolution in time [11]. DTM and its variations have been successfully applied in many domains with evolving data, including in the analysis of web and social media data streams [89, 86].

However, a major limitation of most existing DTM approaches is that they assume a predetermined and fixed span (or epoch) of topics, whereas an evolving document corpus may contain topics of different temporal scales and, moreover, topics at one scale may impact the prediction of the topics at another scale. This leads to several difficulties when considering the diversity of web based data streams.

Difficulty of Picking an Epoch Size. One of the major challenges is that the relationship between epoch size and accuracy is not trivial to establish. Figure 1.2 shows how document- and epoch-level perplexities vary as a function of the epoch size for two sample data sets:

- For the NIPS corpus, representing scientific web collections, epoch-level perplexity increases with the epoch size; i.e, when using larger epochs, it gets more difficult to develop models that describe these epochs using a fixed number of target topics. This is because, the diversity of the corpus increases with time. In contrast, document-level perplexity improves with larger epochs: this is because, in this data, there is little local temporal correlation among documents (hence the document-level perplexity is relatively large) and a small epoch size is not sufficient to explain individual documents.

- Apple Stock data, representing financial data streams, shows a different behavior: for both epoch- and document-level perplexities, the model gets better a larger time periods are considered. This is because the data is too complex to explain accurately focusing on a small time period.



(a) NIPS data (b) Apple data

Figure 1.2: Document- and epoch-level perplexities of different data as a function of the epoch size. This figure illustrates the difficulty of setting the epoch size for analysis: not only different data have different perplexity behaviors; for the NIPS data, the per-epoch and per-document accuracies behave differently

This figure illustrates not only that different data and document streams have different perplexity behaviors, but also that for some data per-epoch and per-document accuracies may behave differently when the epoch size change.

Co-existence and Interdependence of Topics of Different Lengths. The above observations indicate that it may be difficult to set a good epoch size without accurate a priori information about the content, which is often difficult to obtain for most web-based data streams. It is important to note that, in general, such an optimal epoch size may not exist since (a) the relevant time span *may not be fixed and topics of different temporal spans may co-exist in the data stream as an active topic may last for a long time or may abruptly end after a short period.* For example, during a presidential election primaries will be relevant over a lengthy period, while discussions of many other related political events will emerge and disappear, tracking political developments with a shorter span. Moreover, (b) topics of one temporal scale may be predictive of subsequent topics of different temporal scales; in other words, an active topic may be predicted by a mixture of past topics, some of which with long spans and some of which having emerged only recently. Consequently, selecting the appropriate epoch size is not a trivial task:

- if the epoch length is too large, then (since all data objects in the same time epoch are exchangeable) it cannot discover fine grained dynamics (as well as larger patterns that depend on these dynamics) in the data stream;
- if the epoch length is too small, this will not only increase the computational cost during the inference process, but epochs that are too fine grained may not enable us to observe larger/longer patterns in the data stream.

Because of the above, traditional DTMs that assume a fixed epoch length may not be able to capture emergence of latent topics well. In order to tackle these challenges, I design a novel **Incremental Multi-Scale Dynamic Topic Model (IMS-DTM)** [26], which be used to mine the latent topics in a dynamic corpus based on the evidences collected from multiple time scales. Unlike prior work, such as [47], I

consider multiple scales of the current epoch to infer current topics at multiple scales, and achieve this efficiently in an incremental fashion.

Improving Accuracy through Multi-Scale Inference. Using IMS-DTM, one can infer latent topics and their dynamics in multiple scales of time. More specifically, the current epoch's feature distribution prior is based on a weighted average of the past distributions at different scales. Since the impacts of the different time scales of the past are not known *a priori*, these weights are learned by analyzing the inter-dependencies among current topics and past data objects. Moreover, in order to discover dependencies of impact among short and long topics, different scales of current time epochs are also inferred. Since IMS-DTM considers topic interdependencies at multiple time scales, this improves the accuracy of the inference when such interdependencies are in play.

Efficient Multi-Scale Learning. An important related challenge is to prevent the multi-scale analysis from significantly increasing the DTM analysis cost. DTM based inference usually involves some form of *expectation maximization* (EM) approach [39] to discover latent patterns. Due to the inherent cost of EM, optimization techniques, such as Gibbs sampling [39], are often used to efficiently estimate joint feature distributions. Furthermore, acquiring the data for the above analytic methods itself is a great challenge and especially in the case of acquiring large scale data- and model-driven computer simulation ensembles.

1.3 Model Learning with Blackbox Model

As mentioned previously, simulation ensemble is the learning model that requires the most domain knowledge. A simulation is conducted by executing a complex model (usually it is black box to user) that, given appropriate inputs, can approximate patterns that would be observed in the real world. Therefore, the underlying function of

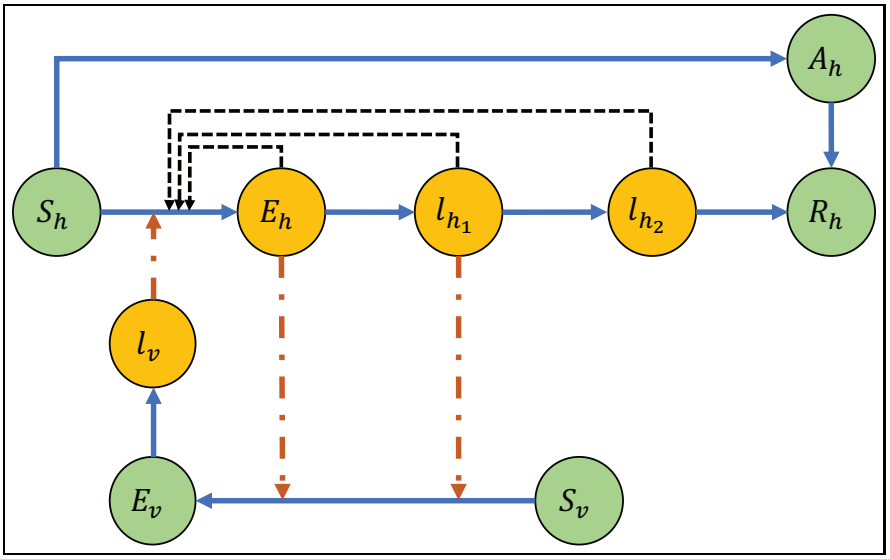


Figure 1.3: Zika epidemic model: parameters of the model control the transition rates among the disease states (e.g. susceptible, Sh; exposed, Eh) for humans and mosquito [34]

the blackbox (which is not visible nor accessible to users) could be itself exponential and hence the tasks involving learning the underlying pattern and obtaining the optimal solutions of it become hard to solve and the time complexity is highly depending on the hardness of the blackbox that one would like to investigate. Data- and model-driven simulations are increasingly critical in many application domains [60, 76, 27]. For example, for predicting the evolution of epidemics and assessing the impact of interventions, experts often rely on epidemic models (Figure 1.3), simulation software, such as GLEaM [84] and STEM [1], and simulation ensemble tools, such as EpiDMS [61].

Simulations have been adopted widely for the assessment of well-specified application scenarios, when decision making and knowledge discovery in the presence of incomplete knowledge are considered, decision makers usually need to generate an ensemble of stochastic realizations, requiring 1000s of individual simulation instances, each with different parameter settings corresponding to different, but plausible, scenarios. Naturally, these ensembles must reflect real-world data (such as epidemic mod-

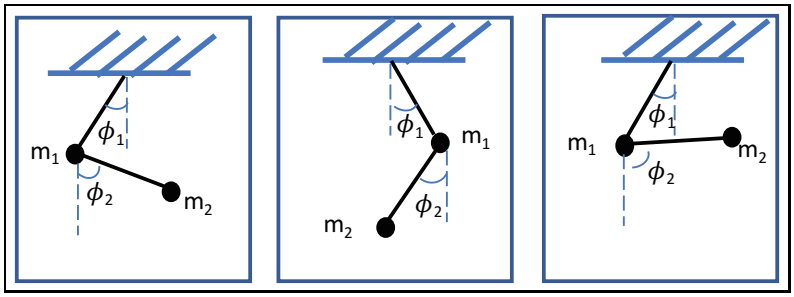


Figure 1.4: States of a multi-pendulum system

els, spatial/surface data, demographic data, mobility networks, evacuation models, and health-care and control intervention data and models), representing the relevant context.

As a consequence, obtaining and interpreting simulation ensembles to generate actionable results present several data challenges: (a) *Limited ensemble simulation budgets:* Since complex, inter-dependent parameters affected by complex dynamic processes have to be taken into account, *execution of simulation ensembles can be very costly* leading to *simulation budget constraints* that limit the number of simulations one can include in an ensemble. (b) *Inherent data sparsity of simulation ensembles:* While the size and complexity of a simulation ensemble can tax decision makers, we note that (however large the given simulation budget is), *a simulation ensemble is inherently sparse (relative to the space of potential simulations one could run)* and this constitutes a significant problem in simulation-based decision making.

Example Consider for example, the simple dynamical system, *double equal-length pendulum*, depicted in Figure 1.4: in this system there are five parameters that one can control: (a) the initial angle of the first pendulum ϕ_1, (b) the initial angle of the second pendulum ϕ_2, (c) the weight of the first bob m_1, (d) the weight of the second bob m_2, and (e) the gravity, g. It is easy to see that if we simply assume that for each parameter we consider, say, 20 distinct values, this would lead to $20^5 = 3200000$ possible simulation instances to potentially consider. $\diamond$

It is easy to see that, as the number of input parameters of a simulation increases, the number of potential situations one can simulate increases exponentially; thus, even for a relatively small number of parameters, any realistic simulation budget is likely to be much smaller than the possible space of all simulations – consequently, the naive approach of randomly sampling the simulation space is likely to lead into sparse ensembles that are difficult to accurately analyze.

Note that cost of simulations and sparsity of ensembles are not the only challenges. In addition, when creating an ensemble, we also need to take into account (c) *post-simulation interpretation and knowledge discovery:* Because of the complexities of key processes and the varying spatial and temporal scales at which they operate, experts often lack the means to drive conclusions from the simulation data generated by these ensembles. This leads to the need for data analytics on simulation ensembles to discover broad, actionable and interpretable patterns. As we see in Figure 1.5, however, the same set of observations may be explained with patterns of different degrees of **fit** and **complicacy**. Obviously, it is important that the patterns that are discovered to explain the simulations have a good fit with the data and also have a *complicacy appropriate for the given application.* For instance, in Figure 1.5, (1) the sinusoidal curve has a good fit to the simulation results, but may not be acceptable in applications where we only expect polynomial complicacy; (2) the cubic polynomial has a better fit to the simulation results, but has both a higher polynomial complicacy (which may or may not be acceptable in a given application) and more parameters to fix to achieve the fit.

Note that, as a further challenge, (d) especially for complex non-linear processes, the behavior of the *simulated process can vary across its input parameter space* (Figure 1.6), which means that different parts of the space may be better explained using different models, leading to heterogeneous simulation samples, and models, across the

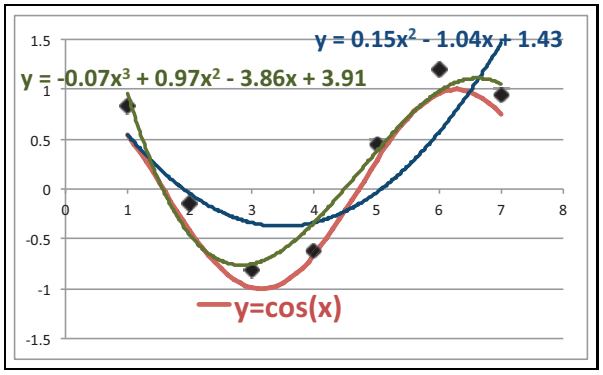


Figure 1.5: Three alternative models (with different degrees of fit and complicacy) for an ensemble with seven instances

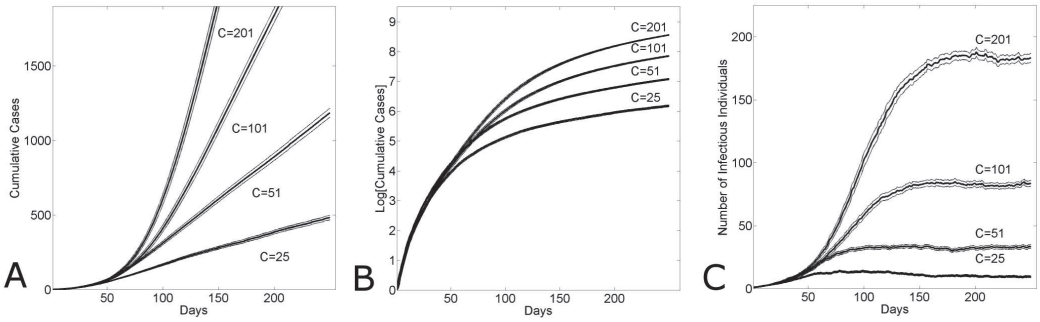


Figure 1.6: Epidemics often show different behaviors (exponential, sub-exponential, and linear) over time [51]

input parameter space.

These observations lead to the following critical question, which we aim to answer:

> *Given a parameter space and a fixed simulation budget, which simulation instances we should include in the ensemble to obtain models with good fit and low complicacy?*

Therefore, I developed a complicacy-guided parameter space sampling (CPSS) algorithm for knowledge discovery with limited simulation budgets. Unlike existing works, CPSS takes as input a *model dictionary* consisting of models with varying complexities. The CPSS algorithm splits the simulation parameter space in a top-down fashion, by incrementally and adaptively assigning simulation instances to different parts of the input space, in such a way that each resulting partition of the parameter

space is explained with a model with good fit and low complicacy. In particular, CPSS relies on a novel *coverage-weighted complicacy guided* enhancement for measures to rank candidate models and a novel *rank-stability based parameter space partitioning strategy* to identify simulation instances to execute. This implies that unlike purely fit-based approaches, CPSS avoids scheduling extensive simulations in difficult-to-fit regions of the parameter space, if the region can be explained with a much simpler model, requiring less simulation samples, possibly with a slightly lower fit. Instead, CPSS uses the simulation budget to seek simpler models in all regions of the parameter space CPSS achieves these by

- taking as input a model dictionary (including model complexities) and ranking these models based on (a) how well they explain the current samples and (b) how complex they are;
- using, as simulation assignment and partitioning conditions, a novel *rank-stability* based measure, which stops issuing new simulation instances when running more simulations is not likely to help differentiate the models in the model dictionary.

In addition, when the simulation system itself is evolving over time, sampling becomes even more complicated. As the simulations evolve over time, existing simulation ensembles may become outdated, and be insufficient to accurately describe and predict future events. Therefore, in dynamic contexts, one may need to refresh (parts of) the simulation ensembles such that they can better capture the evolving simulation state. Thus, it is important to identify new simulation instances that not only help obtain models that explain the existing system well, but also provide sufficient diversity to cover alternative potential futures for the simulated system.

Therefore, I developed a novel incremental *pivot guided parameter space sampling* (PGSS) method to address the above problem. PGSS adaptively assigns new simula-

tion instances to the input space as the simulation evolves. For each additional sample PGSS incrementally selects, it maintains the ability to predict more diverse futures and, at the same time, ensures that the sampled simulation instances accurately explain existing simulation instances. PGSS relies on (a) an *incremental maintenance* mechanism on the factor model of the simulation ensemble space to produce diverse predictions of the future, and (b) a novel *pivot based* sampling selection strategy that helps to identify the next simulation instances that should be executed to enable accurate description of the existing ensembles. The main contributions of PGSS are two-fold:

- the ability to maintain an orthonormal basis for the simulation ensemble space as new simulation instances arrive incrementally. The orthogonality property enables an informative model, which is robust to overfactorization, and at the same time, the non-overlapping factors are able to create models that can help identify diverse patterns.

- a pivot guided sample selection mechanism to incrementally and adaptively select the simulation instances to execute, such that the existing simulation ensembles can be accurately explained and, at the same time, we can gain better knowledge of the unexplored region of the simulation ensemble space.

Given all the challenges and the methods I developed to tackle them, I summarize the thesis outline in next section.

1.4 Thesis Outline

In table 1.1, I categorize my work for tackling the challenges mentioned above.

The rest of this thesis is organized as follows. Chapter 2 describes related works in the literature. I then discuss the incremental factor models in Chapter 3 and 4. In

	Challenges			
	Latent Factor Model - SVD	Latent Factor Model - NMF	Probabilistic Model - DTM	Blackbox Model
Solution	LWI-SVD	GI-NMF	IMS-DTM	CPSS, PGSS

Table 1.1: Categorization on my work for various challenges

Chapter 5, I present the dynamic probabilistic model. In Chapter 6 and 7, I formulate the parameter space sampling problems and present sampling strategies for static and dynamic simulation ensemble spaces. Finally, I conclude my thesis in Chapter 8.

Chapter 2

BACKGROUND AND RELATED WORK

In this chapter, I will introduce the literature of both factor models and probabilistic models, as well as the background on simulation ensembles and simulation sampling.

2.1 Dimensionality Reduction

Feature selection and dimensionality reduction techniques [17, 91] usually involve some (often linear) transformation of the vector space containing the data to help focus on a few features (or combinations of features) that best discriminate the data in a given corpus. These include the Principle Component Analysis (PCA) [7], Karhunen-Loeve Transform, KLT [55], and singular value decomposition, SVD [29]), and non-negative matrix factorization (NMF) [17].

Consider for example a feature-object matrix X representing the content of a set of data objects; X_{ij} denotes the occurrence of feature j in object i. SVD decomposes X into U, S, and V matrices (such that $X = USV^T$), a diagonal core matrix (S) describes the importance of the r features and, thus, less important features can be identified and dropped to obtain a low rank approximation. The SVD low rank approximation is unique and optimal. However, a major drawback of SVD is that it might produce negative components which are not ideal for interpreting the data representations. Furthermore, the factor matrices, U and V, are usually dense (even the input data is sparse); as a result the singular value decompositions of large data matrices are often impossible due to memory limitations. While the Karhunen-Loeve Transform (KLT) works somewhat differently (and is optimal in preserving the

variance – i.e., minimizes the loss in the discrimination power in the data), it also returns two factor matrices and a core matrix and largely shares (advantages and) disadvantages of SVD.

2.2 Singular Value Decomposition

Singular value decomposition (SVD [29]) of a data feature matrix A is of the form $A = USV^T$, where the r *orthogonal* column vectors of U form an r dimensional basis in which the n data objects can be described. Also, the r *orthogonal* column vectors of V (or the rows vector of V^T) form an r dimensional basis in which the m features can be placed. These r dimensions are referred to as the *latent variables* [81] or the *latent semantics* of the database [29]. Intuitively, the columns of U can be thought of as the eigen-objects of the data, each corresponding to one independent concept/cluster, and the columns of V can be thought of as the eigen-features of the collection, each, once again, corresponding to a concept/cluster in the database. In other words, SVD can be used for co-clustering both data-objects and features simultaneously. The $r \times r$ diagonal matrix S, can be considered to represent the strength of the corresponding latent concepts in the database: the amount of error caused by the removal of a concept from the database is proportional to the corresponding singular value.

2.3 Incremental SVD

SVD is computationally costly and therefore a naive implementation does not match the real-time needs of scenarios where data evolve continuously: decomposition of an $n \times m$ matrix requires $O(n \times m \times min(n, m))$ time. While there are various on-line techniques, these are often slow or inaccurate. For example, one of the fastest techniques, SPIRIT [68] focuses on row insertions and cannot directly handle row deletions or column insertions/deletions. While a forgetting factor can be introduced

to discount old objects, it cannot immediately reflect the properties of the removed entries on the decomposition. Moreover, since SPIRIT primarily considers data insertions and deletions, it is not applicable in situations where features of interest themselves evolve with the data (examples include weights of tags extracted from data and proximity to the hubs within an evolving network).

Other incremental SVD algorithms, such as [15, 20, 41, 40, 55], operate on an existing SV decomposition by folding-in new data and features into an existing (often low-rank) SVD; algebraic matrix manipulation techniques are used to rewrite the new SV decomposition matrices in terms of the old SV decomposition and update (including downdating) matrices. [15] showed that a number of database updates (including removal of columns) can all be cast as additive modifications to the original $n \times m$ database matrix, A. These updates then can be reflected on the SVD in $O(nmr)$ time as long as the rank, r, of the matrix A is such that $r \leq \sqrt{min(p, q)}$, where p is the number of new rows and q is the number of new columns. In other words, as long as the latent dimensionality of the database is low, the singular value decomposition can be updated in linear time. [15] further showed that the update to SVD can be computed in a single pass over the data matrix making the process highly efficient for large data. This and other existing algorithms can nevertheless be slow for many real-time applications.

2.4 Non-Negative Matrix Factorization

Non-Negative Matrix Factorization (NMF) is another way of decomposing a data matrix into several factor matrices. It overcomes some of the shortcomings that SVD has, for example, factor matrices are non-negative and these non-negative factor matrices are usually sparse. The non-negativity constraints enable probabilistic interpretation of the results and discovery of generative models: intuitively, NMF

provides basis vectors and participant weights for each data dimension. More formally, given an $n \times m$ data matrix X, NMF seeks factor matrices, ($n \times r$ matrix) W and ($r \times m$ matrix) H, such that

$$\begin{aligned} & min \left\| X_{n\times m} - W_{n\times r} \times H_{r\times m} \right\|_F^2 \\ & \text{such that} \quad W_{n\times r} \geq 0 \quad \wedge \quad H_{r\times m} \geq 0 \end{aligned} \tag{2.1}$$

For a given value of $r \ll n, m$, the two matrices, W and H, define a low rank r approximation of the original matrix X, such that each entry, $x_{i,j}$ in the original matrix, X, is an additive linear combination of the feature strengths $(W_{i,1}, \ldots, W_{i,r})$ in the first factor matrix, W, weighted by the contributions $(H_{1,j}, \ldots, H_{r,j})$ of those features described by the corresponding column vector in the second factor matrix, H. While r is often assumed to be given, there are non-parametric methods, where r is determined from data [78].

2.5 Tensor Decomposition and Incremental Tensor Analysis

Both SVD and NMF work great for low dimensional input matrices, i.e. 2D array. However, with the increasing of volume of the data, higher dimensional data becomes more and more critical. High-dimensional arrays (known as tensors) are commonly used for representing multi-modal data, in order to enable data analysis and knowledge discovery such as clustering, classification, trend detection, anomaly detection for these high dimensional data, one way to obtain representative models from such sparse tensors is to seek principal patterns through tensor decomposition techniques, such as CANDECOMP [19], PARAFAC [42] (a.k.a. CP decomposition), and Tucker decomposition [83]. In CP decomposition, usually an input tensor $\mathcal{X}$ is factorized into factor matrices with r rows (r is a user input that can be treated as the rank of decomposition), one for each mode of the input tensor, besides the factor

matrices, it also produces a core tensor, which describes the spectral structure of the input tensor. If one takes each row from each factor matrix and multiply them together, this operation results in a set of rank-one tensor where the summation of them approximate the original input tensor.

One key challenge within the tensor decomposition based approach is that they are computationally complex, and therefore, they have issues scaling to high dimensions, large amount of data and online applications if applied naively. Besides the computational challenges, such as the high dimensionality that sometimes leads to sparsity in the data (as one is always hard to observe everything to construct a complete dense tensor), memory blow-up when handling large tensors in a streaming data environment, and noise in the tensor which prevents efficient knowledge discovery.

There are various existing work that tackle challenges in tensor based approaches, such as for efficiency challenges when applying tensor decomposition in an evolving environment, the authors in [82] provided a framework to incrementally maintain the factors when the data is evolving without the need to recompute of the whole tensor decomposition. In [46], the authors also avoid redecompose the tensor from scratch and introduced a block incremental decomposition algorithm that only revises the decompositions of the tensors that contain updated data. For other challenges, such as the noise handling in tensors, in [56], the authors provide a grid-based two-phase decomposition framework to tackle the potential non-uniformities in the noise distribution using a sample assignment strategy based on a priori knowledge about the noise profiles.

One thing to note that the resulting factors from CP decomposition is not necessarily orthonormal, unlike SVD that always produces orthonormal factors. Orthogonality sometimes is important in certain applications, since the rank-one tensors could be treated as a set of "weak" model that approximate the original input tensor

in some degrees, and each of them can be used to describe the underlying pattern of the original tensor. However, if they are similar, i.e. contain redundancy, it would be hard to find distinct patterns and thus degrade the power of these models provide. In order to tackle this issue, in [2], in addition to minimize the approximation errors from cp decomposition, the authors added an angular constraints that measures the total cosine redundancy within the factor matrices and try to optimize it together with error term, this helped reduce the redundancy. In [43], the authors proposed a diversified, sparse tensor factorization method, by also introducing an angular penalty term and an $L2$ regularization term on the factors. However, the revised objective function with additional constraints complicate the optimization process. Furthermore, the factors are still not fully orthonormal, which means there is still a certain degree of redundancy in them.

2.6 Dynamic Topic Models and its Extensions

Despite the factorization models mentioned above, during the past few years, researchers have paid increasing attention to Bayesian data analysis [36], and they are sometimes referred as "topic models". In topic models, such as LDA [14], for example, the observations are expressed as generative process of hidden variables. Given a set of observations, posterior inference is performed by computing the conditional distribution of the hidden variables.

Dynamic Topic Models [11, 89, 45] extend the basic topic modeling technique into a dynamic setting, where the topic distribution and word distribution priors are evolving. The main difference between a static topic model and a dynamic topic model is that the static topic model assumes that all the documents are exchangeable for the same set of topics; in contrast, in dynamic topic models, this assumption does not hold because the documents are coming in a streaming manner and the order

of documents reflects the evolution of the topics. While the original DTM [11] is unsupervised, more recent extensions, such as [13], proposed supervised versions by adding a response variable associated with each document as class label; given this, the documents and response are jointly modeled.

Gibbs sampling [39] and variational inference [10] are two well known techniques in solving topic model problems efficiently. Gibbs sampling is widely used in Bayesian inference for topic models. It enables obtaining samples that are approximated from a given joint probability distribution over more than one random variable. The main idea of Gibbs sampling is that given a multivariate distribution, solving the problem of integrating over a joint distribution is harder than simply sampling from the conditional distribution. Due to the effectiveness of Gibbs samplers, various Gibbs samplers have been proposed [38, 77].

Lots of recent work, extends DTMs in different ways. For example, [66] considers multiple data sources contributing to the dynamic topic model. In [86], authors avoid discretization of time and they treat time continuously. [4] proposes a streaming and distributed unsupervised inference method based on topic modeling of user profiles to support recommendation generation. [9] scales up the inference process in the DTMs by a fast and parallelizable algorithm. In a work most related to ours, [47] considers multiple time scales of the past and shows that this can improve the predicting power of the DTM. However, [47] does not consider the fact that also the current epoch (i.e. "now") can be considered at multiple time scales and this would not only enable the user to study topics at multiple time scales simultaneously, but could also be used to improve the overall efficiency of the multi-scale dynamic topic modeling process through incremental processing.

2.7 Simulation Ensembles and Parameter Space Sampling

Ensemble simulations are increasingly critical in supporting decision making for diverse applications, such as energy [27], or epidemics[76, 61]. Yet, (a) designing an ensemble simulation that appropriately covers the input parameter space [22] and (b) interpreting the results of the ensemble simulation[60, 76], are both challenging.

Two main approaches can be adopted in the aim of the first objective: single stage and sequential approaches. Single stage approaches find their root in design of experiments [32, 49] where, given a simulation budget, samples are allocated in a single iteration in the attempt to minimize the variance of the subsumed model (such as general linear models and Gaussian processes). Examples are the commonly used latin hypercubes and uniform sampling. These approaches tend to assume a unique response model form throughout the input parameter space. Concerning sequential approaches, budget constraints and costs of simulations are being taken into account in simulation instance selection [22, 70, 71]. In [22], the authors proposed a time dilation scheme, where, through time dilation, more attention is driven towards more interesting parameter settings. More specifically, the proposed optimal computing budget allocation (OCBA) algorithm attempts to maximize probability of correct selection (PCS) given the budget constraints. While the method is dedicated to optimization, similar concepts can be applied when estimation is of concern. However, the OCBA method suffers from two drawbacks: (a) in terms of time, it is hard to control computational resources and (b) the convergence rate may be slow. In [70], the authors proposed an improvement which combines time dilation and optimal computing budget allocation. Intuitively, this is similar to identifying interesting parameter settings and, at the same time, providing the number of replications and simulations to run. In [71], the authors have shown that there exists a positive correlation between

the asymptotic variance of the estimators and the required computation resource to achieve a certain PCS. An alternative approach is to construct a response surface, usually using Gaussian Process (GP), and then, based on the initial sampled points, to adopt the Modified Expected Improvement (mEI) to create a trust region and estimate a local model [69]. Based on the trust region and new samples, the algorithm is able to decide whether it is needed to augment or shrink the region(s).

Authors in [57] tackled the inherent sparsity of the simulation ensembles (relative to the space of potential simulations one can run) through a partition-stitch sampling scheme, which divides the parameter space into subspaces to collect several lower modal ensembles and complemented this with a novel Multi-Task Tensor Decomposition (M2TD) technique which helps effectively and efficiently stitch these subensembles back together. While [57] also aims to address the sparsity challenge in simulation ensembles, unlike this paper, it (a) focuses primarily on supporting tensor-based analytics and (b) ignores the complexities of the models generated through M2TD.

The above methods also suffer from a main drawback: they do not consider system evolution and its impact on the selection of simulation samples and the diversified future of possible evolution of simulation ensembles.

Authors in [90] proposed cascaded compression sampling method to achieve linear-cost matrix sketching, where they proved that the approximation error is bounded when sampling k columns and k rows of the matrix X, with the intersection W. Therefore, the selected rows and columns of X should be "representatives" in the low-rank space. However, the sampling mechanism requires knowledge of k complete rows and columns which is not practical in the scenario of dynamic simulations.

Chapter 3

LOW-RANK, WINDOWED, INCREMENTAL SINGULAR VALUE DECOMPOSITION ON DATA STREAM

3.1 Introduction

Singular Value Decomposition (SVD) is computationally costly and therefore a naive implementation does not scale to the needs of scenarios where data evolves continuously. While there are various on-line analysis and incremental decomposition techniques, these may not accurately represent the data or may be slow for the needs of many applications. To address these challenges, I propose a *Low-rank, Windowed, Incremental SVD* (LWI-SVD) algorithm, which (a) leverages efficient and accurate *low-rank approximations* to speed up incremental SVD updates and (b) uses a *window-based* approach to aggregate multiple incoming updates (insertions or deletions of rows and columns) and, thus, reduces on-line processing costs. I also present an *LWI-SVD with restarts* (LWI2-SVD) algorithm which leverages a novel highly efficient *partial reconstruction* based change detection scheme to support timely refreshing of the decomposition with significant changes in the data and prevent accumulation of errors over time. Experiment results, including comparisons to other state of the art techniques on different data sets and under different parameter settings, confirm that LWI-SVD and LWI2-SVD are both efficient and accurate in maintaining decompositions.

3.2 Problem Formulation

At time stamp i, given a set of n data tuples $D_i = \{t_{i,1}, t_{i,2}, ..., t_{i,n}\}$, each with a set, F_i, of m features. We are also given the set, L_i, containing the r latent semantics of D_i (and their weights). As time moves, new tuples arrive and some of the existing tuples expire: at the next time stamp, t_{i+1}, the tuple set is $D_{i+1} = (D_i \backslash \Delta D_{i+1}^-) \cup \Delta D_{i+1}^+$, where ΔD_{i+1}^- are the tuples that expired and ΔD_{i+1}^+ are the new tuples that arrived. Moreover, at time $(i+1)$, there is a new set, F_{i+1}, of features, where $F_{i+1} = (F_i \backslash \Delta F_{i+1}^-) \cup \Delta F_{i+1}^+$, where ΔF_{i+1}^- are features that are not of interest anymore and ΔF_{i+1}^+ are the new features of interest. Our goal is to quickly obtain L_{i+1} containing the r latent semantics corresponding to time instance, $i+1$, and efficiently maintain these r latent semantics as time further moves.

3.2.1 Basic Incremental SVD

Let us be given an $n \times m$ data matrix $X = U_x S_x V_x^T$ and an $n' \times m'$ updated data matrix $X' = X + \Delta$, where Δ is a $max(n, n') \times max(m, m')$ change matrix. Note that if X' has larger dimension than X, X is padded with $n' - n$ rows of zero and $m' - m$ columns of zero to match the dimension of Δ, the removal of rows and columns are modeled by additions that result in zeroing of the corresponding rows and columns (which are then dropped from the matrix). Let us further assume that the change matrix Δ can be decomposed into $\Delta = AB^T$. Note that the matrix X' can be rewrited as

$$X' = X + AB^T = \begin{bmatrix} U_x & A \end{bmatrix} \begin{bmatrix} S_x & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} V_x & B \end{bmatrix}^T. \tag{3.1}$$

Given these, [15] incrementally maintains SVD as follows:

QR Decompositions

Let us also define Q_A as the orthogonal basis of $(I-U_xU_x^T)A$ and Q_B as the orthogonal basis of $(I-V_xV_x^T)B$. Both Q_A and Q_B can be obtained through QR decomposition [8] of $(I - U_xU_x^T)A$ and $(I - V_xV_x^T)B$:

$$Q_AR_A = (I - U_xU_x^T)A; \qquad Q_BR_B = (I - V_xV_x^T)B \tag{3.2}$$

Here Q_A and Q_B are orthogonal matrices and R_A and R_B are upper-triangular. It is easy to see, through basic matrix algebra, that the following holds:

$$\begin{bmatrix} U_x & A \end{bmatrix} = \begin{bmatrix} U_x & Q_A \end{bmatrix} \begin{bmatrix} I & U_x^TA \\ 0 & R_A \end{bmatrix} \tag{3.3}$$

$$\begin{bmatrix} V_x & B \end{bmatrix} = \begin{bmatrix} V_x & Q_B \end{bmatrix} \begin{bmatrix} I & V_x^TB \\ 0 & R_B \end{bmatrix} \tag{3.4}$$

Moreover, by substituting Equations 3.3 and 3.4 into Equation 3.1, we can get

$$X' = X + AB^T = \begin{bmatrix} U_x & Q_A \end{bmatrix} K \begin{bmatrix} V_x & Q_B \end{bmatrix}^T \tag{3.5}$$

where K is equal to

$$K = \begin{bmatrix} I & U_x^TA \\ 0 & R_A \end{bmatrix} \begin{bmatrix} S_x & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & V_x^TB \\ 0 & R_B \end{bmatrix}^T \tag{3.6}$$

$$= \begin{bmatrix} S_x & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} U_x^TA \\ R_A \end{bmatrix} \begin{bmatrix} V_x^TB \\ R_B \end{bmatrix}^T \tag{3.7}$$

Matrix K

Let us remember that X is an $n \times m$ matrix and X' is an $n' \times m'$ matrix. Given this

- if $n \geq n'$ and $m \geq m'$, K is a matrix of size $(n+1) \times (m+1)$. This is because, if $n \geq n'$, then U_x is an orthogonal matrix and $U_x U_x^T$ is equal to I. Consequently, $Q_A R_A = (I - U_x U_x^T)A = 0$ and this implies that R_A is simply 0. The same is true for R_B.
- if $n < n'$ and $m < m'$, K is a matrix of size $n' \times m'$. In Section 3.3.3, the shape K takes in this case and the resulting properties are described in detail.
- if $n \geq n'$ and $m < m'$, K is a matrix of size $(n+1) \times m'$.
- if $n < n'$ and $m \geq m'$, K is a matrix of size $n' \times (m+1)$.

Using the Decomposition of K to Obtain the Decomposition of X'

Let us consider the SV decomposition of K; i.e., $K = U_K S_K V_K^T$. Equation 3.1 can be rewritten [15] as

$$X' = X + AB^T = \left(\begin{bmatrix} U_x & Q_A \end{bmatrix} U_K \right) S_K \left(\begin{bmatrix} V_x & Q_B \end{bmatrix} V_K \right)^T \tag{3.8}$$

giving us the SVD of the new tuple matrix, X'.

The challenge, of course, is to obtain the matrices, Q_A and Q_B, and the SV decomposition of K efficiently. In order to keep the complexity down, [15] suggests that A and B should be taken as combination of simple column vectors so that AB^T can be the sum of multiple rank-1 matrices. This, however, may be a significant constraint in real-applications where the change matrix Δ itself can have a large size, indicating great amount of rank-1 matrices it produces and updating a sequence of rank-1 matrix is not effective as treating them as a whole. In the next section, I

will describe how to relax this assumption of [15] without impacting efficiency and accuracy.

3.3 LWI-SVD

We now present our key ideas for efficient incremental SVD operations. As described above, this involves efficiently searching for matrices, Q_A and Q_B, and the SVD of K.

3.3.1 Efficiently Obtaining Q_A and Q_B

As described above, Q_A is the orthogonal basis of $(I - U_x U_x^T)A$ and Q_B is the orthogonal basis of $(I - V_x V_x^T)B$. These can be obtained using two expensive QR decomposition operations for both Q_A and Q_B. One way to reduce the number of QR decomposition operations would be to seek a decomposition of Δ where $X' = X + \Delta = AA^T$; i.e., $A = B$. However, not all Δ will have such a convenient decomposition. When Δ is negative definite, it cannot be written as the format of $A \times B$ where $A = B$.

Instead, the cost of the overall QR decomposition step can be reduced by setting A to the identity matrix I and setting B^T to Δ. This does not lose any generality on the algorithm since Δ (B^T) can be any matrix. When we do this, since $A = I$, it would also be the case that $Q_A = \begin{bmatrix} 0 \\ I \end{bmatrix}$. Therefore, we need only one QR decomposition. What is more, if the Δ only reflect a small amount of data insertions and deletions, then it will be a sparse matrix with last few rows and columns of nonzero values. This lead to efficient computation of $(I - V_x V_x^T)B$ and $V_x^T B$ by block matrix multiplication. Let's first find the zero block of B when it is data insertion. Then Δ (B) is a $n' \times m'$ matrix with a block of zero values on the first $n \times m$ position. We can rewrite B as:

$$B = \begin{bmatrix} 0 & B_1 \\ B_2 & B_3 \end{bmatrix}$$

Then, we can divide $(I - V_x V_x^T)$ and V_x^T into the same block size as B. For example, we can rewrite V_x^T as

$$V_x^T = \begin{bmatrix} V_{x0}^T & V_{x1}^T \\ V_{x2}^T & V_{x3}^T \end{bmatrix}$$

Then, the multiplication of $V_x^T B$ becomes

$$V_x^T \times B = \begin{bmatrix} V_{x1}^T \times B_2 & V_{x0}^T \times B_1 + V_{x1}^T \times B_3 \\ V_{x3}^T \times B_2 & V_{x2}^T \times B_1 + V_{x3}^T \times B_3 \end{bmatrix}$$

Note that, the multiplication of V_{x0}^T and the corresponding block of B is avoided since the corresponding block of B is all zeros. Also, the other part of V_x^T and B are small size thin matrices. Thus, the multiplication of $V_x^T \times B$ can be done very efficiently. The same applies to $(I - V_x V_x^T) \times B$ and when the data are deleted.

3.3.2 Efficiently Decomposing K

The next challenge is to obtain the singular value decomposition of the matrix, K. Performing SVD on K directly would be costly as the SVD operation is expensive. However, proved next, in Section 3.3.3, in the presence of row and column insertions, K takes a special structure:

$$K = \begin{bmatrix} S_x & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} U_x^T A \\ R_A \end{bmatrix} \begin{bmatrix} V_x^T B \\ R_B \end{bmatrix}^T = \begin{bmatrix} S_x & \Pi \\ \Phi & \Gamma \end{bmatrix}.$$

More specifically, in the presence of insertions, (a) since S_x is diagonal, K is mostly sparse, and (b) it is shaped like an arrow: (aside from the diagonal) there are non-

zeros only on its last rows and columns. We verify these next.

3.3.3 Shape of K

Let X be an $n \times m$ matrix and X' be an $n' \times m'$ matrix. In Section 3.2.1, we have seen that K is either of size $n' \times m'$, $(n+1) \times (m+1)$, $(n+1) \times m$, or $n' \times (m+1)$, depending on whether the numbers of rows and columns increase or decrease when the data matrix transforms from X to X'. Let us further assume that $n \leq m$ and $m' > m$ and $n' > n$, which is rows and columns insertion. As already discussed before, let us set $A = I_{n'}$ and $B^T = \Delta$, where $A \in \mathbb{R}^{n' \times n'}$, $B \in \mathbb{R}^{m' \times n'}$ and $\Delta \in \mathbb{R}^{n' \times m'}$ so that AB^T is equal to the update matrix Δ. Finally, let SVD of X be $X = U_x S_x V_x^T$, or simply $X = USV^T$.

Given the fact that $X' = X + \Delta$, we can also deduce that $X' = U'SV'^T + \Delta$, where

$$U' = \begin{bmatrix} U \\ \\ 0 \end{bmatrix} \in \mathbb{R}^{n' \times n} \text{ and } V' = \begin{bmatrix} V \\ \\ 0 \end{bmatrix} \in \mathbb{R}^{m' \times m}.$$

Intuitively, U and V are augmented by padding $n' - n$ rows of zeros to U and $m' - m$ rows of zeros to V to make it compatible with Δ. This padding gives us

$$X' = \begin{bmatrix} X & 0 \\ \\ 0 & 0 \end{bmatrix} + \Delta = U'SV'^T + \Delta.$$

Secondly, using a similar zero-padding, we can get the following equalities:

$$(I_{n'} - U'U'^T)A = \begin{bmatrix} 0 & 0 \\ \\ 0 & I_{n'-n} \end{bmatrix} \tag{3.9}$$

$$(I_{m'} - V'V'^T)B = \begin{bmatrix} 0 \\ B_{m'-m} \end{bmatrix} \tag{3.10}$$

The right hand side of Equation 3.9 has $n' - n$ independent columns and, thus, it has a simple QR decomposition:

$$Q_A = \begin{bmatrix} 0 \\ I_{n'-n} \end{bmatrix} \text{ and } R_A = \begin{bmatrix} 0 & I_{n'-n} \end{bmatrix}$$

Since the right hand side of the Equation 3.10 consists of 0s except for the last $m'-m$ rows, the QR decomposition of the left hand side will be such that $Q_B \in \mathbb{R}^{(m'-m)\times m}$ and $R_B \in \mathbb{R}^{(m'-m)\times n'}$. Let us further partition R_B into two,

$$R_B = \begin{bmatrix} R_{B1} & R_{B2} \end{bmatrix},$$

where $R_{B1} \in \mathbb{R}^{(m'-m)\times n}$ and $R_{B2} \in \mathbb{R}^{(m'-m)\times(n'-n)}$.

Given the above, we can rewrite the matrix, K, as

$$K = \begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} U'^T A \\ R_A \end{bmatrix} \begin{bmatrix} V'^T B \\ R_B \end{bmatrix}^T$$

where

$$\begin{bmatrix} U'^T A \\ R_A \end{bmatrix} = \begin{bmatrix} U^T & 0 \\ 0 & I_{n'-n} \end{bmatrix}$$

$$\begin{bmatrix} V'^T B \\ R_B \end{bmatrix}^T = \begin{bmatrix} 0 & Y \\ R_{B1} & R_{B2} \end{bmatrix}^T.$$

Here Y is a $m \times (n' - n)$ matrix. Note that this can be further rewrited as:

$$\begin{bmatrix} U'^T A \\ R_A \end{bmatrix} \begin{bmatrix} V'^T B \\ R_B \end{bmatrix}^T = \begin{bmatrix} U'^T A \\ R_A \end{bmatrix} \begin{bmatrix} (V'^T B) \\ R_{B1} \quad R_{B2} \end{bmatrix}^T$$

as

$$\begin{bmatrix} U^T & 0 \\ 0 & I_{n'-n} \end{bmatrix} \begin{bmatrix} 0 & Y \\ R_{B1} & R_{B2} \end{bmatrix}^T = \begin{bmatrix} 0 & U^T R_{B1}^T \\ Y^T & R_{B2}^T \end{bmatrix}.$$

Thus, K simplifies to

$$K = \begin{bmatrix} S & U^T R_{B1}^T \\ Y^T & R_{B2}^T \end{bmatrix} = \begin{bmatrix} S & \Pi \\ \Phi & \Gamma \end{bmatrix}.$$

This confirms that when $m < m'$ and $n < n'$, K is shaped like an arrow: it is diagonal, except for the last $n' - n$ rows and last $m' - m$ columns. This, however, is not true when $m \geq m'$ or $n \geq n'$; in this case K can be a dense matrix, with its last row and columns equal to 0. In the rest of this section, especially when $m < m'$ and $n < n'$, we can leverage K's specific structure (sparse, arrow-like) to quickly obtain a highly-accurate approximate decomposition, $K \sim \hat{U}\hat{S}\hat{V}^T$ and use it instead of the exact decomposition $K = U'S'V'^T$.

3.3.4 Decomposition of K through Pivoted QR

Pivoted QR Factorization. Let E be a matrix. A pivoted QR factorization of E has the form $EP = Q_e R_e$ where P is a permutation matrix, Q_e is orthonormal and R_e is upper triangular. [8] has shown that a rank-k approximation can be obtained efficiently through a pivoting process where columns of E are considered one at a time and used to compute an additional column of Q_e and row of R_e. The k^{th} round of the process leads to a rank-k approximation of the pivoted QR factorization of E.

In particular, let us assume that given a QR decomposition of the form $F = Q_f R_f$ and need to compute QR decomposition of $[F\ a]$ for some column vector a:

$$[F\ a] = [Q_f\ q] \begin{bmatrix} R_f & \epsilon \\ 0 & \rho \end{bmatrix}$$

The rank-k approximation can be obtained efficiently by the quasi-Gram-Schmidt method, which further eliminates the need to store dense Q_f matrices [8]: the quasi-Gram-Schmidt process can be applied successively to columns of a given input matrix E to produce a pivoted QR factorization for E.

Low-Rank Decomposition of K. Let us assume that we are targeting a rank-k decomposition of K. We first sample k columns to obtain column-sample matrix $\mathcal{C}$; we can then sample k columns from K^T to obtain a row-sample matrix $\mathcal{R}^T$. We then apply the QR decomposition with column pivoting to $\mathcal{C}$ and $\mathcal{R}^T$ to obtain upper triangular matrices, R_c and R_r.

The sampling is done by selecting the longest row and column vectors. We note that when $m < m'$ and $n < n'$, K is not only sparse, but also has an arrow-like shape:

$$K = \begin{bmatrix} S_x & \Pi \\ \Phi & \Gamma \end{bmatrix},$$

where the $n \times m$ matrix S_x is diagonal, whereas $n \times (m' - m)$ matrix Π, $(n' - n) \times m$ matrix Φ, and $(n' - n) \times (m' - m)$ matrix Γ are potentially dense as discussed in Section 3.3.3. As a result, the sampling is *arrow-sensitive* in the sense that it focuses on the last few rows and columns: The sampled columns usually come from the first few columns (which contain the largest singular values at the top-left corner of the matrix) and the last few columns, which contain entries from the dense, $\begin{bmatrix} \Pi \\ \Gamma \end{bmatrix}$. Similarly, the sampled rows come from the first few rows (which contain large singular values in S_x) and the last few rows from $\begin{bmatrix} \Phi & \Gamma \end{bmatrix}$.

Algorithm 1 LWI-SVD.

Input:

The Base Matrix, X, and its SV decomposition $U_x S_x V_x^T$;

The update matrix, $\Delta = AB^T$, corresponding to a window of updates;

Target rank, r;

Output:

The new SVD results, U'_x,S'_x, and V'_x;

1: Calculate factors R_A and R_B in Equation 3.2 which, as discussed in Section 3.3.1, involves a QR Decomposition and several matrix multiplications;

2: Calculate the matrix K in Equation 3.7;

3: Obtain the low-rank (rank-r) decomposition of K into $K = U_K S_K V_K^T$;

4: Combine the factors as shown in Equation 3.8 to obtain rank-r decomposition U'_x, S'_x, and V'_x;

5: **return** U'_x, S'_x, and V'_x;

Given these, to obtain a decomposition of K, we need to find a matrix H such that $\|K - \mathcal{C}H\mathcal{R}^T\|$ is minimized. According to [80], the value of H which minimizes this can be computed as

$$(R_c^{-1}R_c^{-T})(\mathcal{C}^T K \mathcal{R})(R_r^{-1}R_r^{-T}).$$

. Thus, we can rewrite $\mathcal{C}H\mathcal{R}^T$ as

$$(\mathcal{C}R_c^{-1})(R_c^{-T}\mathcal{C}^T K \mathcal{R} R_r^{-1})(R_r^{-T}\mathcal{R}^T).$$

If we further set $W = R_c^{-T}\mathcal{C}^T K \mathcal{R} R_r^{-1}$ and decompose W into $W = U_w S_w V_w^T$, then we can obtain the SV decomposition of K as $K = U_K S_K, V_K^T$, where

$$U_K = \mathcal{C}R_c^{-1}U_w, \quad V_K^T = V_w^T R_r^{-1} R_r^{-T}, \text{ and } S_K = S_w,$$

where U_K and V_K are orthonormal and S_K is diagonal. While this process also involves an SV decomposition step involving W, since W is a much smaller, $k \times k$, matrix, its decomposition is much faster than the direct decomposition of K.

3.3.5 Pseudocode of LWI-SVD

Algorithm 1 provides the pseudo-code of the proposed Low-rank, Windowed, Incremental Singular Value Decomposition (LWI-SVD) algorithm for incrementally maintaining the SVD of an evolving matrix X. The LWI-SVD algorithm has a smaller approximation error than other algorithms, such as SPIRIT [68], yet is also much faster than optimal as well as the basic incremental SVD [15] algorithms. As in any incremental approximate algorithm, in which each step takes the output of the previous step as its input, there is a likelihood that errors will accumulate over time and the reconstruction error relative to the actual matrix will reach an unacceptable rate. To prevent errors to accumulate, in the next section I propose a novel *LWI-SVD with Restart* (LWI2-SVD) algorithm which restarts the SVD by performing a fresh SVD on the current data matrix.

3.4 LWI2-SVD: LWI-SVD with Restart

In this section, I propose a novel *LWI-SVD with Restart* (LWI2-SVD) algorithm which is built on LWI-SVD and punctuates the incremental SVD sequence by occasionally performing a full SVD on the current data matrix. Obviously, there is a direct, positive correlation between the frequency of restarts and the overall accuracy of the LWI2-SVD algorithm. Unfortunately, however, there is also a strong positive correlation between the cost of LWI2-SVD and the frequency of restarts. Therefore, restart rate should be such that the process is restarted only when the costly SVD is in fact needed to help reduce the overall error.

3.4.1 Types of Errors

We see that there are two distinct types of errors:

- *Accumulated approximation errors (and periodic restarts):* The first type of error that accumulates over time is due to the various approximation terms, including the low-rank approximation of K as discussed in Section 3.3.2. While the absolute value of this error will be different from one iteration of the algorithm to the next, its long term behavior will be roughly constant. Therefore, this type of accumulated approximation errors are best dealt with *periodic restarts*.

- *Error bursts due to structural changes in the input data (and on-demand restarts):* The second type of error in the incremental SVD occurs when there is a significant structural (or spectral) change in the data, necessitating large changes in the SVD. Since the incremental process described in Section 3.3 assumes that the changes are relatively small, a significant structural change in the factor matrices, U_x and V_x, or the core matrix S_x may not be correctly captured, resulting in a large burst of reconstruction error. These bursts are best dealt with *on-demand restarts* that are triggered through a change detection process that tracks the updates to identify when major structural changes in the data occur.

Figure 3.1 shows an example run with and without restarts. Note that without the restarts errors continuously accumulate due to structural changes in the data. Restarts (both periodic and on-demand) can limit the accumulation of errors. Error accumulations due to approximations generally show a regular behavior and the frequency with which periodic restarts are scheduled can be set empirically. The structural changes in the data, however, do not necessarily have a regular behavior; therefore, the challenge is to quickly and efficiently detect the structural changes in the data. We will discuss this next.

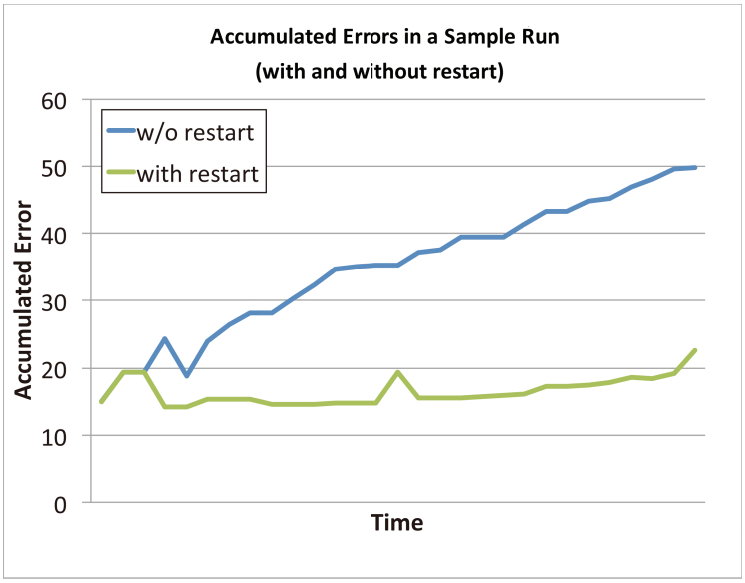


Figure 3.1: Example runs with and without restarts

3.4.2 Change Detection through Partial Reconstruction

In order to detect major structural changes in the data we need to measure or estimate the reconstruction errors. The naive way to achieve this would be to reconstruct the entire matrix from the incrementally maintained decomposition and compare the reconstructed matrix to the ground truth (which is the actual, revised data matrix). If the difference is high, it means that due to some structural changes, the incrementally maintained decomposition deviated from the true decomposition of the matrix. Obviously, performing a full reconstruction of the matrix at each time step would be extremely costly. Instead, in this section, I propose a change detection scheme which relies on a partial reconstruction as depicted in Figure 3.2: (a) a fair data matrix sampler, which identifies a small subset of the matrix cells as ground truth and (b) a partial reconstructor, which reconstructs a given subset of matrix cells, without reconstructing the full data matrix.

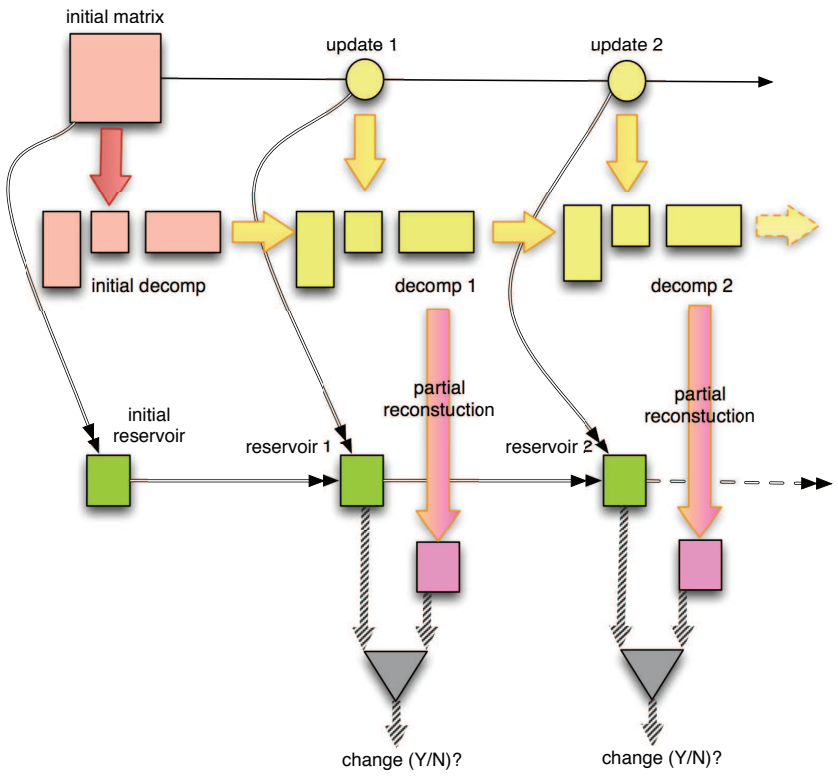


Figure 3.2: Overview of the change detection process

3.4.3 Fair Sampling of an Evolving Matrix

We propose a fair sampler, where all matrix cells have a uniform probability of being selected independently of when they are updated.

Basic Reservoir Sampling. Reservoir sampling [85] is a random sampling method that works well in characterizing data streams. It is especially efficient because (a) it needs to see the data only once and (b) it uses a fixed (and small) buffer, referred to as the "*reservoir*". Furthermore, while (c) it does not require a priori knowledge of the data size, it (d) ensures that each data element has an equal chance of being represented in the sample. Let $\mathcal{S}$ be a data stream consisting of a sequence of elements s_i. The reservoir sample keeps a fixed reservoir of, say w elements. Once the reservoir is full, each new element, s_i, replaces a (randomly) selected element in the

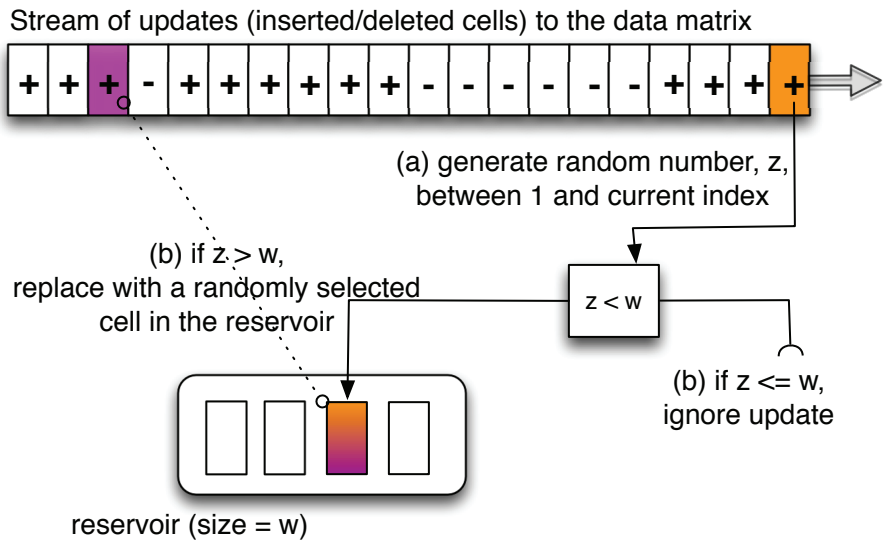
reservoir with a decreasing probability, inversely proportional to the index, i, of the new element s_i. More specifically, a random element in the reservoir is replaced by s_i with probability $\frac{w}{i}$. Intuitively, in a fair sampling, each element up to i should have a w/i chance of being in the random sample of size w. Therefore, s_i is selected to be included in the reservoir with probability $\frac{w}{i}$. The sample it replaces, on the other hand, is chosen randomly among the existing w samples in the reservoir to ensure that the reservoir forms a random sample of the first i elements in the stream.

Matrix-Reservoir Model. As described earlier, I consider the general case where the data matrix can grow or shrink with insertions or deletions of rows and columns. More specifically, I model the evolving data matrix as a stream, $\mathcal{S}$, of $s_i = \pm\langle row_i, col_i\rangle$, where row_i and col_i are the row and columns affected in the update with index i: $+\langle row_i, col_i\rangle$ indicates that the update inserts a new cell in the matrix at location $\langle row_i, col_i\rangle$, whereas $-\langle row_i, col_i\rangle$ indicates that the cell at location $\langle row_i, col_i\rangle$ is being removed.

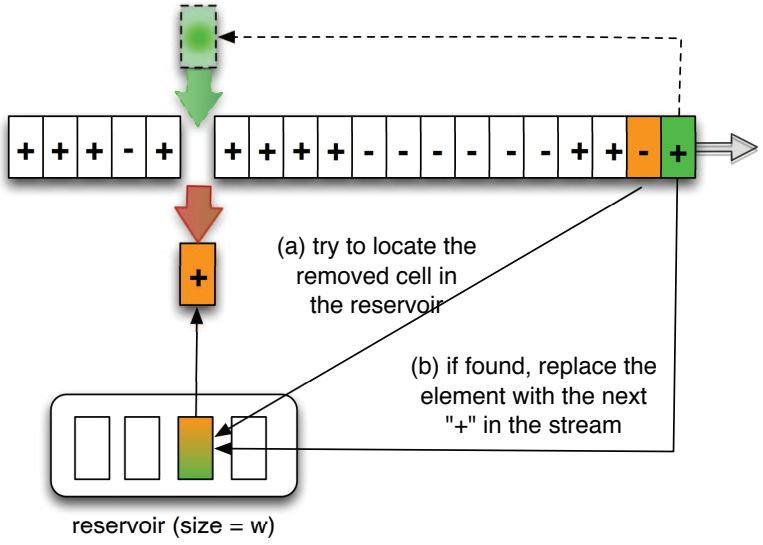
The reservoir, $\mathfrak{R}_i = \{\mathfrak{r}_{i,1}, \ldots, \mathfrak{r}_{i,w}\}$, at time i consists of w matrix cell positions, which serve as the representatives for the current matrix. In other words, each $\mathfrak{r}_{i,j} \in \mathfrak{R}_i$ is a triple of the form $\mathfrak{r}_{i,j} = \langle index_{i,j}, row_{i,j}, col_{i,j}\rangle$, where $index_{i,j}$ is the index of the update that deposited the cell, located at $row_{i,j}$ and $col_{i,j}$, into the reservoir.

Matrix-Reservoir Maintenance for $\mathbf{s_i} = +\langle \mathbf{row_i}, \mathbf{col_i}\rangle$. As discussed earlier, reservoir sampling randomly selects some of the incoming stream elements for the updating the contents of the reservoir. When the (probabilistically) selected incoming stream entry s_i is of the form $+\langle row_i, col_i\rangle$, the basic reservoir sampling process is applied: a random element, $\mathfrak{r}_{i-1,j}$ from the current reservoir $\mathfrak{R}_{i-1}$ is selected and this is replaced with $\langle i, row_i, col_i\rangle$. This process is visualized in Figure 3.3(a).

Matrix-Reservoir Maintenance for $\mathbf{s_i} = -\langle \mathbf{row_i}, \mathbf{col_i}\rangle$. When the (probabilistically) selected incoming entry s_i is of the form $-\langle row_i, col_i\rangle$, on the other hand, the



(a) reservoir maintenance for $\mathbf{s_i} = +\langle \mathbf{row_i}, \mathbf{col_i} \rangle$



(b) reservoir maintenance for $\mathbf{s_i} = -\langle \mathbf{row_i}, \mathbf{col_i} \rangle$

Figure 3.3: Overview of the reservoir based matrix sampling

basic reservoir sampling process cannot be applied as this denotes removal of a cell, not insertion. We handle deletions as follows:

- if there exists no

$$\mathfrak{r}_{i-1,j} = \langle index_{i-1,j}, row_{i-1,j}, col_{i-1,j} \rangle \in \mathfrak{R}_{i-1},$$

  such that $row_{i-1,j} = row_i$ and $col_{i-1,j} = col_i$, then s_i is simply ignored;

- if, on the other hand, there exists a

$$\mathfrak{r}_{i-1,j} = \langle index_{i-1,j}, row_{i-1,j}, col_{i-1,j} \rangle \in \mathfrak{R}_{i-1},$$

  such that $row_{i-1,j} = row_i$ and $col_{i-1,j} = col_i$, then

  - we drop $\mathfrak{r}_{i-1,j}$ from the reservoir and
  - we keep the j^{th} position reserved for a future update of the form $s_h = +\langle row_h, col_h \rangle$.

Intuitively, the matrix reservoir (and its history) is revised as if the future insertion s_h had in fact arrived *in the past*, instead of $s_{index_{i-1,j}}$, which had originally deposited the cell, $\langle row_{i-1,j}, col_{i-1,j} \rangle$ (which is being deleted) into the reservoir. This process is visualized in Figure 3.3(b).

3.4.4 Partial Matrix Reconstruction

At time $t = i$, let us have the reservoir $\mathfrak{R}_i = \{\mathfrak{r}_{i,1}, \ldots, \mathfrak{r}_{i,w}\}$, where for all $1 \leq h \leq w$, $\mathfrak{r}_{i,h} = \langle index_{i,h}, row_{i,h}, col_{i,h} \rangle$. Intuitively, the reservoir consists of a set of matrix cell positions (that were fairly sampled from the overall matrix). During the partial reconstruction step, we use the (incrementally maintained) SV decomposition, U_i, S_i,

and V_i, of the data matrix X_i to reconstruct only the row and column positions that appear in the reservoir, $\mathfrak{r}_{i,h}$.

More formally, the partially reconstructed matrix value set $\hat{\mathfrak{V}}_i = \{\hat{\mathfrak{v}}_{i,1}, \ldots, \hat{\mathfrak{v}}_{i,w}\}$, is such that for all $1 \leq h \leq w$,

$\hat{\mathfrak{v}}_{i,h} = \hat{X}_i[row_{i,h}, col_{i,h}]$, where

$$\hat{X}_i[row_{i,h}, col_{i,h}] = (U_i[row_{i,h}, *])\, S_i \left(V_i^T[*, col_{i,h}]\right).$$

Note that the cost of the partial reconstruction of the matrix depends on the size of the reservoir and when $|\mathfrak{R}_i| \ll |X_i|$, partial reconstruction is much faster than full reconstruction.

3.4.5 Change Detector

At time $t = i$, given the reservoir $\mathfrak{R}_i = \{\mathfrak{r}_{i,1}, \ldots, \mathfrak{r}_{i,w}\}$, we construct a ground truth value set $\mathfrak{V}_i = \{\mathfrak{v}_{i,1}, \ldots, \mathfrak{v}_{i,w}\}$, where for all $1 \leq h \leq w$, $\mathfrak{v}_{i,h} = X_i[row_{i,h}, col_{i,h}]$. Similarly, we also have the partially reconstructed value set $\hat{\mathfrak{V}}_i = \{\hat{\mathfrak{v}}_{i,1}, \ldots, \hat{\mathfrak{v}}_{i,w}\}$, where for all $1 \leq h \leq w$, $\hat{\mathfrak{v}}_{i,h} = \hat{X}_i[row_{i,h}, col_{i,h}]$, where $\hat{X}_i[row_{i,h}, col_{i,h}]$ is the partially reconstructed value for the cell location $\langle row_{i,h}, col_{i,h}\rangle$. Given these, we can detect a major structural change in the data matrix if

$$\sum_{h=1}^{w} (\mathfrak{v}_{i,h} - \hat{\mathfrak{v}}_{i,h})^2 \geq \Theta,$$

where Θ is the inaccuracy threshold.

3.4.6 Pseudocode of the LWI2-SVD Algorithm

We provide the pseudocode of the LWI-SVD with Restart (LWI2-SVD), which was detailed in this section, in Algorithm 2. In the next section, I evaluate the efficiency and effectiveness gains of LWI2-SVD algorithm on top of the gains provided by LWI-SVD.

Table 3.1: Parameters

Symbol	Description	Default	Alternative
$dim(n \times n)$	Initial(for insertions)/Final(for deletions) dimensions of X	100×100	300×300
r	Target rank	5	10
len	Length of the data stream	50	50
num_{upd}	Numbers of columns:rows updated at a given iteration	2:2	6:6
λ_{upd}	Strength of the updates (for synth. data)	5	10
w	Reservoir size	50	150
Θ	On-demand restart threshold	20%	10%
per	Restart period	15	5

3.5 Experiments

In this section, we evaluate the efficiency and effectiveness of LWI-SVD and LWI2-SVD on both synthetic and real datasets and for different scenarios and parameter settings.

Each experiment, consisting of *len* consecutive update iterations, was run 10 times and averages are reported. Note that to simplify the interpretation of the results we have considered *insertion sequences* and *deletion sequences*; but not hybrid *insertion/deletion* sequences. Also, to make sure that the results for experiments involving sequences of insertions and deletions are comparable, we have set the initial dimensions for an insertion sequence and the final dimensions of a deletion sequence to the same value, *dim*.

The various parameters varied in the experiments, default values, and value ranges are presented in Table 3.1. Below we describe the experimental setting, including the data sets, in greater detail.

3.5.1 Real Data: Digg.com Traces

We use Digg.com data set from Infochimps to evaluate the effectiveness and efficiency for real data. The complete data set was recorded from August to November

2008 and has 3 main components: stories, comments and replies. "Stories" contain 1490 articles that users have posted within the time period. For our experiments, we created data streams by considering the first $n + len \times num_{upd}$ articles in the data set(the first n articles make up the initial data matrix; for each of the len iterations in the update stream, we considered num_{upd} new articles).

Given this data set, we removed the stop words and applied stemming. We then selected the first n stories and identified the most frequent n keywords [1] . X_{ij} denotes occurrence of keyword j in story i. Intuitively, the low-rank decomposition of the data matrix X simultaneously cluster stories and keywords, resulting a co-clustering of the data matrix X. We moved the window at each iteration by inserting or deleting num_{upd} records of the story trace and recomputing the n most frequent keywords (meaning that num_{upd} many rows and columns are inserted and deleted). These correspond to row and column insertions/deletions on X.

3.5.2 Synthetic Data: Random Traces

We have also experimented with synthetic data sets where we could freely vary the characteristics of the data and updates to observe the accuracy and efficiency of our algorithms under different scenarios. For these experiments, we have created synthetic activity traces which we then converted into data matrices as before. Since the matrices for real data is sparse, we focus on dense matrices.

In particular, we have generated an initial n-length random sequence of 5 dimensional data, where each dimension has a value from 0 to 10. Given these n consecutive records in the trace, we have created a $n \times n$ initial matrix measuring pairwise Euclidean distances of the records in the sequence. Insertions in the random trace were generated by randomly picking numbers with exponential distribution, with the rate

[1] In these experiments, without loss of generality, we kept the matrix in square shape, i.e., $n = m$

parameter, λ_{upd} (i.e., $prob(x) = exp\_dist(x, \lambda_{upd}) = \lambda_{upd} e^{-\lambda_{upd} x}$). Intuitively, if the rate parameter λ_{upd} is large, there is a higher likelihood of having more large amplitude changes. If the rate parameter λ_{upd} is low, there is a lower frequency of large amplitude changes in the trace.

As before, we enlarged or shrank X at each iteration by adding or deleting num_{upd} units of the random activity trace (meaning that num_{upd} many rows and columns are inserted to or deleted from into the matrix, X).

3.5.3 Evaluation Criteria and Competitors

We evaluate the LWI-SVD and LWI-SVD with Restart (LWI2-SVD) algorithms by comparing them to alternative approaches:

- *Full SVD and SVDS* – SVD is the full SV decomposition of the matrix, we used Matlab's `[U, S, V] = svd(X)` command for this. We also considered with Matlab's `[U, S, V] = svds(X,r)` command which returns the composition results for the top-r components, where r is the desired rank (SVDS tends to perform more efficiently than SVD when r is small and X is large and sparse);

- *Naive Incremental SVD* – this is our implementation of the Brand's algorithm described in [15], it involves a full SVD and pivoted QR based approximation is not leveraged (to implement LWI-SVD and LWI2-SVD, we use this implementation as the basis); and

- *SPIRIT* – this is the algorithm described in [68] which provides fast decompositions, but does not have various desirable properties of incremental SVD; including explicit data deletions and column insertions and deletions.

LWI-SVD family of the algorithms extend our implementation of the Brand's algorithm described in [15] along with the Algorithm 844 [8].

As evaluation criteria, we use three metrics: reconstruction error overhead, execution time, and execution time gain:

- *average relative reconstruction error* (err_{rel}) – this accuracy measure is defined as

$$\frac{1}{len}\sum_{i=1}^{len}\frac{rec\_error(\hat{X}_{i,*},X_i)-rec\_error(\hat{X}_{i,SVD},X_i)}{rec\_error(\hat{X}_{i,SVD},X_i)},$$

  where

  - len is the number of iterations (length of the stream),
  - $\hat{X}_{i,*}$ denotes the decomposition of the data matrix at time i obtained using the algorithm "$*$", and
  - $rec\_error(Y,X)$ denotes the reconstruction error of the decomposition Y against the data matrix X, measured in terms of $Frobenius$ norm.

  Note that a low-rank decomposition of X_i would lead to a reconstruction error, even if it is obtained using full SVD followed by selection of the top r components. Therefore, the denominator of the above term is not equal to 0.

- *absolute execution time* (t_{exec}) – this is the time, in seconds, that is required to complete len consecutive decompositions using the algorithm under consideration.

- *time gain* ($gain_{time}$) – the gain in time is the execution time measured against the execution time of the full SVD; i.e., $\frac{t_{exec,svd}-t_{exec,*}}{t_{exec,svd}}$.

All experiments were conducted using a 4-core Intel Core i5-2400, 3.10GHz, machine with 8GB memory, running 64-bit Windows 7 Enterprise. The codes were executed using Matlab 7.11.0(2010b).

3.5.4 Evaluation with the Default Settings

Real Trace Data Set

Figure 3.4 presents the accuracy and efficiency results for the real trace data for the default parameters reported in Table 3.1.

Accuracy. The first thing to note in Figure 3.4(a), which reports average relative reconstruction errors for the various versions of the LWI-SVD algorithm proposed in this paper, is that restarts discussed in Section 3.4 are highly effective in reducing the overall error. While both partial reconstruction-based and periodic restarts used in LWI2-SVD are effective in improving accuracy over the LWI-SVD (which does not use restarts), the best results are obtained when these are used together, bringing down the average relative reconstruction error to 0.3-0.7% of the low-rank decomposition obtained through full SVD.

The second thing to note in Figure 3.4(a) is that row/column insertions, which bring in new data into the matrix, results in larger relative reconstruction errors than row/column deletions. Note that, when both reservoir-based and periodic restarts are employed, the accuracy penalty relative to the low-rank decomposition of full SVD is negligibly low for both insertions and deletions.

Efficiency. Figure 3.4(b) shows the efficiency results for this data set under the default parameter configuration.

The first thing to note is that there is minimal time difference between the LWI-SVD and LWI2-SVD algorithm. This indicates that the time overhead of reservoir maintenance and occasional on-demand full decompositions are negligible in the long run. Secondly, performing full SVD takes $\sim$ 75-100% more than the proposed LWI-SVD family of algorithms. Under this configuration, the naive incremental SVD takes a little more time than full SVD, as the basic algorithm reported in [15] involves a full

Table 3.2: Impact of setting $A = I$ in Section 3.3.1

	$A = I$	A is free	Impact
Rec. error	5.786	5.765	+0.36%
Exec time	0.174 sec	0.197 sec	−11.71%

SVD with same dimension as the original matrix and several matrix multiplications. Further-more, under this configuration, SVDS takes even longer than the full SVD.

Finally, a close look at the LWI-SVD family of algorithms indicates that insertions require slightly longer time to maintain than deletions. This is expected because, as discussed in Section 3.2.1, there is no need for computing R_A and R_B since they are all zero.

Impact of the QR-Elimination Optimization. In Section 3.3.1, we had discussed an optimization strategy whereby we eliminate one of the two expensive QR operations by forcing A to be equal to the identity matrix, I. As shown in Table 3.2, setting $A = I$ causes less than half percentage point impact on the accuracy; on the other hand, this optimization helps save close to 12% in execution time.

Synthetic Trace Data Set

Figure 3.6 presents results for the synthetic trace data set under the default parameter settings. The key observation from this figure is that the accuracy and efficiency results for the synthetic trace data set are very similar to the results for real trace data set, reported in Figure 3.4. The similarity is especially pronounced in the execution time results in Figure 3.6(b). This indicates that the execution time gains of the LWI-SVD family of algorithms (and to a certain degree, the accuracies they provide – especially with the help of periodic and reservoir-based restarts) are inherent

properties of these algorithms rather than being highly data specific.

SPIRIT. Since the SPIRIT [68] algorithm approaches the problem differently (e.g. cannot directly handle deletions, cannot handle deletions/insertion of columns), we present it separately from the rest in Figure 3.5. For these experiments, we use a synthetic data trace that does not include any column insertions or deletions on the data matrix X. As the figure shows, SPIRIT algorithm works much faster than SVD or LWI2-SVD for the default configuration. However, this speed comes with a significant increase in the reconstruction error, relative to the optimal low-rank decomposition using SVD. In contrast, LWI2-SVD achieves an accuracy almost identical to the optimal, yet costs only half as much.

3.5.5 Impacts of Data and System Parameters

In this subsection, we evaluate the impacts of the various data and systems parameters on the efficiency and effectiveness of the LWI-SVD family of algorithms. As representative, we select the LWI2-SVD with the default parameters. We then vary, one-by-one, the various data and system parameters, and compare the results against the optimal SVD based rank-r decomposition. Since, as we have seen, the results are similar for real and synthetic data, for the most part we report the results with the real trace data. We use the synthetic trace only for experiments where we vary the strengths of the updates.

Varying the Target Rank, r

Figure 3.7 presents efficiency and accuracy results for the real trace data set where the target rank, r is varied. The results show that, as expected (due to the low-rank nature of the LWI-SVD family of algorithms), as the target rank increases, the time gain drops and the relative error rate slightly increases. The drop in time gains

is because the incremental process involves a lot of matrix multiplications where the sizes of matrices are directly related to the target rank. This confirms the observation that LWI-SVD and LWI2-SVD are most effective when the target rank is low.

Varying the Dimensions, dim, of the Matrix

Figure 3.8 presents accuracy and efficiency results when we change the dimensions, dim, of the initial data matrix (for insertions) and the final data matrix (for deletions). Here, we see that increasing the size of matrix does not have a big impact on accuracy and efficiency.

Varying the Rate of Updates, num_{upd}

Figure 3.9 presents efficiency and accuracy results for the real trace data set where the number, num_{upd}, of row and column updates per each iteration is varied. The results indicate that, as expected, an increase in the number of updates per iteration impacts accuracy as well as efficiency. The slight impact on the accuracy is due to the approximation nature of the algorithm. The impact on the time gain is due to more on-demand restarts.

Varying the Reservoir Size, w

Figure 3.10 presents efficiency and accuracy results for the real trace data set where the reservoir size, w, is varied. The results confirm that a larger reservoir (even only $\sim 1.5\%$ of the matrix) can help to trigger on-demand restarts more fairly, since larger reservoir has more accurate amortized error measuring.

Varying the Change Threshold, Θ

Figure 3.11 confirms that a slightly tighter threshold, $\Theta = 0.1$ instead of the default $\Theta = 0.2$ will trigger more on-demand restarts and thus can further reduce the error rates (which are already very low), with little impact on execution time gains.

Varying the Restart Period, *per*

Figure 3.12 confirms that increasing the number of restarts by reducing the restart period, *per*, may improve the final accuracy. However, unlike the on-demand restarts based on change detection (shown in Figure 3.11), blindly increasing the frequency of the periodic restarts may negatively impact the time gain.

Varying the Update Strength, λ_{upd}

Finally, in Figure 3.13, we see the impact of the strength (in amplitude) of the incoming insertions. The figure shows that, when λ_{upd} increases, the LWI2-SVD algorithm adjusts its operation by scheduling more on-demand restarts at a cost of decreasing the time gain.

3.5.6 Scalability of LWI Algorithms

The results shown above are conducted with small window size, however, in some cases, we need large windows to monitor and analyze a large portion of the data. In this subsection, we analyze the scalability of LWI Algorithm by choosing large base number. Since we have shown that under the small base number condition, SVD out performs SVDS in execution time, however, when the base number is large, seeking a low rank deposition using SVDS is more efficient. Also, as we know that SVDS is very efficient when the data is sparse, but performs less efficient on dense data. We showed that LWI algorithm can concur this short coming when the data is dense.

Table 3.3: Results for Large Dim

dim	LWI2 Exec. Time(s)	LWI2 Rel. Error	SVDS Exec. Time(s)
$1000 * 100$	8.2604	0.143%	15.91
$1000 * 1000$	6.8087	0.06%	10.839
$1500 * 1500$	17.483	0.03%	23.469
$2000 * 2000$	34.35	0.002%	41.491
$3000 * 3000$	96.577	0.00097%	93.622

Recall in section 3.3.3, we showed that K is an arrow-like matrix which is very sparse, this leads to the efficiency by using pivoted QR compared to a direct SVDS on the dense data. Therefore, in the incremental maintenance of SVD on a dense matrix, we are actually seeking a second layer reduced rank approximation of a sparse matrix K. It is the main advantage of LWI algorithm compared to SVDS when the data is dense and the base dimension is large. Table 3.3 shows the execution time and error overhead results under a synthetic dense data, the results confirm that with big base number especially when the base is a thin and tall matrix, LWI algorithm can have advantages in execution time with negligible error overhead .

3.6 Conclusion

In this work, I presented a *Low-rank, Windowed, Incremental SVD* (LWI-SVD) algorithm, which relies on low-rank approximations to speed up incremental SVD updates. LWI-SVD algorithm also aggregates multiple row/column insertions and deletions to further reduce on-line processing cost. We also presented a *LWI-SVD with restarts* (LWI2-SVD) algorithm which performs periodic and change detection based on-demand refreshing of the decomposition to prevent accumulation of errors. Experiment results on real and synthetic data sets have shown that the LWI-SVD family of incremental SVD algorithms are highly efficient and accurate compared to

alternative schemes under different settings.

As discussed in Section 2.4, NMF is another way of decomposing a data matrix into several factor matrices, and it creates non-negative factors which in some applications, the probabilistic interpretation of the results leads to better understand of the data and decision making. Therefore, I also explore the scenario where we have to apply NMF to a streaming data environment. In the next Chapter, I present the solutions to tackle the challenges when applying NMF to data streams that contains redundancy.

Algorithm 2 LWI2-SVD

Input:

The Base Matrix, X, and its SV decomposition $U_x S_x V_x^T$;

The update matrix, $\Delta = AB^T$, corresponding to a window of updates;

Target rank, r;

Reservoir, $\mathfrak{R}$;

Restart Threshold, Θ;

Periodic Restart Flag, f;

Output:

The new SVD results, U'_x,S'_x, and V'_x;

The new Reservoir, $\mathfrak{R}'$;

$X' = X + \Delta$

if f = true **then**

  $\langle U'_x, S'_x, V'_x \rangle = \text{topK\_SVD}(X', r)$;

  $\mathfrak{R}' = \text{updateReservoir}(\mathfrak{R}, \Delta)$;

else

  $\langle U'_x, S'_x, V'_x \rangle = \text{LWI-SVD}(X, U_x, S_x, V_x, \Delta, r)$;

  $\mathfrak{R}' = \text{updateReservoir}(\mathfrak{R}, \Delta)$;

  $\hat{\mathfrak{V}} = \text{partialReconstruct}(\mathfrak{R}', U'_x, S'_x V'_x)$;

  $E = \text{measurePartialError}(\hat{\mathfrak{V}}, \mathfrak{R}', X')$;

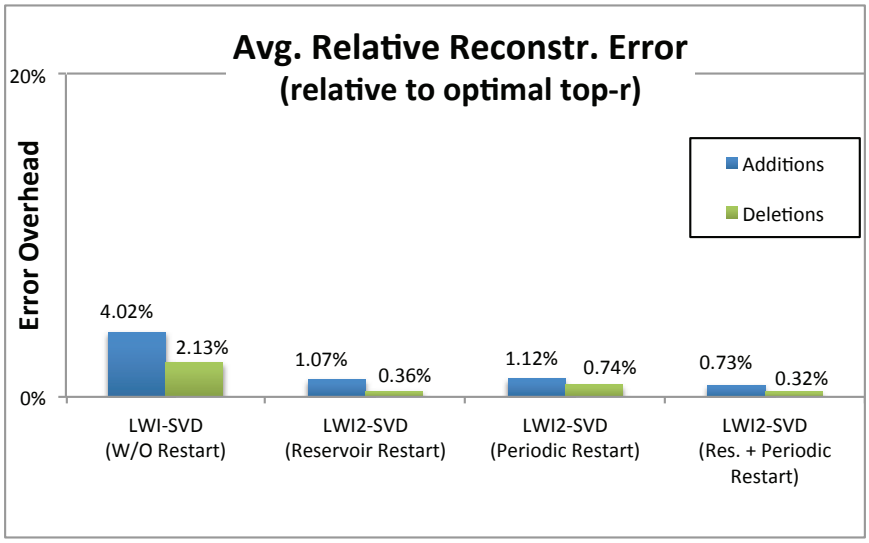
  if $E > \Theta$ **then**

    $\langle U'_x, S'_x, V'_x \rangle = \text{topK\_SVD}(X', r)$;

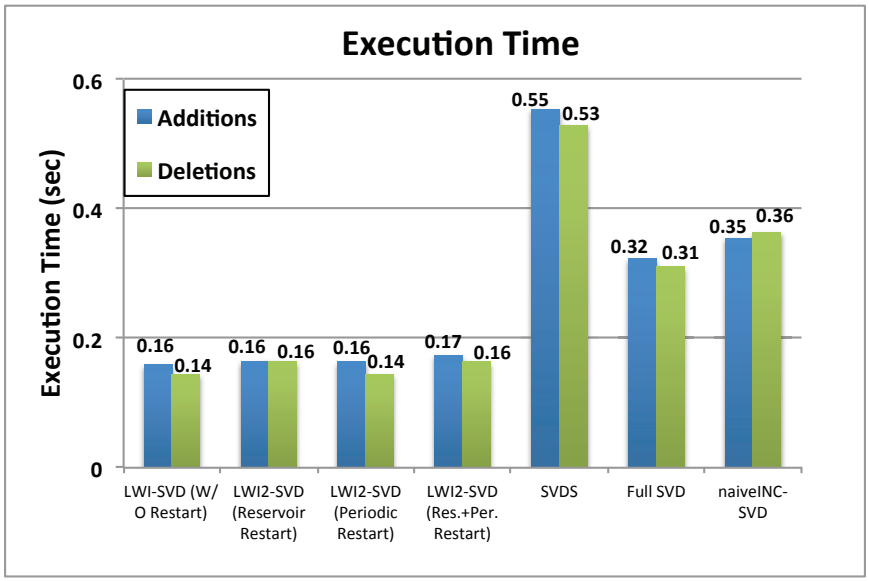
  end if

end if

return $U'_x, S'_x, V'_x, \mathfrak{R}'$;



(a) average relative reconstruction error



(b) execution time

Figure 3.4: Accuracy and efficiency for the *real trace data set - default settings*

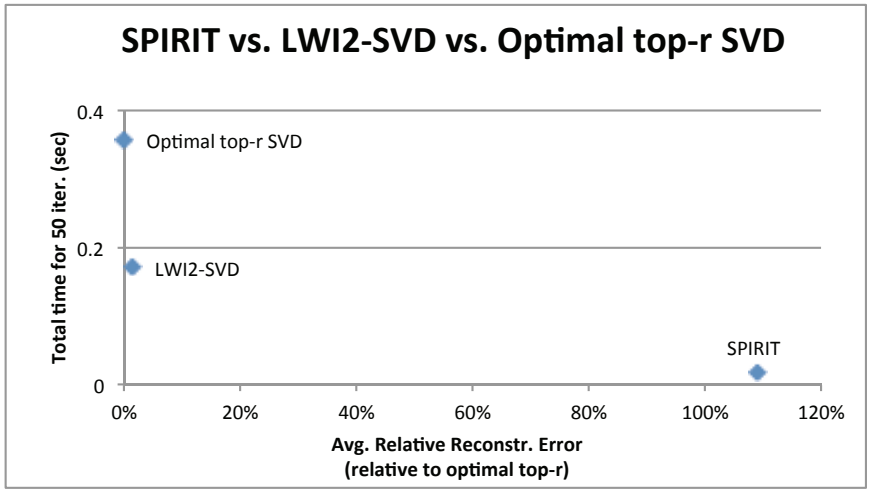
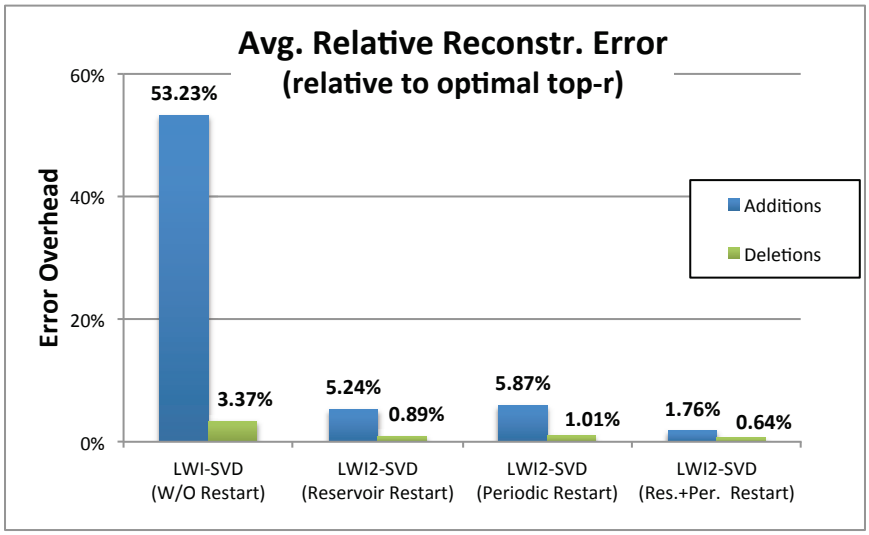
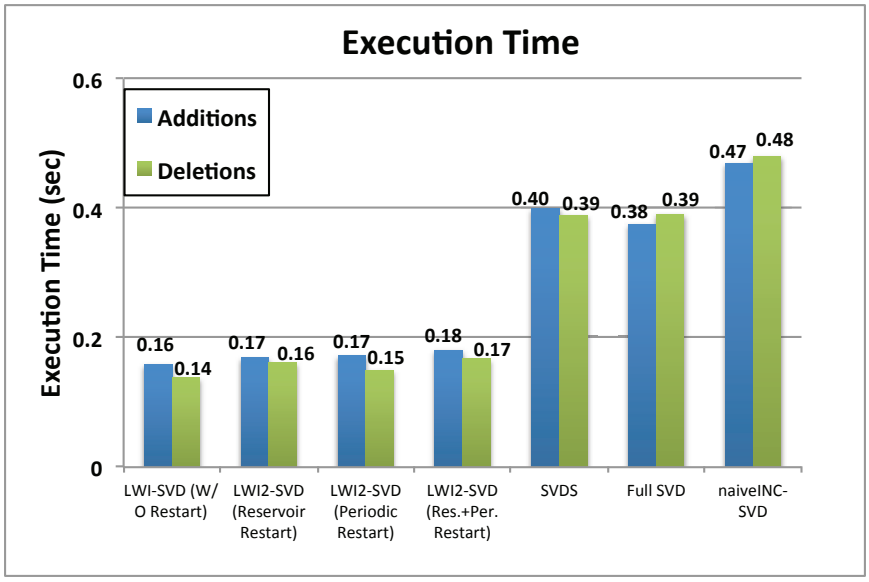


Figure 3.5: Accuracy and efficiency results for the *synthetic trace data set* for SVD, Spirit, and LWI2-SVD with periodic and on-demand refreshes



(a) average relative reconstruction error



(b) execution time

Figure 3.6: Accuracy and efficiency for *synthetic trace data set - default settings*

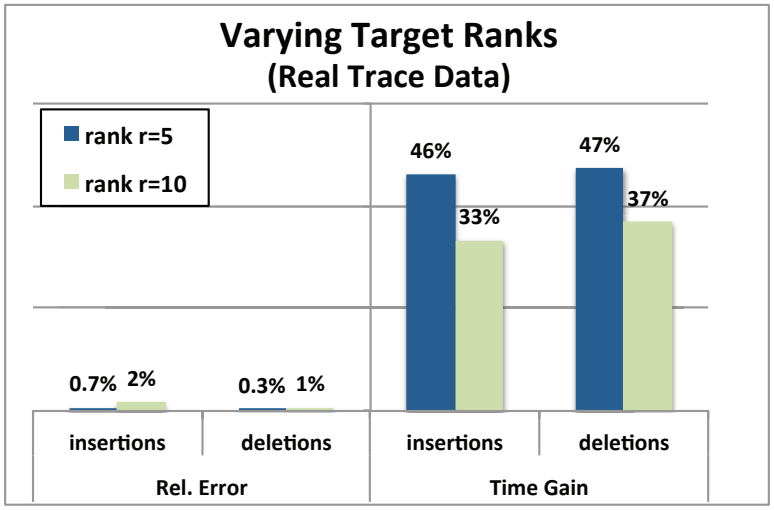


Figure 3.7: Accuracy and efficiency for *real trace data set* for different *rank, r*

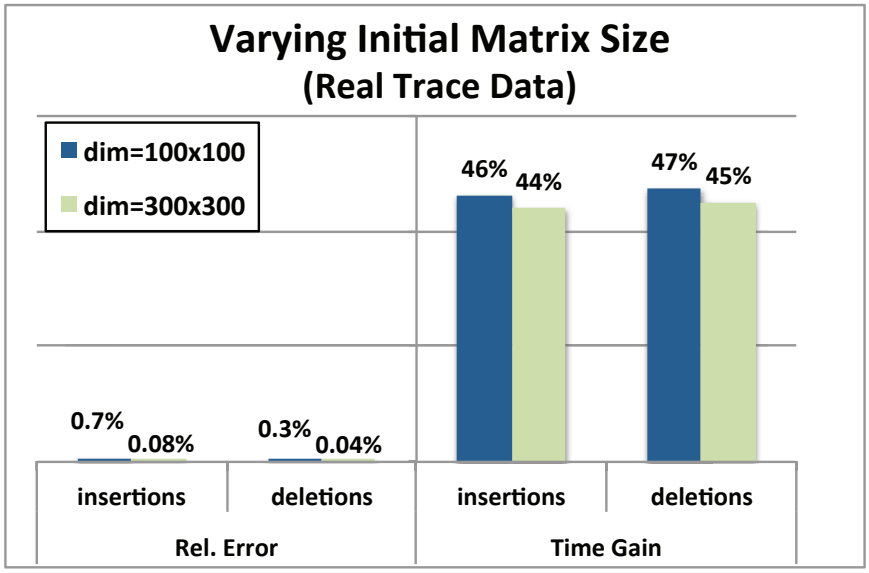


Figure 3.8: Accuracy and efficiency results for the *real trace data set* varying the *size, dim, of the initial (for insertions) / final (for deletions) matrix*

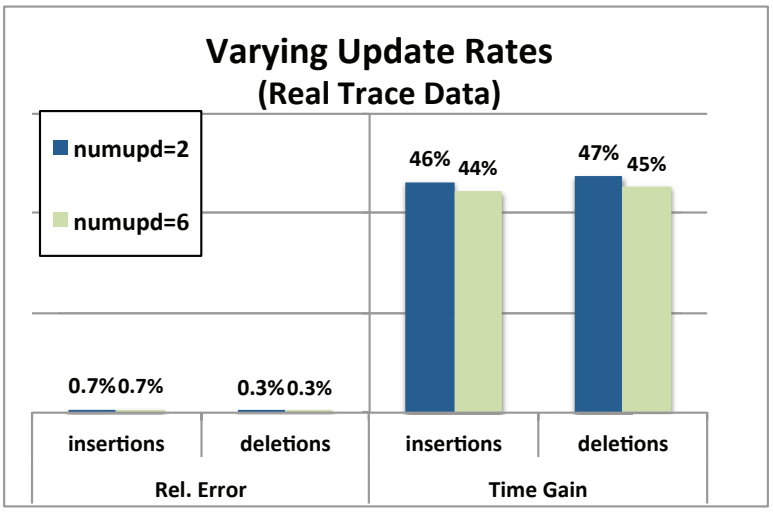


Figure 3.9: Accuracy and efficiency results for the *real trace data set* varying the *amount updates per iteration*, num_{upd}

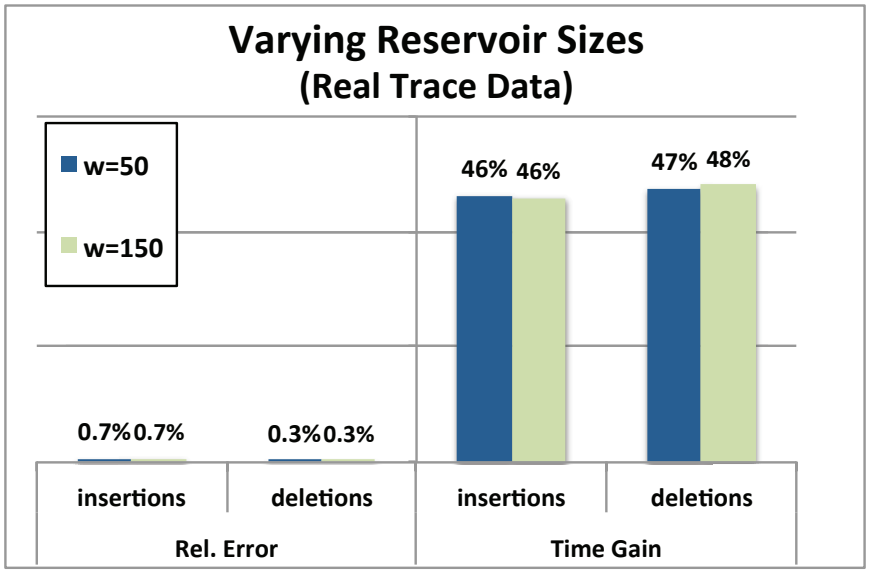


Figure 3.10: Accuracy and efficiency for *real trace data* varying *reservoir size*, w

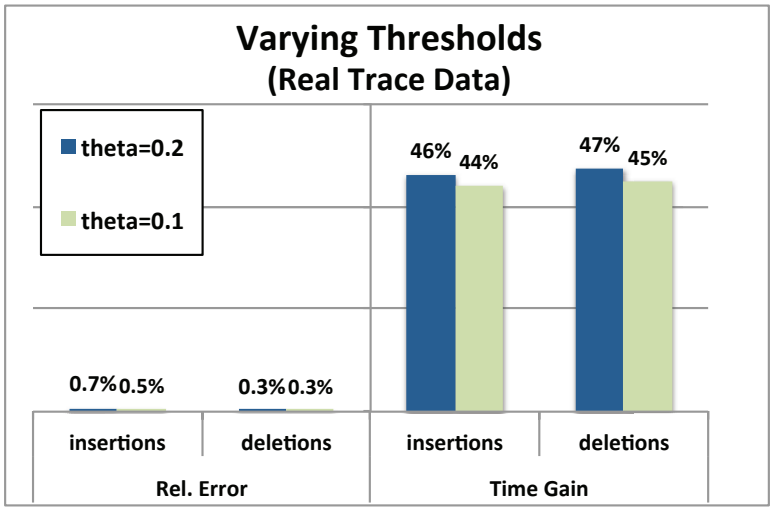


Figure 3.11: Accuracy and efficiency results for the *real trace data set* varying the *change threshold,* Θ, for on-demand restarts

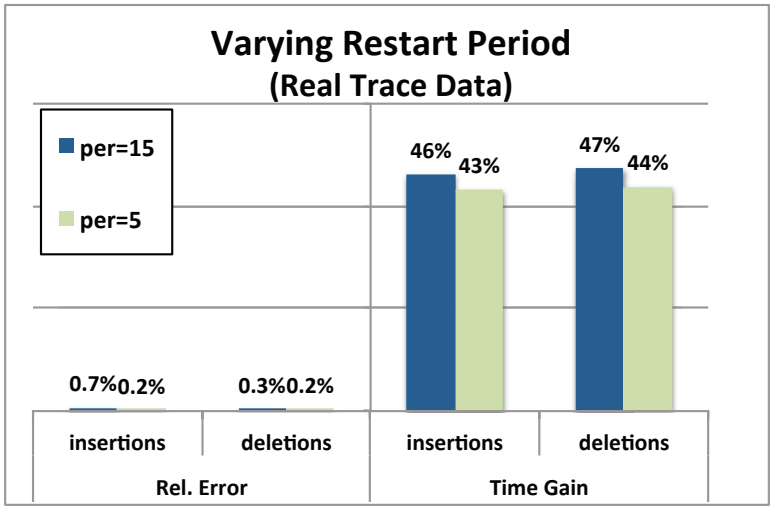


Figure 3.12: Accuracy and efficiency results for the *real trace data set* varying the *restart period, per*, for periodic restarts

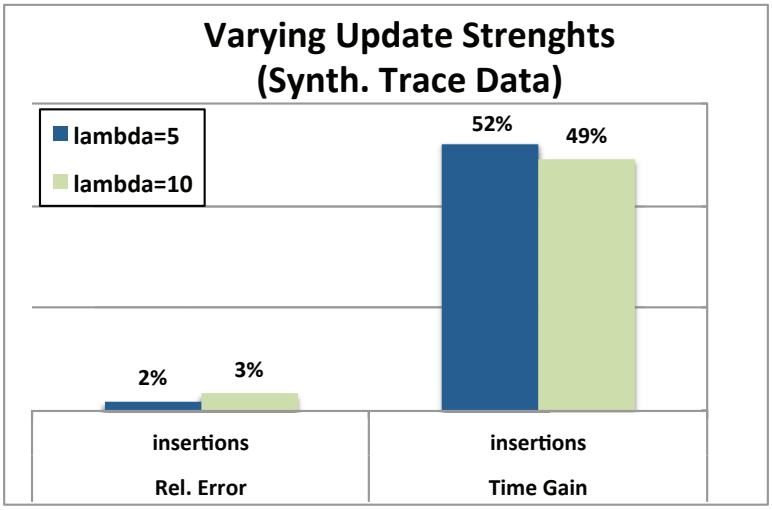


Figure 3.13: Accuracy and efficiency results for the *synthetic trace data set* varying the *update strength*, λ_{upd}

Chapter 4

GROUP INCREMENTAL NON-NEGATIVE MATRIX FACTORIZATION ON DATA STREAMS

4.1 Introduction

Non-negative matrix factorization (NMF) is a well known method for obtaining low rank approximations of data sets, which can then be used for efficient indexing, classification, and retrieval. The non-negativity constraints enable probabilistic interpretation of the results and discovery of generative models. One key disadvantage of the NMF, however, is that it is costly to obtain and this makes it difficult to apply NMF in applications where data is dynamic. Recognizing that many applications involve redundancies and these redundancies can and should be leveraged for reducing the computational cost of the NMF process: Firstly, online applications involving data streams often include temporal redundancies. Secondly, and perhaps less obviously, many applications include integration of multiple data streams (with potential overlaps) and/or involves tracking of multiple similar (but different) queries; this leads to significant data and query redundancies, which if leveraged properly can help alleviate computational cost of NMF. Based on these observations, I will introduce *Group Incremental Non-Negative Matrix Factorization* (GI-NMF) which leverages redundancies across multiple NMF tasks over data streams. The proposed algorithm relies on a novel *group multiplicative update rules* (G-MUR) method to significantly reduce the cost of NMF. G-MUR is further complemented to support incremental update of the factors where data evolves continuously. Experiments show that GI-NMF significantly reduces the processing time, with minimal error overhead.

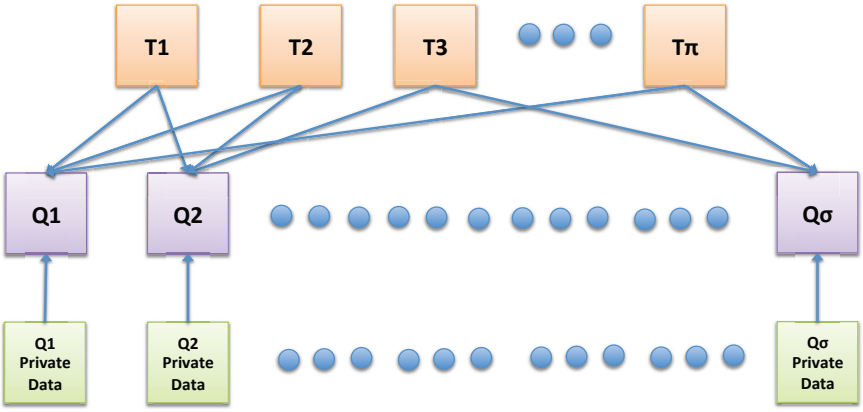


Figure 4.1: Overview of the continuous NMF query model (with π sharable data sources and σ NMF queries – each query may also have its private (unshared) data source).

4.2 Problem Formulation

I focus on applications where multiple NMF queries (e.g. triggers) are continuously executed on time-evolving data streams, such as those from sensors (for example providing continuous readings in different zones of a smart building to support various optimization and prediction tasks) or from social media (for example to support personalized recommendations or various collaborative filtering tasks) [44, 79, 73, 35, 33, 58]. Figure 4.1 shows an overview of the problem setting, where we have

- a set $\mathcal{D}$ of sharable data sources $\mathcal{D} = \{D_1, D_2, ..., D_\pi\}$, where each D_k has an *update rate*, $u(D_k)$, with which D_k receives new data,
- a set $\mathcal{Q}$ of continuous NMF queries $\mathcal{Q} = \{q_1, q_2, ..., q_\sigma\}$. In addition to using sharable data sources, an NMF query may also have its own private (unshared) data source. We denote the *update rate* of the private source for query q_j as $u(q_j)$.

We describe which data source is used for which query (or conversely, which query access which data sources) with a $\pi \times \sigma$ data source-query relationship matrix, $\mathbb{M}$.

We denote

- the set of data sources used for the query q_j as $\mathcal{D}(j)$, and
- the set of NMF queries that access content from the sharable data source D_k as $\mathcal{Q}(k)$.

At time stamp $time_i$, for each query, $q_j \in \mathcal{Q}$, we have a feature-object data matrix, $D_{i,j}$, extracted from the data sources describing the objects that query q_j received from all relevant sharable data sources, plus any private data source.

Group NMF (G-NMF) For a given target rank, r, the *group NMF* (G-NMF) problem seeks to identify the set $\mathcal{L}_i = \{L_{i,1}, \ldots, L_{i,\sigma}\}$, of low rank ($r$) NMF approximations of each $D_{i,j}$. More specifically, each $L_{i,j}$ is a pair $\langle W_{i,j}, H_{i,j} \rangle$ where $W_{i,j}$ and $H_{i,j}$ are the rank-r NMF of the matrix $D_{i,j}$. $\diamond$

Sharable data sources make new data available to these continuous NMF queries at their own updating rates. Each NMF query's private data source also streams in new data. For NMF query, q_j, at time stamp, $time_{i+1}$, the feature-object matrix $D_{i+1,j} = \left(D_{i,j} \cup \left(\bigcup \Delta D_{i+1,j}^{Shared}\right)\right) \cup \Delta D_{i+1,j}^{Unshared}$, where $\bigcup \Delta D_{i+1,j}^{Shared}$ are the new data from sharable data sources, $\mathcal{D}(j)$, which cover query q_j and $\Delta D_{i+1,j}^{Unshared}$ are the new data from q_j's private data source.

Group Incremental NMF (GI-NMF) For a given target rank, r, and the set, $\mathcal{L}_i$, of rank-r NMFs of the matrices $D_{i,j}$ at time $time_i$, the *group incremental NMF* (GI-NMF) problem seeks to identify the set $\mathcal{L}_{i+1} = \{L_{i+1,1}, \ldots, L_{i+1,\sigma}\}$, of low rank ($r$) NMF approximations of each $D_{i+1,j}$ at time $time_{i+1}$. $\diamond$

As an example, consider a media service like Twitter where users subscribe to popular data provider accounts and these sources are shared by many users. In this

context, GI-NMF queries would help the service provider to efficiently keep track of the index terms and content clusters and provide accurate and timely recommendations. As a second example, consider a building energy management system (BEMS), which divides a large building into zones, such that (a) each zone has sensors collecting locally relevant data, but (b) building zones are also organized such that there are sensory information shared by multiple zones (such as outside air temperature, number of individuals in a floor or office space, the amount of cooling air pushed by a central AC unit at a given point in time). GI-NMF queries could help the BEMS analyze the sensory data from different sensors for real-time patterns (such as faults and optimization opportunities).

4.3 Group Non-negative Matrix Factorization (G-NMF)

We now describe our *Group non-negative matrix factorization* (G-NMF) algorithm, which takes advantage of the data source-query matrix, $\mathbb{M}$ in efficiently identifying the r latent semantics of a group of queries.

Let us consider the time instance $time_i$, where the feature-object [1] data matrix, $D_{i,j}$, of the query q_j is an $n \times m_j$ matrix, where n is the number of features and m_j is the number of objects q_j has received so far. Note that $D_{i,j}$ is a combination of the data provided by the shared data sources (i.e., $\mathcal{D}(j)$ and the data from the unshared data source associated with query q_j. In other words, $D_{i,j} = \left(\sum_{D_k \in \mathcal{D}(j)} \oplus \mathbb{D}_{i,k}\right) \oplus \mathbb{Q}_{i,j}$, where $\mathbb{D}_{i,k}$ is a feature-object matrix corresponding to shared data source D_k, $\mathbb{Q}_{i,j}$ is the feature-object matrix corresponding to the unshared data source of query q_j, and $\oplus$ indicates horizontal concatenation of the data matrices (i.e., columns of object vectors from different shared data sources are packed next to each other horizontally).

[1] While it is customary to represent the data matrices in the form of object-feature matrices, for convenience of the later discussion, assume we are given a feature-object representation, where rows represent features and columns represent objects.

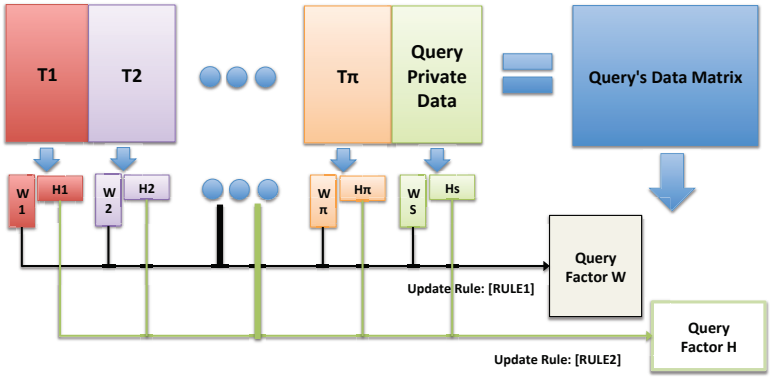


Figure 4.2: Overview of the Group NMF process

As formalized above, we are interested in the r latent semantics (i.e., rank-r low rank NMF approximations) of each query results.

As described in the introduction, the scenario includes a significant amount of redundancies (especially across queries that access great amount of data from the shared data sources). In particular, for any pair, q_j and q_h, of queries that access at least one shared data source, there will be certain degree of shared object columns as they will be receiving objects from the same shared data source. Therefore, performing NMF individually on all queries would be wasteful. In Figure 4.1, I see an example where query 1 accesses shared data sources $\{1, 2, \pi\}$ and query 2 accesses shared data sources $\{1, 2, 3\}$. In addition, each query has its own unshared data sources. The naive way of analyzing the latent semantics of these two queries would be performing NMF on each query's data matrix $D_{i,1} = (\sum_{k\in\{1,2,\pi\}} \oplus \mathbb{D}_k) \oplus \mathbb{Q}_1$ and $D_{i,2} = (\sum_{k\in\{1,2,3\}} \oplus \mathbb{D}_k) \oplus \mathbb{Q}_2$ separately. However, a close look at these two data matrices indicates that these two queries share two data source (D_1 and D_2), which means the columns in the feature-object matrices corresponding to the objects from these two shared data sources will be exactly the same in the two queries resulting data matrices. Figure 4.2 shows that an alternative approach to decomposing each query's data matrix, would be to decompose each data source's data matrix first to

obtain the subfactors for each data source and, then, combine these corresponding subfactors based on the matrix $\mathbb{M}$ which describes the relationships between the data sources and queries (along with the query's unshared data source) to compute the NMF factors.

4.3.1 Group-MUR (G-MUR) For Queries

The key challenge, then, is to efficiently combine sub-factors of the shared data source (and the unshared data source) into the factors of the queries. In the proposed G-NMF algorithm, this is achieved by adopting and extending the MUR-based tensor block decomposition technique (originally proposed for the parallelization of tensor decomposition operations [74]) for efficiently combining sub-factors arriving from different sources. We refer to this as the group MUR (or G-MUR) process. Let us consider a query [2] , q, with an $n \times m$ non-negative matrix D. Let us further assume that the matrix $D \sim W \times H$ can be (approximately) factorized into two factor matrices, $W \in \mathbf{R}^{n \times r}$ and $H \in \mathbf{R}^{r \times m}$, as specified in Equation 2.1.

Let us first partition D into $p \leq m$ partitions (according to the matrix $\mathbb{M}$ which describes the relationship between the shared data sources and queries) along its columns ($D = [D^{(1)}, D^{(2)}, ...D^{(p)}]$), such that for each $k \leq p$, $D^{(k)}$ is a data matrix from a shared data source in the database or a unshared data source. Note that, we can approximate each submatrix, $D^{(k)}$ via its own factors:

$$D^{(k)} \sim W^{(k)} \times H^{(k)}. \tag{4.1}$$

Let us next split the factor matrix H into p parts, $H = [H_{(1)}, H_{(2)}, ..., H_{(p)}]$, where each $k \leq p$ corresponds to a shared data source or a unshared data source. Given these, it is also true that each sub-matrix. $D^{(k)}$, can also be approximated, via the

[2] We drop the subscript j when we are focusing on only one query.

common factor W, as $D^{(k)} \sim W \times H_{(k)}$, where, $H_{(k)}$ indicates the part *in the overall factor* H of (data matrix D) corresponding to partition k. Given these, the error term, $E = \|D - W \times H\|_F^2$, can be reworded as:

$$E = \sum_{k=1}^{p} \left\| D^{(k)} - W H_{(k)} \right\|_F^2 \tag{4.2}$$

As discuss next, this error term (which includes data and low-rank approximations from a group of shared data sources) gives rise to an effective MUR-based algorithm for obtaining the NMF for D. The proposed group-MUR (G-MUR) relies on two update rules, one for W and the other for H, both leveraging the special group structure of the error term.

4.3.2 G-MUR for Query Factor Matrix W

To obtain the update rule for factor matrix W, we need to take the derivative of the error function E with respect to the factor W. Relying on the fact [72] that the Frobenius norm of a matrix X can be rewritten in terms of the trace, $Tr(X)$, of X as

$$||X||_F = \sqrt{\sum \sum x_{ij}^2} = \sqrt{Tr(X^T X)} \tag{4.3}$$

and that the trace has the following three properties

$$\begin{aligned} Tr(X + A) &= Tr(X) + Tr(A) \\ Tr(X) &= Tr(X^T) \\ Tr(XA) &= Tr(AX) \end{aligned} \tag{4.4}$$

we first expand the cost function from equation 4.2 based on equations 4.3 and 4.4 as

$$
\begin{aligned}
E &= \sum_{k=1}^{p} Tr((D^{(k)} - WH_{(k)})^T (D^{(k)} - WH_{(k)})) \\
&= \sum_{k=1}^{p} Tr({D^{(k)}}^T D^{(k)} - 2{D^{(k)}}^T WH_{(k)} + WH_{(k)} H_{(k)}^T W^T).
\end{aligned}
$$

Secondly, relying on the fact [72] that the derivative of trace with respect to the first order and second order of matrix X are

$$
\begin{aligned}
\frac{d}{dX} Tr(XA) &= A^T \\
\frac{d}{dX} Tr(XBX^T) &= XB^T + XB
\end{aligned}
\tag{4.5}
$$

I take the derivative of the error function E with respect to the factor W as follows:

$$
\begin{aligned}
\frac{dE}{dW} &= \sum_{k=1}^{p} \left(-D^{(k)} {H_{(k)}}^T + WH_{(k)} {H_{(k)}}^T\right) \\
&= \sum_{k=1}^{p} \left(-D^{(k)} {H_{(k)}}^T\right) + WHH^T.
\end{aligned}
$$

Note that if we first set the derivative of the error term, E, with respect to W to 0 (to find point at which E is minimum) and then replace $D^{(k)}$ with its approximate decomposition in equation 4.1, we can obtain the following update rule for W:

$$
\begin{aligned}
[\textbf{RULE1}] \quad W &\leftarrow \left(\sum_{k=1}^{p} D^{(k)} {H_{(k)}}^T\right) \left(HH^T\right)^{-1} \\
&= \left(\sum_{k=1}^{p} W^{(k)} H^{(k)} {H_{(k)}}^T\right) \left(HH^T\right)^{-1}
\end{aligned}
$$

In other words, we can fix H and improve W by using matrices H, $H_{(k)}$, $H^{(k)}$, and $W^{(k)}$. Last two of these correspond to the factorization of a shared data source's (or unshared data source's) data matrix; on the other hand, query's factor matrix H (and its partitions $H_{(k)}$) is initially unknown and thus selected randomly and improved in the G-MUR process as described next.

4.3.3 G-MUR for Query Factor Matrix H

When updating the factor matrix H for a fixed W, note that the error term specified in equation 4.2 relies on the partitions $[H_{(1)}, H_{(2)}, ..., H_{(p)}]$ of H. Therefore, we take the derivatives of E with respect to individual $H_{(k)}$, separately:

$$\frac{dE}{dH_{(k)}} = -W^T D^{(k)} + W^T W H^{(k)}.$$

Once again, setting the derivative of E with respect to $H_{(k)}$ equal to 0 and replacing $D^{(k)}$ with its approximation in Equation 4.1, we obtain the following update rule for $H_{(k)}$:

$$\begin{aligned}[\textbf{RULE2}] \quad H_{(k)} &\leftarrow W^T D^{(k)} (W^T W)^{-1} \\ &= W^T W^{(k)} H^{(k)} (W^T W)^{-1}\end{aligned}$$

Thus, given an initial decomposition of $D \sim W \times H$, we can fix W and improve each partition $H_{(k)}$ of H by using matrices W, $H^{(k)}$, and $W^{(k)}$, last two of which correspond to the factorization of the shared data source's (or unshared data source) data matrix.

4.3.4 G-NMF Algorithm

Given these group update rules for W and H, the Group-NMF algorithm is presented in Algorithm 3.

Intuitively, the G-NMF algorithm for efficiently obtaining the NMF of the query data matrices by leveraging the NMF of the shared data source's data matrices, the task of factoring the query data matrices is essentially transformed to factoring shared data sources' data matrices and combining the relevant subfactors of shared data sources to obtain the corresponding factors for the queries. This is especially

Algorithm 3 G-NMF (Using G-MUR))

Input:

The data source-query relationship matrix, $\mathbb{M}$; Factor matrices W^{D_j} and H^{D_j} for each shared data source D_j;

A given query's data matrix, D, combining data from relevant shared data sources according to $\mathbb{M}$;

Target rank, r;

Output:

NMF factor matrices, W, and H for the NMF query;

1: **repeat**

2: Update W using [RULE1] and factor matrices of relevant shared data sources according to $\mathbb{M}$

3: Update $H_{(k)}$ for each relevant shared data source, D_k, using [RULE2]

4: Combine all relevant $H_{(k)}$ into the final factor H for the query

5: **until** the stopping condition for G-MUR is met.

useful in scenarios where the number of shared data sources is significantly smaller than the number of continuous NMF queries, which is often the case.

To see the advantage of using G-NMF, let us consider the following case: Let us assume that we have K shared data sources, each of size $m \times m$. Let us also assume that we have Q NMF queries, each using $\frac{K}{2}$ of the shared data sources and has an equal amount of private data sources (i.e., each NMF query involves an $m \times m$ matrix). Assuming the target rank is r, with the basic NMF algorithm, the execution time would be $O(KQm^2r)$. With G-NMF, however, the total work is $O(Km^2r + K\frac{Q}{2}m^2r + 2Qr^2Km)$. Since (a) the target rank r is far smaller than m and (b) Q and K are potentially very large, G-NMF would provide significant gains over basic NMF in this scenario. G-NMF can be further improved if there are clusters of data sources

that are shared across large numbers of queries. If this is the case, G-NMF can be performed hierarchically: first G-NMF would be used to maintain the rank-r factor matrices $W_{\mathcal{C}}$ and $H_{\mathcal{C}}$ of a given cluster, $\mathcal{C} \subseteq \mathcal{T}$, of shared data sources (by treating the cluster as a *virtual queries*) and then these factor matrices would be used to maintain the factor matrices of the individual query who share this cluster (by treating the cluster as a *virtual data source*).

4.4 Group Incremental NMF (GI-NMF)

Since data will be inserted continuously and therefore, even if the number of shared data sources is relatively small, repeatedly applying NMF on the shared data sources' data would be wasteful and costly. Instead, we need a mechanism by which we leverage temporal redundancy and maintain the NMF incrementally, by considering only newly inserted data. In this subsection, I introduce *group incremental non-negative matrix factorization* (GI-NMF), which updates the subfactors of each the shared data source incrementally and then combines the resulting subfactors to obtain the factor matrices of the queries. The key advantages of this approach are that (a) the number of shared data sources is often far less than the number of queries and this reduces the number of matrices we need to explicitly maintain and (b) often shared data sources' data matrices are far smaller than queries' data matrices (which access multiple shared data sources), which significantly reduces computation costs during the incremental update process.

Below, I first introduce a vector-incremental update scheme, which updates the shared data sources' factors incrementally for each update vector.

4.4.1 Vector Incremental Updates on the Shared Data Sources

In this section, I introduce an incremental update scheme for vector streams, by extending the vector-incremental approach proposed in [16]. More specifically, the authors assume that the newly arriving vector affects the previous encoding matrix *H only by adding a new column.* Relying on this assumption, authors propose a cost/error function with respect to the old data matrix, newly arriving vector, and their relative weights. In this section, I build on this (a) by extending it into a group incremental framework and (b) considering block updates (updates involving blocks of data) – as opposed to vector updates (involving individual data vectors).

To develop the update rules for incrementally updating the shared data sources' matrices, let us first assume that, at time $time_i$, the shared data source has a feature-object matrix M_i. Recall that the standard error function of NMF at time stamp $time_i$ is

$$E_i = ||M_i - W_i \times H_i||_F^2,$$

where, W_i and H_i are the two non-negative factors which reconstructed back to M_i. As new data received by the shared data source, we need to modify the error function accordingly. Let us assume that a single new object arrives at time stamp $time_{i+1}$. We can revise the error function for the newly arriving object as

$$\begin{aligned} E_{i+1} &= \alpha E_{i+1}^{old} + (1-\alpha) e_{i+1} \\ &= \alpha E_{i+1}^{old} + (1-\alpha) ||m_{i+1} - W_{i+1} \times h_{i+1}||_F^2, \end{aligned}$$

where (a) m_{i+1} is the vector representing the new object that arrives at time stamp $time_{i+1}$, (b) h_{i+1} is the encoding vector for the new object using the new basis vectors described by W_{i+1}, (c) α is the weighting factor for the contribution of the old objects to the overall error, and (d) E_{i+1}^{old} is the contribution of the old objects to the error at

time stamp $time_{i+1}$. Since, as in [16], we adopt the assumption that the new object simply appends a new column to H_i (to give H_{i+1}), the contribution of the old objects to the error at time stamp $time_{i+1}$ can be computed as

$$\begin{aligned} E_{i+1}^{old} &= ||M_{i+1}^{old} - W_{i+1} \times H_{i+1}^{old}||_F^2 \\ &= ||M_i - W_{i+1} \times H_i||_F^2 \end{aligned}$$

Given the above (and relying on the properties of the Frobenius norm stated in Equations 4.3 and 4.4), we can rewrite E_{i+1} as

$$\begin{aligned} E_{i+1} = \alpha\ Tr(&M_i M_i^T - 2W_{i+1}H_i M_i^T + W_{i+1}H_i H_i^T W_{i+1}^T) \\ &+ (1-\alpha)\ Tr(m_{i+1}m_{i+1}^T - 2W_{i+1}h_{i+1}m_{i+1}^T \\ &+ W_{i+1}h_{i+1}h_{i+1}^T W_{i+1}^T), \end{aligned}$$

where W_i and H_i indicate the NMF factors of time stamp $time_i$.

Vector-Incremental GI-MUR for Shared Data Source's Factor Matrix W.

Given this and using Equation 4.5, we can obtain the derivatives of the cost function, E_{i+1}, with respect to W_{i+1} and h_{i+1}, leading to the new update rules for the two factors. In particular, setting the derivative with respect to W_{i+1},

$$\begin{aligned} \frac{dE_{i+1}}{dW_{i+1}} &= \alpha(-M_i H_i^T + W_{i+1}H_i H_i^T) \\ &\quad + (1-\alpha)(-m_{i+1}h_i^T + W_{i+1}h_{i+1}h_{i+1}^T), \end{aligned}$$

to 0 gives the update rule for factor W_{i+1}:

$$[\textbf{RULE3}] \quad W_{i+1} \leftarrow \frac{\alpha M_i H_i^T + (1-\alpha)(m_{i+1}h_i^T)}{\alpha H_i H_i^T + (1-\alpha)h_{i+1}{h_{i+1}}^T}.$$

Note that h_{i+1} is initially selected randomly and improved in the GI-MUR process.

Vector-Incremental GI-MUR for Shared Data Source's Factor Matrix H.

Similarly, setting the derivative of E_{i+1} respect to h_{i+1},

$$\frac{dE_{i+1}}{dh_{i+1}} = (1-\alpha)(-(m_{i+1}^T W_i)^T + {W_{i+1}}^T W_{i+1} h_{i+1}),$$

to 0 gives the update rule for factor h_{i+1}:

$$[\textbf{RULE4}] \quad h_{i+1} \leftarrow \frac{W_i^T m_{i+1}}{{W_{i+1}}^T W_{i+1}}.$$

Since, as in [16], we adopt the assumption that the new object simply appends a new column to H_i, appending h_{i+1} to H_i to gives the factor H_{i+1} at time stamp $time_i$.

Summary

One major advantage of this process is that I can incrementally update the shared data source's factors W and H without involving the original data matrix which significantly reduces the storage requirement. Moreover, the intermediate matrices, $W_i^T m_{i+1}$ and $M_i H_i$, need to be computed only once during one update cycle, which reduces the online computation cost significantly.

Note that, using these two update rules, I can maintain the two subfactors of each shared data source individually. The NMF factors of a given NMF query's data matrix can then be obtained from the subfactors of the relevant shared data sources as described next.

4.4.2 Combining Subfactors to Obtain Query's Factors

In the previous subsection, we have discussed how to obtain the non-negative factors for each shared data source's data matrix in a given update cycle. Given

these, we extend the Group-MUR (G-MUR) technique discussed in Section 4.3.1 to combine these subfactors (using the update rules [RULE1] and [RULE2]) according to each data source-query relationship to obtain the factors for each individual query.

GI-MUR for NMF Query's Matrix W.

Remember from Section 4.3.1 that, for a given user query s, with an $n \times m$ non-negative matrix $D(\sim W \times H)$, we partition D into $p \leq m$ partitions along its columns ($D = [D^{(1)}, D^{(2)}, ...D^{(p)}]$), such that for each $k \leq p$, $D^{(k)}$ is a data matrix from a shared data source or a unshared data source (according to the data source-query matrix $\mathbb{M}$ which describes the coverage relationships between the shared data sources and continuous NMF queries).

$$W \leftarrow \left(\sum_{k=1}^{p} W^{(k)} H^{(k)} {H_{(k)}}^T \right) \left(H H^T \right)^{-1},$$

where $W^{(k)}$ and $H^{(k)}$ are the non-negative factors corresponding to shared data sources k and $H_{(k)}$ indicates the part *in the overall factor* H of (data matrix D) corresponding to shared data source k. Given this, an NMF query's non-negative factor W at time stamp $time_{i+1}$ can be updated as:

$$[\textbf{RULE5}] \;\; W_{i+1} \leftarrow \left(\sum_{k=1}^{p} W_{i+1}^{(k)} H_{i+1}^{(k)} {H_{(k)}}_{i+1}^T \right) \left(H_{i+1} H_{i+1}^T \right)^{-1}$$

Here, p is the number of shared data sources that the query is accessing and $W_{i+1}^{(k)}$ and $H_{i+1}^{(k)}$ are the non-negative factors of those corresponding shared data sources, which are incrementally maintained using [RULE3] and [RULE4]. H_{i+1} is the non-negative factor of this user query at time stamp $time_{i+1}$, while ${H_{(k)}}_{i+1}$ are the partitions of H_{i+1} as we have described before. Note that H_{i+1} (and ${H_{(k)}}_{i+1}$) are initially selected randomly and improved in the GI-MUR process as described next.

GI-MUR for NMF Query's Matrix H.

Since group update rules(G-MUR) for $H_{(k)}$ (the part of user query's factor matrix H corresponding to shared data source k) is given in [RULE2] as

$$H_{(k)} \leftarrow W^T W^{(k)} H^{(k)} (W^T W)^{-1}.$$

we can update the ${H_{(k)}}_{i+1}$ at time stamp $time_{i+1}$ using the update rule

$$[\textbf{RULE6}] \;\; {H_{(k)}}_{i+1} \leftarrow W_{i+1}^T W_{i+1}^{(k)} H_{i+1}^{(k)} (W_{i+1}^T W_{i+1})^{-1}$$

Given this, horizontally concatenating all ${H_{(k)}}_{i+1}$ will give the query's non-negative factor, H_{i+1}, at time stamp $time_{i+1}$.

4.4.3 Block Updates of Shared Data Source's Factors

As mentioned earlier, the vector incremental update scheme for shared data sources' factors, described in Section 4.4.1, has a significant drawback: if the number of updates is large, then though the cost for updating one vector is small, the cumulative cost for updating the shared data sources' factors can be very large. To tackle this problem, in this subsection, I introduce a *block update scheme* for the shared data sources.

Let us focus on one of the shared data sources, D_k, at time stamp $time_i$. Assume that, at time $time_i$, the feature-object matrix, M_i, of shared data source D_k was of size $n \times m$ and let the updates at time $time_{i+1}$ be represented as a feature-object matrix, ΔM_{i+1}, of size $n \times m'$. We can naturally save significant amount of time if we can obtain the updated non-negative factors by considering the update matrix, ΔM_{i+1}, as a whole instead of considering the individual object updates one at a time.

Recall that the G-NMF algorithm we have introduced in Section 4.3.1 treats the feature-object matrix $M_{i+1} = M_i \oplus \Delta M_{i+1}$ as the horizontal concatenation of M_i and ΔM_{i+1}. Our key observation is that since we already have the non-negative factors of $M_i (\sim W_i \times H_i)$ from the previous update cycle, if we can obtain the non-negative factors for the update matrix $\Delta M_{i+1} (\sim \Delta W_i \times \Delta H_i)$, we may be able to combine these to obtain the factors of $M_{i+1} (\sim W_{i+1} \times H_{i+1})$.

Note that since ΔM_{i+1} is in practice *much smaller* than M_i, its factors ΔW_i and ΔH_i can be obtained efficiently by directly applying a naive NMF algorithm, such as the multiplicative update rules in [54]. Therefore, the remaining challenge is to revise the update rules [RULE3] and [RULE4] for shared data sources' factors for block updates. We propose to achieve this by rewriting G-MUR, [RULE1] and [RULE2], presented in Section 4.3.1 in a way that accounts for ΔW_i and ΔH_i.

4.4.4 Block GI-MUR for Shared Data Source's Factor Matrix W.

Since, the non-negative feature-object matrix M_{i+1} consist of two partitions, M_i and ΔM_{i+1}, with known factor matrices, we can revise the update rule [RULE1] in Section 4.3.1 to obtain the factor W at time stamp $time_{i+1}$ as follows:

$$[\textbf{RULE3/B}] \;\; W_{i+1} \leftarrow (W_i H_i {H_{(M_i)_{i+1}}}^T + \Delta W_{i+1} \Delta H_{i+1} {H_{(\Delta M_{i+1})_{i+1}}}^T)(H_{i+1} H_{i+1}^T)^{-1}$$

Here (a) W_i and H_i are the non-negative factors of M_i, (b) ΔW_{i+1} and ΔH_{i+1} are the non-negative factors of ΔM_{i+1} at time stamp $time_{i+1}$, and (c) $H_{(M_i)_{i+1}}$ and $H_{(\Delta M_{i+1})_{i+1}}$ are the two partitions of the shared data source's factor matrix H_{i+1}, corresponding to M_i and ΔM_{i+1}, respectively. Note that $H_{(M_i)_{i+1}}$ and $H_{(\Delta M_{i+1})_{i+1}}$ are initially selected randomly and improved in the GI-MUR process as described next.

4.4.5 Block GI-MUR for Shared Data Source's Factor Matrix H.

Unlike before where we were maintaining a single matrix H, when performing block updates of the shared data source's factor matrix H, we need to incrementally maintain two matrices, $H_{(M_i)_{i+1}}$ and $H_{(\Delta M_{i+1})_{i+1}}$ in the process. We achieve this by revising the update rule [RULE2] in Section 4.3.1 to account for ΔW_{i+1}, ΔH_{i+1}, and W_{i+1} (maintained by update rule [RULE3/B]):

$$[\mathbf{RULE4/B_1}] \;\; H_{(M_i)_{i+1}} \leftarrow W_{i+1}^T W_i H_i (W_{i+1}^T W_{i+1})^{-1}$$

$$[\mathbf{RULE4/B_2}] \;\; H_{(\Delta M_{i+1})_{i+1}} \leftarrow W_{i+1}^T \Delta W_{i+1} \Delta H_{i+1} (W_{i+1}^T W_{i+1})^{-1}$$

Given these, the factor matrix H_{i+1} is obtained by concatenating the two partitions $H_{(M_i)_{i+1}}$ and $H_{(\Delta M_{i+1})_{i+1}}$.

4.4.6 Summary

When the block GI-MUR process for the shared data sources finishes, we have W_{i+1} and H_{i+1} for each shared data source. Given these, we then apply update rules [RULE5] and [RULE6] discussed in Section 4.4.2 to suitably combine the shared data sources' factors to obtain the individual query's factor matrices at time stamp $time_{i+1}$.

4.4.7 GI-NMF Algorithm

The pseudo-code of the GI-NMF algorithm for incrementally updating shared data sources' and queries' factor matrices using the block update scheme is presented in Algorithm 4. As we see in the experiment results reported in the next section, the GI-NMF algorithm provides significant improvements in performance, especially when query overlaps are large and the database updating rates are high. These

Table 4.1: System and Data Parameters

	Description	**Default**	**Alt. 1**	**Alt. 2**
σ	Number of NMF queries involved	50	30	70
h	Number of sharable data sources involved (in percentage of total number of queries)	40%	60%	20%
r	Target rank	50	100	25
n	Number of distinct features	1500	2500	500
l	Number of distinct features per object	300	750	100
$Coverage$	Each NMF query's coverage by shared data source	50%	80%	20%
$u(D_k)$	Update rate for shared data source D_k chosen from a uniform distribution based on	100	150	200
$u(q_j)$	Unshared data source update rate for query q_j	50%	100%	25%

multiplicative rules have a relative low time complexity but the convergence is not achieved fast [75].

4.5 Experiments

In this section, we evaluate the efficiency and effectiveness of our Group Incremental NMF algorithm for different scenarios and parameter settings (Table 4.1). In particular, we consider two continuous query applications based on a document clustering task, which has been shown to benefit from NMF based analysis, in particular for bootstrapping generative models of the data [31, 88].

4.5.1 Synthetic Data Streams and Queries

Synthetic data sets allow us to freely vary the characteristics of the data and updates to observe the accuracy and efficiency of our algorithms under different scenarios: We generate $u(D_k)$ new random data objects per update cycle for k^{th} sharable data source's and $u(q_j)$ objects per cycle from private data source for j^{th} query, as described in Section 4.2. Note that the total number of objects received by the j^{th} query per update cycle is $u(q_j) + \sum_{\mathbb{M}_{jk}=1} u(D_k)$. The random objects have a density of 0.4, which means an object includes around 40 percent of the features. Also, we assume that at every $c = 10$ steps, the system refreshes their entire NMF (to prevent error accumulation), with $c \times u(D_k)$ documents for k^{th} shared data source and $c \times (u(s_j) + \sum_{\mathbb{M}_{jk}=1} u(D_k))$ objects for the j^{th} query.

4.5.2 Document Data Stream and Queries

While synthetic data streams enable us to study the algorithms under different settings, they may not necessarily correspond to real data content and topic evolution. To address this, we generated a second data stream and associated queries using the popular and well-studied DBLP data set [30]: more specifically

- we use real data content and topic evolution (publications at conferences at different years) and real registered queries (authors),
- but as explained below, we simulate the arrival rate by speeding up the time clock.

The above setup enables us to observe *whether G-NMF and GI-NMF schemes can indeed leverage the various redundancies in data streams with real content:*

- *Objects and Features:* For the experiments reported in this section, we used

28,569 document abstracts from 20 conferences spanning four high-level content areas: databases, data mining, information retrieval and artificial intelligence. These abstracts contained 11,771 unique terms that were treated as features.

- *Data Sources:* Each conference is treated as an individual data source, with the different years serving as a stream timestamps.

  Note that, obviously, the *real* arrival rate of the conference publications is very slow (few hundred articles per year per conference). As we mentioned earlier, since our goal is to assess the scalability of the proposed incremental algorithms based on real data content (such as term distribution, data sharing, and topic evolution), we use the publication timestamps only to indicate the data arrival order – to assess scalability, we significantly speed-up the time clock: For each conference, we select 50% of all the articles as the base data at $t = 0$ and 15% of its articles as its per-cycle update set. In other words, we consider a total of 4 time instances: the base data set at time, $t = 0$, contains $\sim 14K$ abstracts; whereas at each of the following three time instances, the systems receives a total of $\sim 5K$ abstracts.

- *Registered Continuous Queries:* These articles were written by 28,702 authors. As continuous analysis tasks, we selected a random subset of the authors and tracked of each author's interest (for example to support personalized recommendations) by continuously analyzing the conferences in which s/he publishes.

  More specifically, for each author we picked the top 5 conferences where s/he publishes and used the set of papers published in these conferences (along with a randomly selected subset of papers published in non top-5 conferences) to roughly characterize the evolution of the author's research interest.

  Note that the set of top-5 conferences for a given author is treated as sharable

data sources and the set of randomly selected subset of papers from the rest is treated as un-shared, author private data source.

Important: Due to performance limitations of the baseline NMF (which con not leverage sharing), in these experiments, we consider only 50 registered queries (i.e., 50 authors with the highest number of publications). As we later see in Section 4.5.6, the proposed GI-NMF algorithms become increasingly advantageous as the number of queries increases. Therefore, the GI-NMF gains presented in Section 4.5.5 on the DBLP data set should be seen as lower-bounds in performance gain.

4.5.3 Algorithms and Evaluation Criteria

In the experiments, we compare

- *NMF* – NMF algorithm using the multiplicative update rules described in [54];
- *G-NMF* – group NMF algorithm described in Section 4.3 (used only for decomposing an initial base matrix);
- *I-NMF* – incremental NMF algorithm described in [16]; each query's data matrix is incrementally maintained at every update cycle;
- *GI-NMF(V)* – group incremental NMF with vector updates, Section 4.4.1; and
- *GI-NMF(B)* – group incremental NMF with block updates, Section 4.4.3,

for execution time and reconstruction accuracy. In particular, we define *reconstruction error overhead* (err_{rel}) as

$$\frac{1}{len}\sum_{i=1}^{len}\frac{rec\_error(\hat{X}_{i,*}, X_i) - rec\_error(\hat{X}_{i,NMF}, X_i)}{rec\_error(\hat{X}_{i,NMF}, X_i)},$$

where

- *len* is the number of iterations (length of the stream),
- $\hat{X}_{i,*}$ denotes the factorization of the data matrix at time i obtained using the algorithm "$*$", and
- $rec\_error(Y, X)$ denotes the reconstruction error of the decomposition Y against the data matrix X, measured in terms of $Frobenius$ norm.

Note that a low-rank decomposition of X_i would lead to a reconstruction error, even if it is obtained using full NMF followed by selection of the top r components. Therefore, the denominator of the above term is not equal to 0.

4.5.4 Experiment Configurations

All experiments were conducted using an 8-core Intel(R) Xeon(R) X5570 2.93GHz CPU and 24GB memory Machine. The codes were executed using Matlab 2012b. To ensure the fairness for all MUR involved processes, we use a fixed number (25) of iterations. We also add a very small ϵ value (10^{-9}) to the denominator of all update rules to avoid zero denominators. For vector based incremental algorithms, I-NMF and GI-NMF(V), we use $\alpha = 0.96$ as the weight factor of existing documents relative to new ones (as recommended by [68]).

4.5.5 Results under Default Settings

Synthetic Data Streams (Figure 4.3 and 4.4).

As the two figures show, the proposed G-NMF and block GI-NMF processes which leverage the overlaps provide significant improvements on the execution time, with only a negligible error overhead (0.5% over the error of NMF). Importantly, the standard I-NMF scheme (which does not leverage any redundancies) perform very

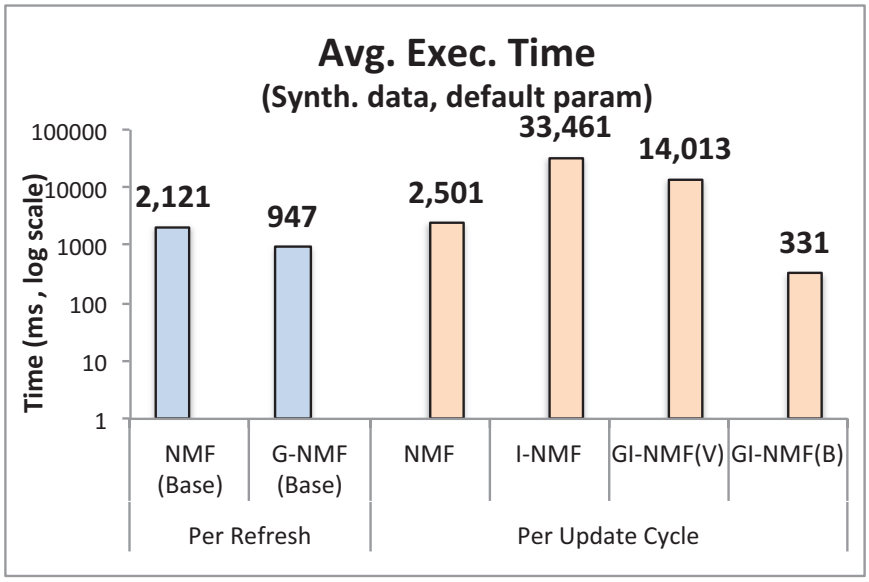


Figure 4.3: Efficiency results for the synthetic trace data set under default settings

Table 4.2: Efficiency Results for the DBLP Data Stream (Relative to NMF)

# **Authors**	**Refresh Time Gain(G-NMF)**	**Update Cycle Time Gain(I-NMF)**	**Update Cycle Time Gain(GI-NMF(B))**
25	17.1%	-1124% (i.e., loss)	16.3%
50	23.9%	-1143% (i.e., loss)	25.1%
100	32.8%	-1123% (i.e., loss)	25.3%

poorly – even worse than NMF in execution time. The vector incremental GI-NMF significantly reduces the cost, but still cannot compete with NMF when update rate per cycle is high. The proposed block GI-NMF, however, performs extremely fast and with very competitive accuracy.

Document Data Streams (Tables 4.2 and 4.3).

While the absolute values of the time and error rates are different (due to lower degree of coverage, ≤ 5 queried conferences per author), the overall trends with the DBLP data confirms the above observations: standard I-NMF is very costly and results

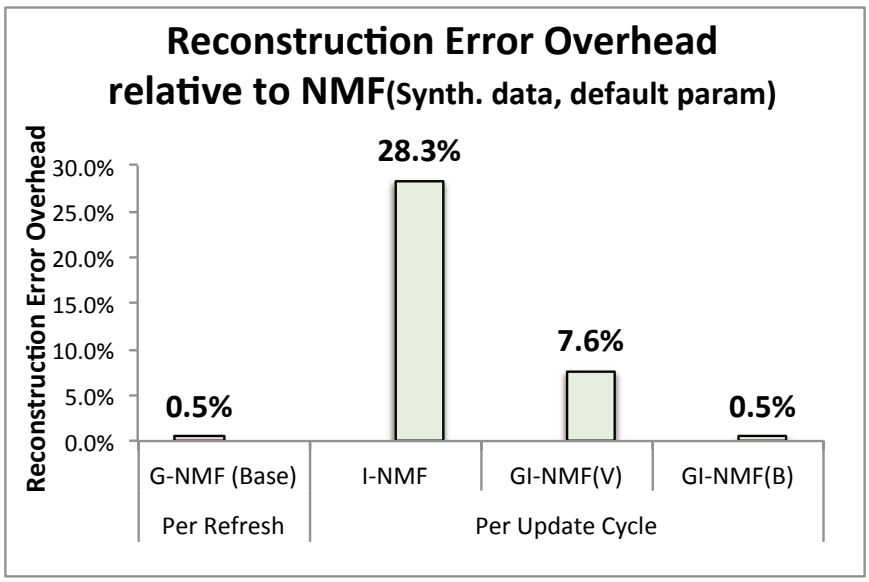


Figure 4.4: Accuracy results for the synthetic data stream under default settings

Table 4.3: Accuracy Results for the DBLP Data Stream (Relative to NMF)

# Authors	**Refresh Error Ovhd.(G-NMF)**	**Update Cycle Error Ovrhd.(I-NMF)**	**Update Cycle Error Ovrhd.(GI-NMF(B))**
25	3.0%	9.1%	3.4%
50	3.1%	9.1%	3.7%
100	3.1%	9.1%	6.1%

in higher error rates. Note that, for block based GI-NMF, as the number of authors increases the time gain increases, whereas the error overhead remains relatively small. We next detail the impacts of various parameters on the performance of block based GI-NMF using synthetic data sets.

4.5.6 Impacts of Data and System Parameters

Table 4.4 presents the impacts of the various parameters on the efficiency and effectiveness on the random data stream. In particular, we focus on NMF and block

Table 4.4: Impacts of Various Parameters on Performance of GI-NMF (Relative to NMF)

Parameter	**Refresh Time Gain**	**Refresh Error Ovrhd.**	**Update Cycle Time Gain**	**Update Cycle Error Ovrhd.**
$\sigma = 30$ (# queries)	52.36%	0.45%	86.20%	0.46%
$\sigma = 70$	65.30%	0.48%	93.86%	0.51%
$h = 20\%$ (# shared data sources – perc. of queries)	57.46%	0.41%	85.53%	0.42%
$h = 60\%$	59.05%	0.45%	87.25%	0.47%
$n = 500$ (# terms)	44.91%	1.66%	75.83%	1.72%
$n = 2500$	61.09%	0.27%	92.43%	0.29%
$l = 100$ (# terms per document)	59.12%	0.91%	87.09%	1.09%
$l = 750$	59.64%	0.44%	87.69%	0.45%
$Coverage = 20\%$ (shared data sources per queries)	52.64%	0.44%	85.11%	0.45%
$Coverage = 80\%$	58.27%	0.48%	90.47%	0.49%
$u(D_k) = uni(150)$ (shared data sources' update rate)	59.29%	0.47%	91.89%	0.48%
$u(D_k) = uni(200)$	63.49%	0.49%	92.69%	0.50%
$u(q_j) = 25\%$ (queries' unshared data source updating rate)	83.55%	0.45%	87.35%	0.48%
$u(q_j) = 100\%$	43.63%	0.47%	88.67%	0.48%
$r = 25$ (target rank)	58.30%	0.28%	93.90%	0.28%
$r = 100$	44.00%	0.57%	76.19%	0.59%

GI-NMF techniques which prove to be competitive with high update rates.

The Table 4.4 shows that block GI-NMF is consistently more effective in terms of time gain as the problem size (number of terms, percentage of shared data sources involved, updating rate, unshared data source updating rate) increases, again with negligible further loss. Experiments confirm that block GI-NMF saves more time and is more accurate especially for low rank NMF applications.

4.6 Conclusion

Recognizing that, in continuous NMF workloads, there are significant redundancies that can be leveraged for characterizing data sources/queries based on the data/document streams they access, I develop a *group incremental non-negative matrix factorization* (GI-NMF) approach for time-evolving data sets. The proposed GI-NMF algorithm solves multiplicative update rule (MUR) subproblems by partitioning the original data matrices into submatrices (representing data shared by multiple queries) and combining the subfactors into the final factors. Moreover, the GI-NMF algorithm maintains each partition's subfactors incrementally as the matrices change over the time in order to scale to the needs of time-evolving data. Experiment results showed that the proposed block GI-NMF technique provides significant time gains in maintaining NMF results, with negligible impact on accuracy.

As discussed in Section 2.6, in addition to factorization model, probabilistic generative model is another method of extracting the latent patterns of the data, and it takes advantage of the Bayesian theory and try to infer the hyperparameter of a graphical model. However, there are some fundamental challenges due to the nature of the model, such as the length of interest. I will present a formal problem and challenge and how to solve it in the next Chapter.

Algorithm 4 Block Incremental GI-NMF

Input:

Each shared data source D_j data matrix, $M_i^{D_j}$, and its two NMF factors $W_i^{D_j}$ and $H_i^{D_j}$ from the previous time stamp $time_i$;

New objects for each shared data source $\Delta M_{i+1}^{D_j}$;

The data source-query relationship matrix, $\mathbb{M}$;

Target rank, r;

Output:

The NMF factor matrices at time stamp $time_{i+1}$ for all shared data sources and queries;

1: **for all** each shared data source D_j **do**
2: Apply naive NMF algorithm to obtain non-negative factors $\Delta W_{i+1}^{D_j}$ and $\Delta H_{i+1}^{D_j}$ for new objects ΔM_{i+1}.
3: **repeat**
4: Update $W_{i+1}^{D_j}$ using [RULE3/B].
5: Update $H_{(M_i)_{i+1}}^{D_j}$, $H_{(\Delta M_{i+1})_{i+1}}^{D_j}$, and $H_{i+1}^{D_j}$, using [RULE4/B_1] and [RULE4/B_2].
6: **until** stopping condition for GI-MUR is met.
7: **end for**
8: **for all** each query q_j **do**
9: **repeat**
10: Update $W_{i+1}^{q_j}$ using [RULE5], shared data source' factors, and the data source-query matrix, $\mathbb{M}$,
11: Update $H_{i+1}^{q_j}$ using [RULE6], shared data sources' factors, and the data source-query matrix, $\mathbb{M}$,
12: **until** stopping condition for GI-MUR is met.
13: **end for**

Chapter 5

INCREMENTAL MULTI-SCALE DYNAMIC TOPIC MODELS ON DATA STREAMS

5.1 Introduction

Dynamic topic models (DTM) are commonly used for mining latent topics in evolving web corpora. In this work, noted that a major limitation of the conventional DTM based models is that they assume a predetermined and fixed scale of topics. In reality, however, topics may have varying spans and topics of multiple scales can co-exist in a single web or social media data stream. Therefore, DTMs that assume a fixed epoch length may not be able to effectively capture latent topics and thus negatively affect accuracy. Here, I present a Multi-Scale Dynamic Topic Model (MS-DTM) and a complementary Incremental Multi-Scale Dynamic Topic Model (IMS-DTM) inference method that can be used to capture latent topics and their dynamics simultaneously, at different scales. In this model, topic specific feature distributions are generated based on a multi-scale feature distribution of the previous epochs; moreover, multiple scales of the current epoch are analyzed together through a novel multi-scale incremental Gibbs sampling technique. We show that the proposed model significantly improves efficiency and effectiveness compared to the single scale dynamic DTMs as well as prior models that consider only multiple scales of the past.

5.2 Problem Formulation

Table 5.1 presents key notations that are used. Dynamic Topic Modeling (DTM) considers a corpus stream that contains documents generated sequentially over a fixed vocabulary set and assumes that the timeline is split into fixed size epochs. At epoch t, there are D^t documents and a document d_i^t in this epoch is represented as a set of words, i.e. $d_i^t = \{w_1^t, w_2^t, ..., w_{|d_i^t|}^t\}$, where $|d_i^t|$ is the vocabulary size of document d_i^t. DTM infers, a set, L^t, of K latent topics and associate a latent topic, $k_{i,j}^t$, to each word/document pair $\langle i, j\rangle$ that occurs at time epoch t. [11] addresses this by extending the static latent Dirichlet allocation (LDA [14]) model along time. Data is divided into slices (epochs) and each slice is modeled with a K component topic model. The collection of topic models are tied by chaining topics and topic proportions across consecutive slices.

5.2.1 DTM at Multiple Scales

As discussed in the Introduction, this basic dynamic topic model has several difficulties, including the fact that the length of the epoch has to be decided ahead of the time. Moreover, basic DTMs do not recognize that multiple time scales may be relevant for the inference task: the list of active topics may be predicted by a mixture of past topics of different lengths and current topics may last for different scales. Therefore, in order to accurately model the latent topics and their dynamics, we need to model multiple scales of past and current topics. We consider a stream of documents generated over a fixed vocabulary:

- The timeline is split into S many scales of epochs of different lengths. The length of the smallest epoch is l_{min} and the difference of lengths of t consecutive scales s_{h+1} an s_h is also l_{min} (in other words, the length of the scale s is $s \times l_{min}$).

Table 5.1: Notations

Symbol	Description
l_{min}	Length of the smallest time epoch
S	Number of scales
α^t	Dirichlet prior for the topics at time epoch t
K	Number of latent topics to be inferred
W	Vocabulary, the unique word set
N	Total number of words in the corpus
D^t	Documents at time epoch t
$D^{t,s}$	Documents at time epoch t with time scale s
N_i^t	Number of words in the i^{th} document at time epoch t
$w_{i,j}^t$	j^{th} word in the i^{th} document at time epoch t
$k_{i,j}^t$	Topic of j^{th} word in the i^{th} document at time epoch t
θ_i^t	Multinomial distribution over topics for the i^{th} document at time epoch t
ϕ_k^t	Multinomial distribution over words for the k^{th} topics at time epoch t
$\psi_{k,m}^t$	Multinomial distribution over words for the k^{th} document at time epoch t in the m^{th} scale in the past
μ_m^t	Weighting factor for the m^{th} scale in the past at time epoch t

All S scales of epoch t start at the same time, but last for different durations.

- At epoch t at scale s, there are $D^{t,s}$ documents and a document $d_i^{t,s}$ in this epoch is represented as a set of words, i.e. $d_i^{t,s} = \{w_1^{t,s}, w_2^{t,s}, ..., w_{|d_i^{t,s}|}^{t,s}\}$, where $|d_i^{t,s}|$ is the vocabulary size of document $d_i^{t,s}$.

The goal of MS-DTM is to infer, for each epoch, t at scale s, a set, $L^{t,s}$, of K latent topics and associate a latent topic, $k_{i,j}^{t,s}$, to each word/document pair $\langle i, j\rangle$ that occurs

at time epoch t at scale s.

5.2.2 Incremental Multi-Scale Inference

Due to the non-conjugacy of the Gaussian and multinomial models, inference is often done using approximate techniques, such as variational methods [11] or Gibbs sampling [39] (as suggested in [47]), the online inference and parameter estimation can be efficiently achieved by a stochastic EM algorithm, where collapsed Gibbs sampling of latent topics and the maximum likelihood estimation of hyper-parameters are alternately performed). A key difficulty in a multi-scale approach is that the overall work can multiply, rendering the multi-scale approach impractical. Therefore, we need new incremental inference techniques, such as incremental multi-scale Gibbs sampling, to prevent the need to independently Gibbs sample for different time scales.

5.3 Multi-Scale Dynamic Topic Model (MS-DTM)

In this section, we will introduce the proposed multi-scale DTM (MS-DTM) model that captures evolution of topics of multiple scales. Since multi-scale analysis is performed on past epochs and the current epoch, MS-DTM can be considered in two parts, one dealing with the past and the other dealing with "now".

5.3.1 Multi-Scale Modeling of the Past

In order to model the impacts of the past documents towards the topics in the current epoch, (a) I consider multiple time scales of word distributions from the previous documents that have been seen and (b) assume that the topic-specific word distribution for the current epoch is a linear combination of the previous word distributions [47]. To be more specific, the topic-specific word distribution ϕ_k^t for topic k at the current epoch, t, is computed as a function of the past word distributions as

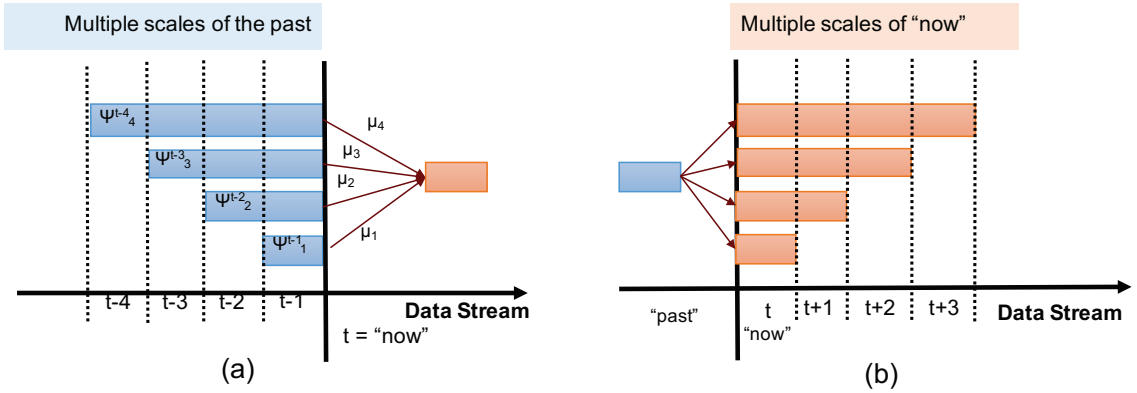


Figure 5.1: Multi-scale Modeling of the Past(a) and "Now"(b)

follows:

$$\phi_k^t \sim Dirichlet(f(\psi_{k,1}^{t-1}, \psi_{k,2}^{t-2}, \ldots, \psi_{k,S}^{t-S})), \tag{5.1}$$

where $f()$ is a function that incorporates the previous word distributions. We enforce that the mean of the Dirichlet parameter for current epoch is proportional to the weighted sum of the word distributions at the previous epochs, i.e.

$$f(\psi_{k,1}^{t-1}, \psi_{k,2}^{t-2}, \ldots, \psi_{k,S}^{t-S}) = \sum_{m=1}^{S} \mu_{k,m}^t \psi_{k,m}^{t-m}. \tag{5.2}$$

Here, $\mu_{k,m}^t$ are weighting factors that relate the word distributions of the previous epochs to the word distributions of the current epoch and will be learned using the documents in the current time epoch. As visualized in Figure 5.1(a), these weighting factors enable us to infer the impact of the past scales on the current epoch: if the topics in the current epoch solely depend on the topics in the immediate past, then we would expect that the weighting factor, $\mu_{k,1}^t$, would be large; in contrast, if the current documents are more likely to be influenced by topics of larger temporal scales in the past, then the weighting factors for $m \gg 1$ should be higher. Given this, for each topic $k = 1, 2, ..., K$ at epoch t, we can pick the topic-specific word distribution as $\phi_k^t \sim Dirichlet(\sum_{m=1}^{S} \mu_{k,m}^t \psi_{k,m}^{t-m})$, and, for each document $d_i^t \in D^t$, we can obtain a topic distribution, $\theta_i^t \sim Dirichlet(\alpha^t)$ relying on the Dirichlet prior for the topics at

epoch t. Given these, we can then select words in the document by first picking topics, $k \sim Multinomial(\theta_i^t)$, using the topic distribution and then picking corresponding words, $w \sim Multinomial(\phi_k^t)$, using the topic-specific word distribution picked at the beginning.

5.3.2 Multi-Scale Modeling of "Now"

In the previous subsection, we have shown how MS-DTM models the multinomial distribution over words for each topic and relates them, through weighting factors, to the current multinomial distribution over words for these topics. While the above model relates multiple scales of the past with a single scale of now, since "now" can also be considered at multiple scales, we need to extend the model to also account for multiple current scales.

As visualized in Figure 5.1(b), we achieve this by associating S scales to each topic t. All these S scales start concurrently; however, they span different durations. As stated in Section 5.2.1, if the length of the smallest epoch at scale 1 is l_{min}, then the length of the epoch at scale s is equal to $s \times l_{min}$. Given this, we can revise the generative process for the various scales of epoch t as follows:

For each current scale $s = 1, 2, ..., S$ at epoch t

(a) For each topic $k = 1, 2, ..., K$ at epoch t scale s

- $\phi_k^{t,s} \sim Dirichlet(\sum_{m=1}^{S} \mu_{k,m}^t \psi_{k,m}^{t-m})$

(b) For each document $d_i^{t,s} \in D^{t,s}$ at epoch t scale s

i. Draw $\theta_i^{t,s} \sim Dirichlet(\alpha^{t,s})$

ii. For each word in the document $d_i^{t,s}$

- Select a topic $k \sim Multinomial(\theta_i^{t,s})$
- Select a word $w \sim Multinomial(\phi_k^{t,s})$

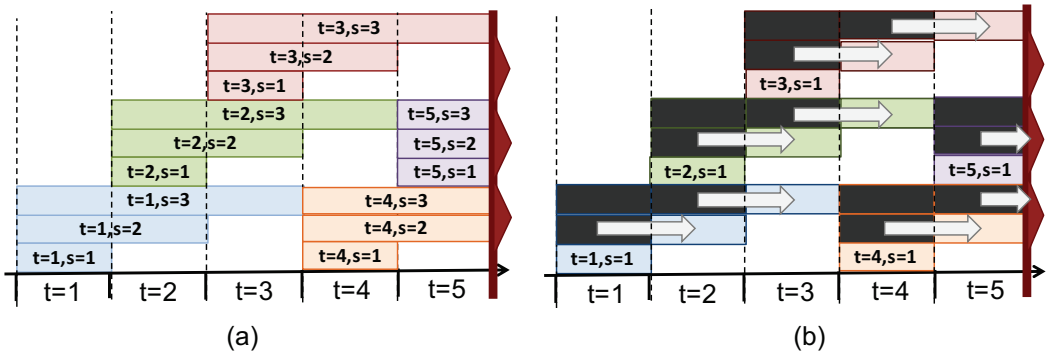


Figure 5.2: Naive vs. Incremental Multi-scale Gibbs Sampling

5.4 IMS-DTM: Incremental Multi-Scale Dynamic Topic Model Inference

In the previous section, we presented the proposed multi-scale DTM (MS-DTM) model, which associates multiple scales to each epoch and analyzes the relationships among the topics in these scales. This is visualized in Figure 5.2(a). In this example, each epoch is associated with three scales that start simultaneously, but last for different durations. From this example, however, it should be clear that if we naively execute the inference process (e.g., we sample for epochs of all scales) the amount of work will increase significantly. In this section, I discuss how to avoid this potential difficulty within the proposed multi-scale approach.

5.4.1 Overlaps across Scale-Epoch pairs

We can readily notice in Figure 5.2(a) is that the different scales of different epochs overlap with each other: in fact, in this example where we have 3 scales for each epoch, up to 6 scale-epoch pairs may overlap (see epochs $t = 3, 4, 5$ in the figure).

A second thing that we can easily notice from Figure 5.2(a) is that the smallest scale of a given epoch, t, is covered by not only the larger scales of the same epoch, but also the larger scales of the epochs that pre-date it. In general, given S scales, the smallest scale of a given epoch, t, is covered by, $S-1$ scales of the same epoch as well

as by the scales $j+1$ through S of the time epoch $t-j$, for $1 \leq j \leq S-1$. Therefore, the total amount of overlaps of the smallest scale at epoch t can be computed as $S + \sum_{j=1}^{S-1} \Big(S - \big(j+1\big) + 1 \Big)$. In the above example, since $S = 3$, this would lead to $3 + (6/2) = 6$ overlaps, which can be confirmed by counting the overlaps for epochs $t = 3, 4, 5$.

While this looks like it can cause serious efficiency problems (a given document may be relevant for a quadratic number of overlapping epoch-scale pairs), Figure 5.2(b) shows that this is not the case. While it is true that the number of overlaps can be quadratically large, if we can reuse the work done for the smallest scale at epoch t for $S-1$ scales of the same epoch as well as by the corresponding scales of the previous epochs, then we can potentially reduce the amount of work back to linear in the number of scales. In the figure, it is shown that, if the Gibbs sampling performed for the smallest scale at epoch t can be leveraged also for the larger scales, this can help eliminate the need to collect a large number of Gibbs samples. We discuss how to enable this reuse next.

5.4.2 Multi-Scale Collapsed Gibbs Sampling

Collapsed Gibbs sampling is a common technique to infer latent topics [39]. It integrates out the variables controlling multinomial distribution over topics documents, i.e. θ and multinomial distribution over words for topics, i.e. ϕ, while only the latent topic variable k is sampled. In particular, the topic assignment of word v is sampled according to its conditional distribution,

$$P(k_v | k_{\mathcal{N}\setminus v}, W_{\mathcal{N}}) \propto \frac{n^{w_v}_{k_v,\mathcal{N}\setminus v} + \beta}{n^{(.)}_{k_v,N\setminus v} + W\beta} \times \frac{n^{d_v}_{k_v,\mathcal{N}\setminus v} + \alpha}{n^{(d_v)}_{.,\mathcal{N}\setminus v} + K\alpha} \tag{5.3}$$

where α and β are Dirichlet prior for the topics and words respectively, $\mathcal{N} \setminus v$ is the set minus, $n^{w_v}_{k_v,\mathcal{N}\setminus v}$ is the number of times that word w_v is assigned to topic k_v,

and $n^{d_v}_{k_v,\mathcal{N}\setminus v}$ is the number of times a word in document d_v is assigned to topic k_v. Given this, collapsed Gibbs sampling iterates through all words in all documents to approximate the posterior distribution $P(k_{\mathcal{N}}|W_{\mathcal{N}})$.

Since the counts of all words in all documents are considered, when new documents are added, the posterior distribution can change – which implies that assignments of the words to the topics may change with new data. Therefore, collapsed Gibbs sampling cannot be directly used when data evolves. [18] expands collapsed Gibbs sampling to the cases where the set of documents evolves over time, by relying on a decayed MCMC approach: In particular, it keeps a *rejuvenate list*, which contains the topic assignments of some previously seen words. When new documents arrive, it re-samples the topic variables for the words in the rejuvenate list and new samples may alter the word-topic assignments.

We adopt [18] to develop an incremental, multi-scale Gibbs sampler. In particular, for each epoch, t, we apply collapsed Gibbs sampling on the documents, $D^{t,1}$, in smallest scale, $s = 1$. Then, second scale is incrementally sampled from scale $s = 1$, while each later scale can be incrementally sampled from its previous scale. In other words, to obtain the collapsed Gibbs sampling for the S scales of epoch t, we apply collapsed Gibbs sampling on documents in $D^{t,1}$, for each scale $s = 2$ to S, and then $\Delta = diff(D^{t,s}, D^{t,s-1})$ is computed, and incremental Gibbs sampling on documents in Δ is applied.

5.4.3 Multi-Scale Online Inference

In this section, we discuss how we implement efficient online inference in IMS-DTM. In particular, to achieve efficient online inference using the proposed multi-scale DTM (MS-DTM) model built on a stochastic EM based method and incorporate temporal scales. The latent topics are inferred by using incremental multi-scale Gibbs

sampling and Dirichlet hyperparameters are determined by maximum likelihood estimation. Below is this process.

Formulating the Joint Probability

Our first step is to formulate the joint probability of documents and topics

$$P(D^{t,s}, Z^{t,s}|\alpha^{t,s}, \mu^{t,s}, E^{t,s}) = P(Z^{t,s}|\alpha^{t,s}) \times P(D^{t,s}|Z^{t,s}, \mu^{t,s}, E^{t,s}), \tag{5.4}$$

where $D^{t,s}$ is the set of documents at time t and scale s, $Z^{t,s}$ is a set of topics, and $E^{t,s}$ is the multiscale matrix containing multinomial distribution over topics, i.e. $E^{t,s} = [\psi^{t,s}_{k,m}]$, and $\mu^{t,s}$ is a vector of weighting factors corresponding to $E^{t,s}$.

Integrating out the Multinomials

Given this, we can take advantage of the Dirichlet-multinomial conjugacy and integrate out the multinomial distribution parameter, $\theta_{i,t}$ to rewrite the first term on the right hand side as follows:

$$P(Z^{t,s}|\alpha^{t,s}) = \prod_d \left(\frac{\Gamma(\sum_{z=1}^K \alpha_z^{t,s})}{\prod_{z=1}^K \Gamma(\alpha_z^{t,s})} \times \frac{\prod_z \Gamma(N_{d,z}^{t,s} + \alpha_z^{t,s})}{\Gamma(N_d^{t,s} + \sum_z \alpha_z^{t,s})} \right). \tag{5.5}$$

Here $N_{d,z}^{t,s}$ is the number of times a specific word is assigned to topic z from document d at time epoch t and scale s, and $N_d^{t,s}$ indicates the total number of times that a word has been assigned to each topic, i.e. $N_d^{t,s} = \sum_z N_{d,z}^{t,s}$.

In fact, the second term on the right hand side can also be rewritten by integrating out the multinomial distribution parameter, $\phi_{i,t}$:

$$\begin{aligned} P(D^{t,s}|Z^{t,s}, \mu^{t,s}, E^{t,s}) = \prod_z \Bigg(& \frac{\Gamma(\sum_{m=1}^S \mu_{z,m}^{t,s})}{\prod_w \Gamma(\sum_{m=1}^S \mu_{z,m}^{t,s} \psi_{z,m,w}^{t,s})} \\ & \times \frac{\prod_w \Gamma(N_{z,w}^{t,s} + \sum_{m=1}^S \mu_{z,m}^{t,s} \psi_{z,m,w}^{t,s})}{\Gamma(N_z^{t,s} + \sum_{m=1}^S \mu_{z,m}^{t,s})} \Bigg), \end{aligned} \tag{5.6}$$

where $N^{t,s}_{z,w}$ is the number of times a specific word w appears in topic z at time epoch t and epoch scale s and N^{t}_{z} indicates the total number of words appeared in topic z; i.e. $N^{t,s}_{z} = \sum_{w} N^{t,s}_{z,w}$.

Applying Multi-Scale Incremental Collapsed Gibbs Sampling

Next, I use collapsed Gibbs sampling to sequentially sample each topic variable, depending on the current state of all other variables to see that the new probability for the topic assignment, $P(z^{t,s}_{x} = j | D^{t,s}, Z^{t,s}_{\setminus x}, E^{t,s}, \mu^{t,s})$, is proportional to

$$\left(\frac{N^{t,s}_{j,w_j \setminus x} + \sum_{m=1}^{S} \mu^{t}_{j,m} \psi^{t,s}_{j,m,w_j}}{N^{t,s}_{k \setminus x} + \sum_{m=1}^{S} \mu^{t,s}_{j,m}} \right) \times \left(\frac{N^{t,s}_{d,j \setminus x} + \alpha^{t,s}_{j}}{N^{t,s}_{d \setminus x} + \sum_{j} \alpha^{t,s}_{j}} \right).$$

Here, index symbol x denotes the quadruple (t, s, d, n), which corresponds to the n^{th} word from document d at time epoch t at scale s, and $\setminus x$ means excluding the count of n^{th} word from document d at time epoch t and epoch scale s. The first ratio shows the probability of word w_j under topic j, using weighted multi-scale distribution from the past, and the second ratio shows the probability of topic j in document d at time t and scale s.

Updating Weighting Factors

Given the above, we find the weighting factors for the multi-scale parameters by directly maximizing the joint distribution in Equation 5.4, using the fix-point iteration method proposed in [64]. More specifically, by taking gradient of the log-likelihood of Equation 5.6 and setting it to 0, we obtain the following update rule for $\mu^{t,s}_{z,m}$:

$$\mu^{t,s}_{z,m} \leftarrow \frac{\mu^{t,s}_{z,m} \sum_{w} \psi^{t,s}_{z,m,w} H}{Q}, \tag{5.7}$$

where $\Psi()$ is the digamma function and we have

$$H = \Psi\left(N_{z,w}^{t,s} + \sum_m \mu_{z,m}^{t,s}\psi_{z,m,w}^{t,s}\right) - \Psi\left(\sum_m \mu_{z,m}^{t,s}\psi_{z,m,w}^{t,s}\right) \tag{5.8}$$

and

$$Q = \Psi\left(N_z^{t,s} + \sum_m \mu_{z,m}^{t,s}\right) - \Psi\left(\sum_m \mu_{z,m}^{t,s}\right). \tag{5.9}$$

Learning the Hyper-parameter, α

Finally, to complete the inference, the hyper-parameter α^t from the new data using maximum-likelihood estimation is learned. Again, by taking the gradient of the log-likelihood of Equation 5.5 and setting the gradient to 0, we obtain the following update rule for $\alpha_z^{t,s}$:

$$\alpha_z^{t,s} \leftarrow \frac{\alpha_z^{t,s}\sum_d\left(\Psi\left(N_{d,z}^{t,s} + \alpha_z^{t,s}\right) - \Psi\left(\alpha_z^{t,s}\right)\right)}{\sum_d\left(\Psi\left(N_d^{t,s} + \sum_z \alpha_z^{t,s}\right) - \Psi\left(\sum_z \alpha_z^{t,s}\right)\right)} \tag{5.10}$$

Summary

Algorithm 5 presents the pseudocode of the proposed iterative multi-scale inference process (IMS-DTM), which leverages the multi-scale update rules introduced above. Through iterative multi-scale Gibbs sampling, IMS-DTM is able to update both the weighting factors and the Dirichlet priors for the topics for all scales of all epochs with minimal overhead.

5.5 Experiments

In this section, we present experimental evaluations of the efficiency and effectiveness of the proposed IMS-DTM algorithm. All experiments were conducted in Matlab 2015b using an Intel Core i5-2400 machine with 8GB memory. In these experiments, we set the number, S, of scales to 4. The default parameters for Dirichlet prior for the topics for time epoch t is set to $\frac{\bar{D}^t \times 0.05}{K}$, where $\bar{D}^t$ is the average document length

at time epoch t.

We compare the proposed IMS-DTM (we refer to this as *multi-past multi-current DTM* (**MPMC**)) with multi-scale dynamic topic model proposed in [47]. Since it considers only multiple scales of the past, we refer to this approach as *multi-past single-current DTM* (**MPSC**). We also consider a baseline dynamic topic model with *single-past* [1] *and single-current scales* (**SPSC**) proposed in [11] and a *single-past multi-current* (**SPMC**) approach, implemented based on [18]. In the rest of this section, we refer to our approach as *multi-past multi-current* approach.

5.5.1 Evaluation Criteria

To evaluate accuracy of the models, we use the perplexity measure [14]. We define *epoch-level perplexity* to assess how well a probability model predicts the epoch. More formally, given a dynamic topic model $\mathbf{M} = \{w, k\}$, the epoch-level perplexity of an epoch $D^t = \{w_i\}$ can be computed as $PLX_{epoch}(t) = P(D^t|\mathbf{M}) = exp\left(-\frac{\sum_{i=1}^{|D^t|}\log p(w_i|\mathbf{M})}{\sum_{i=1}^{|D^t|}N_i}\right)$, where w_i is a word token in the i^{th} document in the epoch and N_i is the total number of words in that document. Similarly, we define *document-level perplexity* of a given epoch as follows: given a dynamic topic model $\mathbf{M} = \{w, k\}$, the document-level perplexity of an epoch $D^t = \{w_i\}$ is $PLX_{doc}(t) = avg_{d\in D^t}\left(P(d|\mathbf{M})\right) = avg_{d\in D^t}\left(exp\left(-\frac{\log p(w_i|\mathbf{M})}{N_i}\right)\right)$. A lower perplexity indicates a better model. Since the goal is often to characterize the epochs, we use *epoch-level perplexity* as the default accuracy measure.

5.5.2 Datasets

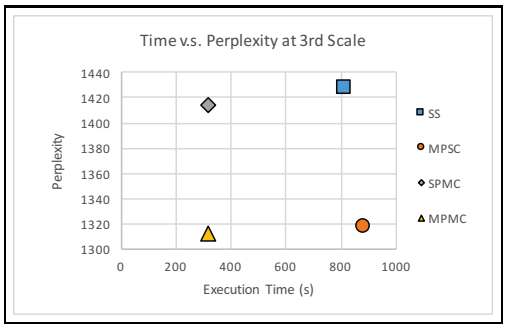
In this section, we consider various publicly available datasets, including text streams and numerical time series data.

[1] By default, when we refer to single past scale, we consider the smallest of the considered scales.

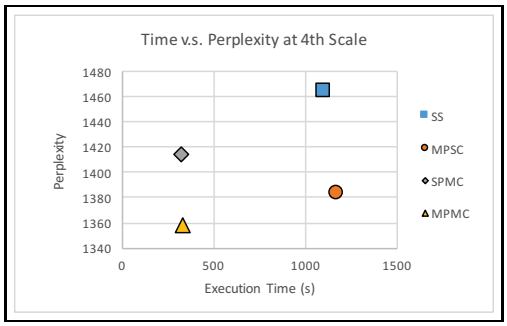
NIPS Data. We obtained the NIPS data, representing web-based scientific data streams, from UCI Machine Learning Repository Bag of Words Data Set. In this text collection, there are 1500 documents, the size of vocabulary set is 12419 and there are approximately 1.9 million words in the corpus. For this data, the default epoch length is 100 documents. The number, K, of latent topics is set to be 50, which is in line with [47] for easy comparison.

NYSK Data. NYSK (New York v. Strauss-Kahn) data set also comes from UCI Machine Learning Repository, representing web-based news data streams, is a collection of English news articles about the case relating to allegations of sexual assault against the former IMF director Dominique Strauss-Kahn in May, 2011. There are $\sim$10420 documents in the corpus. For this data, the default epoch length is 45 documents. Since the dataset is more focused (and crawled using just three specific hashtags), we expect the number of latent topics to be lower than NIPS data, and hence we set K to 20.

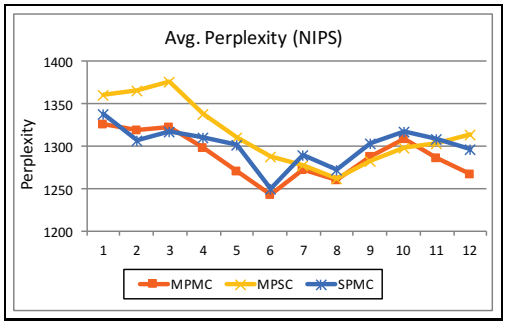
Apple Stock Data. Apple stock data, representing web-based financial data streams, is in the form of numerical time series from 1981 to 2015 from Quandl website. We use SAX [59] to discretize the numerical time series into sets of "documents" such that each "document" has one month worth of closing price data (i.e., 22 5-character SAX words, each corresponding to a moving average of 3 days). The data is split into year-length epochs. For this data set, which tracks stock market price movement, we set the target number, K, of topics to 5 – i.e., we are interested in a few major patterns in the data.



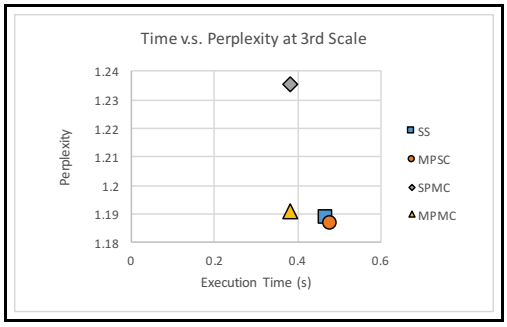
(a) Time v.s. perplexity at 3rd scale (NIPS dataset)



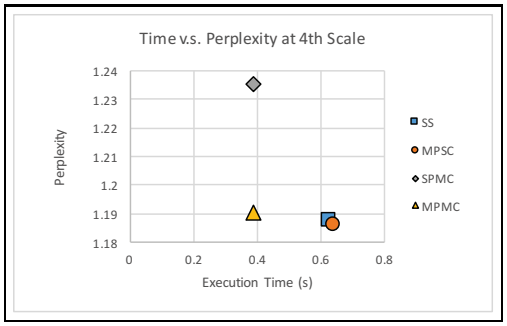
(b) Time v.s. perplexity at 4th scale (NIPS dataset)



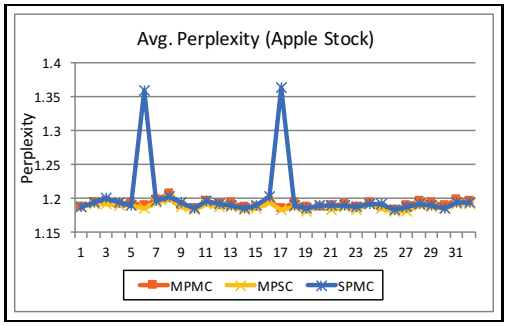
(c) Avg. perplexity over time (NIPS dataset)



(d) Time v.s. perplexity at 3rd scale (Apple Stock dataset)

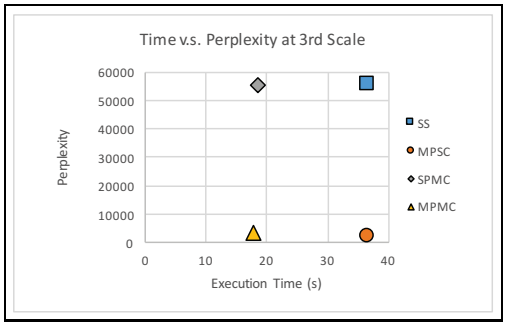


(e) Time v.s. perplexity at 4th scale (Apple Stock dataset)

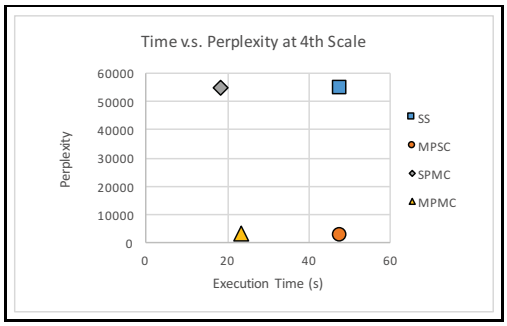


(f) Avg. perplexity over time (Apple Stock dataset)

Figure 5.3: Results for the NIPS Dataset, Apple Stock Dataset



(a) Time v.s. perplexity at 3rd scale (NYSK dataset)



(b) Time v.s. perplexity at 4th scale (NYSK dataset)

Figure 5.4: Results for the NYSK Dataset

5.5.3 Results

NIPS Dataset

Figure 5.3 (a,b,c) summarizes the results for the NIPS data set [2] . As we see in Figure 5.3(a,b), the multi-past multi-current scheme (MPMC) provides the lowest execution time, with best accuracy among all four approaches; whereas single scale (SS) results in the worst time/accuracy trade-off. The figure also shows that multi-past single-current (MPSC) provides better accuracy than single-past multi-current (SPMC) and the proposed multi-past multi-current scheme (MPMC) outperforms the MPSC accuracy. The figure also shows that, while gains in accuracy provided by MPSC comes with the penalty of significantly higher execution times than the other approaches, the accuracy gain of MPMC does not come with any execution time penalty. In fact, due to the incremental nature, the amount of time MPMC uses to compute the model for each scale is roughly equal to the time the single scale approach (SS) needs to compute the model for the smallest scale, even though MPMC is able to provide the best overall accuracy. In Figure 5.3(c), we plot the average perplexity

[2]Due to limitations of space, we only report results for the 3rd and 4th scales; others also produce similar results.

as function of time: the figure shows that at different epochs, MPSC or SPMC might be more advantageous than each other, while the proposed MPMC scheme performs almost always better than the best of MPSC or SPMC.

Apple Stock Dataset

Figure 5.3 (d, e, f) summarizes the results for the Apple stock data set. Once again, the multi-past multi-current scheme (MPMC) provides the best time/accuracy performance among all four approaches and the multi-scale approaches improve accuracy relative to single scale execution (SS): While, unlike the other two data sets, the MPSC provides a slightly better perplexity than MPMC; the proposed incremental multi-past multi-current scale approach (MPMC) provides the fastest execution time (Figure 5.3(d,e)), without incurring errors that the SPMC scheme introduces (Figure 5.3(f)) – i.e., even in this data set where (as we see in Figure 1) the model tends to get better as larger time periods are considered, once again MPMC provides the best of the both worlds in terms of efficiency and effectiveness.

NYSK Dataset

As we see in Figure 5.4 (a, b), the results for the NYSK data resembles the results for the NIPS data: the multi-past multi-current scheme (MPMC) provides the best time/accuracy performance among all four approaches and the multi-scale approaches improve accuracy relative to single scale execution (SS): the multi-scale incremental nature of MPMC ensures that the amount of time MPMC uses to compute the model for each scale is roughly equal to the time the single scale approach (SS) needs to compute the model for the smallest scale, even though MPMC provides as high accuracy as SPMC at the execution time cost point of the SS.

5.6 Conclusion

Data on the web reflect the evolution of the events and topics in the real world. A major limitation of most existing dynamic topic modeling approaches is that they assume a predetermined and fixed span (or epoch) of topics, whereas an evolving document corpus may contain topics of different temporal scales and, moreover, topics at one scale may impact the prediction of the topics at another scale. In this work, I developed a novel multi-scale dynamic topic model (MS-DTM), which considers both "past" and "now" in multiple scales. I further developed a multi-scale incremental Gibbs sampling mechanism for incremental multi-scale dynamic topic model (IMS-DTM) inference. The experiments show that the proposed IMS-DTM provides accuracy and efficiency gains for data streams with evolving topics of varying lengths.

As discussed in Section 1.3, despite the factorization models described in Chapter 3 and Chapter 4 and probablistic generative models described in this chapter, the third model family which requires most domain knowledge is the black box models, usually involves a simulator and a large simulation input parameter space to explore. We aim to select from a large space of a potential simulations, a subset to be executed and maximize the ease-of-interpretation (in addition to its descriptive power). In Chapter 6 and Chapter 7, I will describe two algorithms that sequentially look for new simulation instances to execute in both static and dynamic simulation systems.

Algorithm 5 IMS-DTM Algorithm

Input:

Streaming corpus D; Number of topics K; Number of time epochs $\mathcal{T}$; Number of multi scale epochs S; Number of Gibbs sampling iterations $iter$; Hyperparameter update frequency $iterUpdate$;

Output:

The incremental multi-scale dynamic topic model $\mathcal{M}$;

for t = 1 **to** $\mathcal{T}$ **do**
  Initialize Dirichlet prior for the topics at time epoch t as $\alpha^t = \frac{\bar{D}^t \times 0.05}{K}$, where $\bar{D}^t$ is the average document length at time epoch t
  Initialize count variables, Multinomial distribution θ, ϕ and topic assignments.
  Initialize multi-scale parameter μ as $\mu = \frac{1}{S}$
  Initialize Dirichlet prior for the words at time epoch t as $\beta_z^t = \mu_z^t \times E_z^t$
  for i =1 **to** $iter$ **do**
    Update $\theta^{t,1}$, $\phi^{t,1}$, $Z^{t,1}$ using Collapsed GibbsSampling($D^{t,1}$, $\alpha^{t,1}$, $\beta^{t,1}$)
    for s = 2 **to** S **do**
      Update $\theta^{t,s}$, $\phi^{t,s}$, $Z^{t,s}$ using Incremental GibbsSampling($D^{t,s}$, $\alpha^{t,s}$, $\beta^{t,s}$)
      if (i mod $iterUpdate$) == 0 **then**
        Update $\alpha^{t,s}$ using Equation 5.10
        Update $\mu^{t,s}$ using Equation 5.7
        Set new word Dirichlet prior as $\beta = \mu \times E$
      end if
    end for
  end for
end for

Chapter 6

COMPLICACY-GUIDED PARAMETER SPACE SAMPLING FOR KNOWLEDGE DISCOVERY WITH LIMITED SIMULATION BUDGETS

6.1 Introduction

As mentioned in Section 1, knowledge discovery and decision making through data- and model-driven computer simulation ensembles are increasingly critical in many application domains. However, these simulation ensembles are expensive to obtain. Consequently, given a relatively small simulation budget, one needs to identify a sparse ensemble that includes the most informative simulations to help the effective exploration of the space of input parameters cost-effective.

6.2 Problem Formulation

In this section, I formulate the budgeted simulation sampling problem we solve in this paper.

6.2.1 Simulation Space and Simulation Ensemble

Let us be given a complex system, S, with N input parameters, such that the i^{th} input parameter can take I_i distinct values. For simplicity of the discussion, let us further assume that for each input parameter combination $\langle v_1, \ldots v_N \rangle$, the complex system S generates a single value $S(v_1, \ldots, v_n)$.

Let Y be the set of all simulations of the system S one can execute and the corresponding results; i.e., $Y = \{y_i = \langle \langle v_{i,1}, \ldots, v_{i,N} \rangle, S(v_{i,1}, \ldots, v_{i,N}) \rangle \parallel 1 \leq i \leq I_1 \times I_2 \times \ldots \times I_N\}$. Note that Y can be encoded as a tensor $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$, where

for all $y_i \in Y$, $\boldsymbol{\mathcal{Y}}(v_{i,1}, \ldots, v_{i,N}) = S(v_{i,1}, \ldots, v_{i,n})$.

Ideally, to study the system, S, we would construct a complete tensor $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$. However, it is easy to see that this process would be prohibitively costly. Firstly, this would require us to execute $I_1 \times \ldots \times I_N$ simulations, which can be computationally overwhelming. Even if this many simulations can be obtained, the analysis of the resulting dense tensor may be prohibitively expensive. Instead, given a budget $B \ll I_1 \times \ldots \times I_N$ of simulations, we identify and execute a set, $X = \{x_j = \langle\langle v_{j,1}, \ldots, v_{j,N}\rangle, S(v_{j,1}, \ldots, v_{j,N})\rangle \parallel 1 \leq j \leq B\}$ of B simulations, leading to a simulation ensemble tensor, $\boldsymbol{\mathcal{X}}$.

6.2.2 Ensemble Construction Criteria

Let $D()$ be a decision function [1] , which takes a complex system as input and outputs a decision. Naturally, the simulation instances included in the simulation ensemble should be selected in such a way that the sparse tensor, $\boldsymbol{\mathcal{X}}$, would lead to similar decisions as if the complete tensor, $\boldsymbol{\mathcal{Y}}$, was available to the decision maker: $D(\boldsymbol{\mathcal{X}}) \sim D(\boldsymbol{\mathcal{Y}})$. However, given that in practice we neither have access to the complete tensor, $\boldsymbol{\mathcal{Y}}$, nor the decision function, $D()$, we need to replace this criterion with a similar, but less strict condition: assuming that the decision function, $D()$, will rely on an intermediate data-driven model, M_D, constructed from the available data, we can replace the above criterion with $M_D(\boldsymbol{\mathcal{X}}) \sim M_D(\boldsymbol{\mathcal{Y}})$.

This revised criterion removes the need to access the decision function, D, directly to assess the simulation ensemble and, instead, needs only to consider the data driven model used for decision making. However, the criterion still needs access to the complete system, $\boldsymbol{\mathcal{Y}}$, which in practice is not available [2] . Therefore, we further

[1] The decision function, $D()$, might represent the user's domain expertise, subjective preferences, rules and regulations

[2] In certain situations, partial information about the system may be available, either through

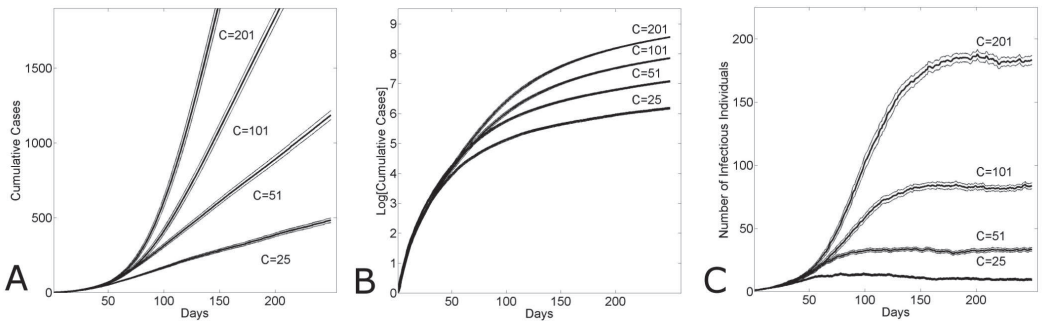


Figure 6.1: Epidemics often show different behaviors (exponential, sub-exponential, and linear) over time

rephrase the above criterion and state it in terms of the properties of $M_D(\boldsymbol{\mathcal{X}})$:

- *Fit/Error:* It is important that the model, $M_D(\boldsymbol{\mathcal{X}})$, provides a good explanation of the ensemble, $\boldsymbol{\mathcal{X}}$. In other words, given a fit/error function $fit(\cdot,\cdot)$ (or $error(\cdot,\cdot)$) measuring how well a given model explains given data, we would desire that $fit(M_D(\boldsymbol{\mathcal{X}}), \boldsymbol{\mathcal{X}})$ is high ($error(M_D(\boldsymbol{\mathcal{X}}), \boldsymbol{\mathcal{X}})$ is low).

- *Coverage:* Optimizing only for a good fit may not be a good strategy, as it might be possible to select a simulation ensemble with an easy to fit model (for example, by selecting simulation instances at the flat region in Figure 6.1), but not completely representing the complete system, $\boldsymbol{\mathcal{Y}}$. We, therefore, extend the above criteria as follows: Let $\boldsymbol{\mathcal{X}} \in (\mathbb{R} \cup \perp)^{I_1 \times I_2 \times \ldots \times I_N}$ be a simulation ensemble constructed with B simulation instances and $coverage(\boldsymbol{\mathcal{X}})$ be a function denoting the ratio of the simulation space $I_1 \times I_2 \times \ldots \times I_N$ covered by the simulations ensemble. We desire that $coverage(\boldsymbol{\mathcal{X}})$ is high ($\simeq 1$).

- *Simplicity/Complicacy:* While we rarely have access to the complex (and often implicit) decision function, we can have certain additional expectations about the models that support these decisions. One of these expectations is that the

existing observations or through validation experiments one can execute. Without loss of generality, we ignore these partial knowledge scenarios in this paper.

model, M_D, should be *simple* to interpret and, as discussed in the introduction, its form should match the underlying process (remember from Figure 1.5 that even though the sinusoidal curve has a good fit to the simulation results, it may not be a good model in applications where we only expect polynomial complicacy). Therefore, given a complicacy function, *complicacy*(), we desire that the simulation instances included in $\boldsymbol{\mathcal{X}}$ lead to a model with a low complicacy; i.e., we expect that *complicacy*($M_D(\boldsymbol{\mathcal{X}})$) is low.

6.2.3 Problem Statement

Let us be given a complex system, S, with N input parameters, such that the i^{th} input parameter can take I_i distinct values. Let us also be given a simulation budget of $B \ll I_2 \times \ldots \times I_N$ simulations to execute. Our goal is to construct a simulation ensemble, $\boldsymbol{\mathcal{X}}$, that leads to a data-driven model, $M_D(\boldsymbol{\mathcal{X}})$, that (a) provides a good degree explanation, (b) has a large coverage, and (c) has low complicacy. We then discuss our solution to this problem.

6.3 Complicacy-Guided Space Sampling

The *complicacy-guided parameter space sampling (CPSS)* algorithm (depicted in pseudo-code form in Algorithm 6) relies on a top-down iterative sampling method, where the budgeted simulation samples are incrementally spent to maximize the overall model quality as previously described. CPSS starts by sampling from the complete parameter space as a whole and, *as needed*, recursively splits the parameter space into smaller regions to be described by their own model. This ensures that the selected simulation instances *cover* the entire parameter space. Moreover, to ensure that the resulting ensemble has also high *fit*, and low *complicacy*, CPSS relies on innovative techniques, including 1) a model dictionary, with models of varying complicacies; 2)

a flexible mechanism to quantify model complicacy; 3) an effective strategy to decide which partition of the parameter space to assign additional simulation instances; and 4) a strategy to decide whether to further partition a given region. We discuss each in detail in the following subsections.

6.3.1 Model Dictionary

Unlike existing works, CPSS takes as input a model dictionary $\mathcal{M} = \{M_1, M_2, ..., M_k\}$ that consists of a diverse set of parametric models of different form (linear, polynomial, logarithmic, exponential, sinusoidal). Given this dictionary and given a set, $X = \{x_i = \langle\langle v_{i,1}, \ldots, v_{i,N}\rangle, S(v_{i,1}, \ldots, v_{i,N})\rangle \parallel 1 \leq i \leq s\}$, of simulations, CPSS uses a regression-based algorithm to identify a parameter instantiation $\Pi(M_i, \boldsymbol{\mathcal{X}})$, for each model M_i in the dictionary, that provides the best fit for that model.

In the rest of the paper, when we refer to a model M_i and its fit and complicacy, we will implicitly refer to the fit and complicacy of the model instantiated with $\Pi(M_i, \boldsymbol{\mathcal{X}})$.

6.3.2 Model Complicacy

As discussed in the Introduction, different models within the dictionary may present different degrees of complicacy to the users. Intuitively, to support effective decision making, all things being equal, we prefer models that are (a) simpler and (b) more suitable to the application. In this paper, we propose a complicacy model that relies upon the following rules to capture application specific complicacy demands. The main intuition behinds these rules are based on the degree of polynomials, and the number of roots (solutions) is based on the highest degree of its monomials.

- The complicacy of a polynomial is determined as:

$$C(a) = 0;\;\; C(x) = 1;\;\; C(x^a) = \begin{cases} abs(a) & \text{if } a \geq 1 \\ 1/abs(a) & \text{if } a < 1 \end{cases}$$

- The complicacy of a non-polynomial is application specific: $C(sin(x)) = W_s$; $C(e^x) = W_e$; $C(\log x) = W_l$, etc.

- For any combination of the aforementioned forms, we derive the complicacy measure using the following rules:

$$\begin{aligned} C(f(g(x))) &= C(f(x)) \times C(g(x)) \\ C(f(x) + g(x)) &= \max(C(f(x)), C(g(x))) \\ C(f(x) \times g(x)) &= C(f(x)) + C(g(x)). \end{aligned}$$

The above rules consider, not only the *shape of the surface* defined by the function, but also the *specific form the description of the function* takes. Note that the specific form to describe the function plays an important role in our complicacy measure since the raw function may take several steps before converted into its final simpler form, but this requires one to spend more effort and thus needs to be taken into consideration under the complicacy measure.

Example We have $C(a \times e^{b \times log(c \times x)}) \rightarrow W_e \times W_l$, even though $a \times e^{b \times log(c \times x)} = a \times b \times c \times x$ and $C(a \times b \times c \times x) = 1$. This is because (a) when presented to the user $e^{b \times log(c \times x)}$ is likely to be harder to interpret than x (even though they are mathematically equivalent) and (b) while the function, $a \times e^{b \times log(c \times x)}$ has apparently three parameters that need to be considered, in its simpler form, $a \times b \times c \times x$, it has one parameter, $d = a \times b \times c$. Therefore, the complicacy function, $C()$, makes a distinction between these two forms of the model. $\diamond$

6.3.3 Complicacy Guided Model Penalty

In order to be able to create ensembles that result in models that both have good fit to the data and are also simple, we need to combine complicacy and fitness. There are several existing measures, such as Akaike information criterion (AIC [5]) and normalized R^2 [65]. Let M be a model learned from data X. Let $\pi(M)$ be the number of parameters to be inferred in the model M and L be the maximum value of the likelihood function for the model; i.e. $L(M, X) = P(X|\theta, M)$, where θ are the parameter values that maximize the likelihood function. The AIC penalty is defined as

$$AIC(M, X) = 2\pi(M) - 2\ln(L(M, X)).$$

Note that, the AIC aims to find a balance between the number of parameters to be inferred (which is a measure of the model complicacy) and the model's ability to explain the data. In particular, AIC penalizes models with large numbers of parameters.

Complicacy Guided Akaike Information Criterion (C-AIC)

Note that the model complicacy (i.e., the number of model parameters) AIC relies on is rather limited. We therefore introduce a *Complicacy Guided Akaike information Criterion (C-AIC)* measure, which utilizes the complicacy rules described in Section 6.3.2 to associate a complicacy penalty for each model in the model dictionary:

$$C - AIC(M, X) = 2C(M) - 2\ln(L(M, X)).$$

Intuitively, C-AIC replaces the term $\pi(M)$ (the number of free parameters) with the complicacy measure, $C(M)$, defined in Section 6.3.2 (which considers both the number of free parameters, but also the interpretability of the model).

Coverage-Weighted Complicacy Guided Akaike Information Criterion (C2-AIC)

While C-AIC is capable of capturing both the degree of explanation and degree of simplicity of a model for a given set, X, of simulation instances, when we consider different regions of the parameter space, we may not be able to directly compare their C-AIC values. In particular, if two ensembles X_1 and X_2 have two different degrees of coverage of the parameter space, the impact of their complicacies are limited within the scopes of the parameter spaces they cover. Therefore, we need to weigh the models in terms of the coverages of the ensemble data sets that support them. In other words, if X is a simulation ensemble with coverage, $coverage(X)$, and M is a model in the model dictionary, we define the corresponding *Coverage-Weighted Complicacy Guided Akaike information criterion (C2-AIC)* penalty as follows:

$$C2 - AIC(M, X) = 2 \times \underbrace{C(M)}_{complic.} \times \underbrace{coverage(X)}_{size} - 2 \underbrace{\ln(L(M, X))}_{fit}$$

Note that C2-AIC not only takes into account the complicacy and the fitness of the model, but also considers the coverage of the ensemble. Intuitively, C2-AIC would limit complicate models to only small portions of the input parameter space and would seek simpler models for large portions of the simulation space.

Other Measures of Model Assessment

AIC is not the only way to combine complicacy and fitness. Another commonly used measure is the *normalized* R^2 (NR), often used in regression analysis:

$$NR(M, X) = \underbrace{\frac{|X| - 1}{|X| - \pi(M)}}_{term\ 1} \times \underbrace{\frac{\sum_{x_j \in X} (S(x_j) - M(x_j))^2}{\sum_{x_j \in X} (S(x_j) - \bar{S})^2}}_{term\ 2},$$

where $\bar{S}$ is the average of the observed simulation outcomes over X. Note that, here, the first term represents the model complicacy in terms of the number parameters. We

can, therefore, revise NR by replacing the first term with a more general complicacy term, $C(M)$. This term can be multiplied with $coverage(X)$ to obtain the coverage-weighted and complicacy-guided version of the measure.

6.3.4 Partition Selection with Rank Stability

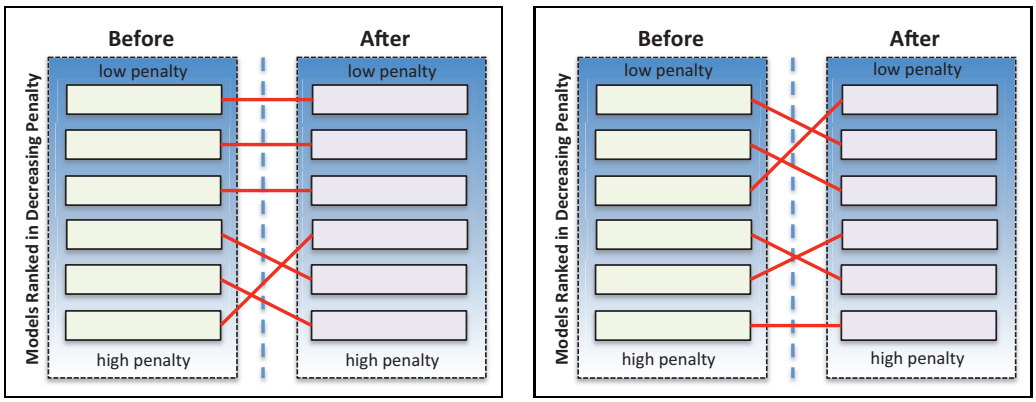
As we discussed in Section 1, a complex system may show completely different patterns in different regions of the parameter space and, in order to ensure that the ensemble is created in a way that properly covers and describes the underlying phenomenon, CPSS starts by sampling from the complete parameter space as a whole and, *as needed*, splits the parameter space into finer regions to be described by its own model. If all current partitions match the target quality requirement or the simulation budget has been consumed in its entirety, the process ends.

Conventional Approaches

In this paper, we consider several strategies: (a) *random strategy* which selects a partition at random; (b) *fit-based strategy* which selects the partition with the worst fit; and (c) *complicacy-based strategy* which selects the partition with the worst model complicacy.

Rank Stability

Unfortunately, in most practical applications, it is hard to declare what is an appropriate fit or complicacy target as this might be both application and region dependent. Instead, we need a measure which can, not only take different penalty functions as input, but also simplify the tuning of the partition selection and stopping criteria. In this section, we propose a novel (d) *rank-stability based strategy* which selects the partition with the worst rank-stability (under the appropriate penalty function) for



(a) high rank stability (b) low rank stability

Figure 6.2: When an ensemble has high rank stability, additional simulations do not change the order of the models with low penalty; see (a) vs. (b)

further investigation.

Let us be given a complex system, S, with N input parameters, such that the i^{th} input parameter can take I_i distinct values. Let, as before, X be the set of simulations that have been selected to be executed so far. Let $\mathcal{P} = \{P_1, \ldots, P_p\}$ be a set of non-overlapping partitions of the input parameter space, such that $\cup_{P_i \in \mathcal{P}} P_i = I_1 \times \ldots \times I_N$ (to satisfy the *coverage criteria*), and let X_i denote the (non-empty) subset of X that falls in partition P_i. Let us further assume that, given a model dictionary, $\mathcal{M} = \{M_1, M_2, ..., M_k\}$, we associate a penalty, $penalty_{i,j}$, to each model in $M_j \in \mathcal{M}$ in partition P_i(for example using C2-AIC the penalty criterion discussed in Section 6.3.3).

Let us assume that the partition P_i is selected for further investigation and we extend the simulation set, X_i, with new simulation instances, ΔX_i. Let us denote the extended simulation set as $X'_i = X_i \cup \Delta X_i$ and associate a revised penalty, $penalty'_{i,j}$, to model $M_j \in \mathcal{M}$. Given these, we define the rank stability, $RS(P_i, X_i, \Delta X_i)$, of $P_i \in \mathcal{P}$ for simulation set, X_i, and new instances, ΔX_i, as

$$\sum_{\substack{M_j \in \mathcal{M} \\ \text{in decreasing order of penalty}_{i,j}}} \frac{e^{(min\_penalty'_i - penalty'_{i,j})/2}}{log_2(j+1)},$$

where $min\_penalty_i$ is the minimum penalty computed for all models in $\mathcal{M}$. Intuitively, $RS()$ is analogous to normalized discounted cumulative gain (NDCG [48]) of the old model ranking as a function of the new model ranking, it quantifies if the additional simulation samples for the given partition result in a major shift in the ranking of the models in the model dictionary: note that the numerator re-normalizes the model penalties, such that the models with lower penalties are given higher weights in measuring rank stability. Intuitively, if the rank stability is low, it means that the current partition is hard to describe with X_i or X_i'; in contrast, if the rank stability is high, it means that additional simulations do not impact the ranks of the models in the dictionary, thus, additional simulation samples for the given partition may not be necessary.

6.3.5 Partition Split Strategy with Look-Ahead

Once a partition, P_i is picked, CPSS assigns a set, ΔX_i, of new simulations to the partition. Once these simulations have been executed, models in the model dictionary, $\mathcal{M}$, are re-ranked based on the revised set, $X_i' = X_i \cup \Delta X_i$, of simulations for the given partition and a new rank stability measure, $rs_i = RS(P_i, X_i, \Delta X_i)$ is computed for the partition. If rs less than a threshold $rs_\perp$, then the partition is not rank stable and needs to be further partitioned.

If $rs \geq rs_\perp$, however, P_i may or may not need further partitioning. To decide, CPSS relies on a look-ahead mechanism: (a) P_i is *virtually* split into sub-partitions and (b) a model ranking is obtained for each sub-partition based on the corresponding simulation instances; (c) next, for each sub-partition, its rank stability is computed relative to P_i and checked whether this rank stability is below $rs_\perp$: if none of the partitions fail the rank stability test, then the order of the models of sub-partitions are aligned with the order of the models for P_i, therefore P_i does not need to be

further split; otherwise, P_i is composed of heterogeneous regions that needs to be further split into smaller partitions.

6.3.6 Result

When the process ends (either because the budget has been consumed or rank-stability is achieved for all partitions), we have a simulation ensemble and a ranked list of instantiated models for each partition. One advantage of dictionary-based approach is that the user can be provided with several top models for each partition as candidates.

6.4 Experiments

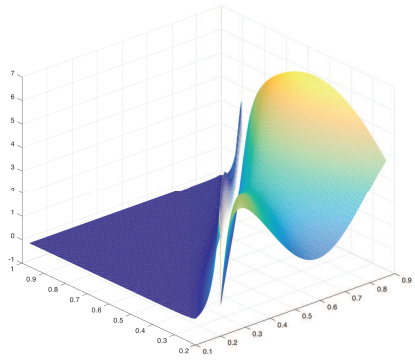
In this section, we present experimental evaluations of CPSS under various settings and simulation scenarios.

Continuous Complex System

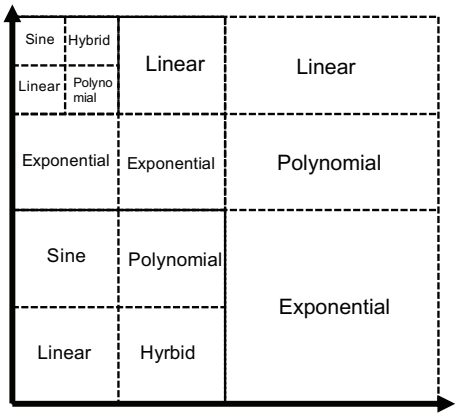
Firstly, we consider the 2-parameter complex system in Figure 6.3(a), where $f(x_1, x_2) = \Lambda(x_1, x_2)\big(e \times sin(8x_1) + 5sin(3x_2)\big)$ when $x_1 < x_2$ and $f(x_1, x_2) = (1 - \Lambda(x_1, x_2))\big(e \times sin(8x_1) + 5sin(3x_2)\big)$ when $x_1 \geq x_2$. Furthermore, $\Lambda(x_1, x_2)$ is an exponential smoothing term, which smooths the transition, where $x_1 \sim x_2$.

Piecewise Complex System

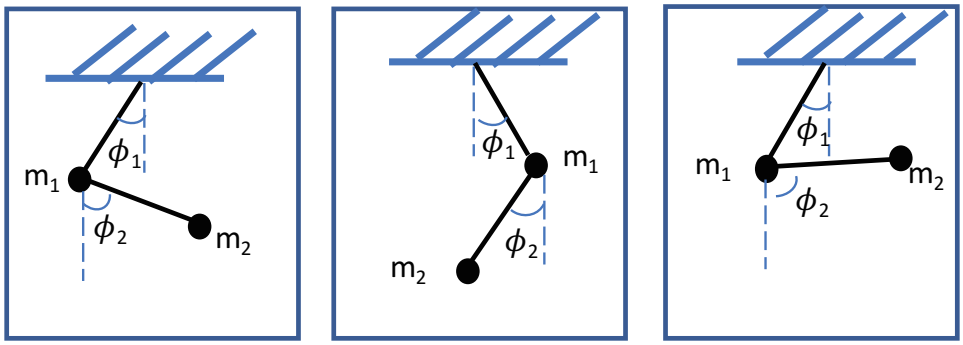
Secondly, we consider a *piecewise complex system*, in Figure 7.2(b): the parameter space is split into 14 regions of different sizes, shapes, and complicacies (linear, polynomial, exponential, or hybrid).



(a) continuous complex



(b) piecewise complex

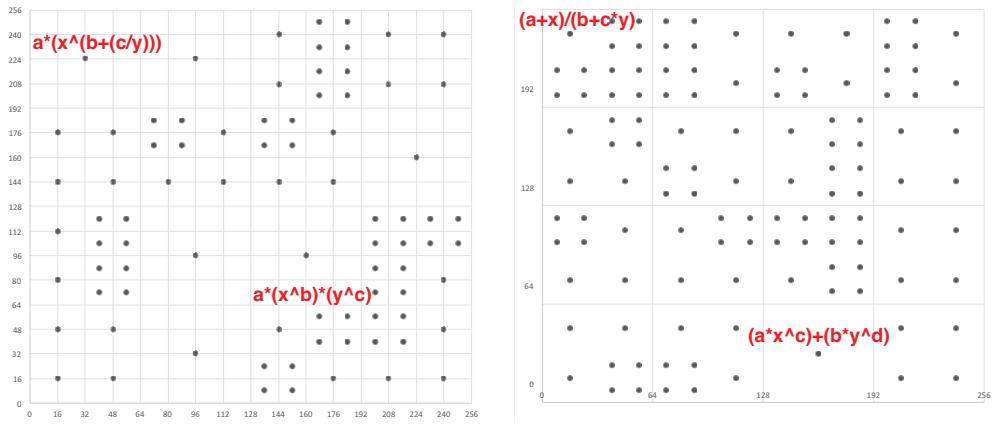


(c) double-pendulum system

Figure 6.3: Continuous and Piecewise Complex Scenarios

Pendulum System

We consider a pendulum system, in Figure 7.2(c), with three variables: the gravity, g, the initial angle, θ, and weight, m of the pendulum. We select a random initialization of these variables as the ground truth and, for each simulation, we output the Euclidean distance from this ground truth.



(a) fit-based sampling (b) complicacy-guided sampling

Figure 6.4: Simulation samples based on different criteria for the system in Figure 7.2(a); each dot corresponds to a partition created by the algorithm. (a) fit-based sampling picks models with large complicacies; whereas (b) complicacy-guided sampling results in models with less complicacy

Epidemic Simulations

Finally, we use epidemic simulations created using STEM [1]: we consider a simulation scenario, where an epidemic is occurring in the US. We use SEIR model with three input parameters (transmission, recovery, and mortality rates), each parameter is a continuous real number ranging from 0 to 1, and we consider 40 distinct values for each parameter, i.e., they all vary from 0.025 to 1 with increment of 0.025. For the simulation outcome, we focus on the number of deaths over time; We select a random initialization of these variables as the ground truth and, for each simulation, we output the Euclidean distance from the ground truth.

6.4.1 Simulation Sampling Strategies

In this section, we experiment with different strategies for creating simulation ensembles: we consider (a) normalized R^2 (NR, as known as Adjusted R^2) as a widely used state-of-art baseline [65]; and 3 variants of the baseline, which incorporate complicacy and look-ahead mechanism: (b) complicacy-guided NR (CNR); (c) NR

with look-ahead; (d) CNR with look-ahead. We also consider three variants of the proposed CPSS algorithm, (e) AIC; (f) complicacy-guided AIC (C-AIC); and (g) complicacy-guided and coverage-weighted AIC (C2-AIC). Note that the NR based measures do not leverage rank stability; in contrast, for AIC based measures, we apply rank-stability and look-ahead by default. Unless stated, we use the random partition selection strategy as default.

6.4.2 Experiment Settings

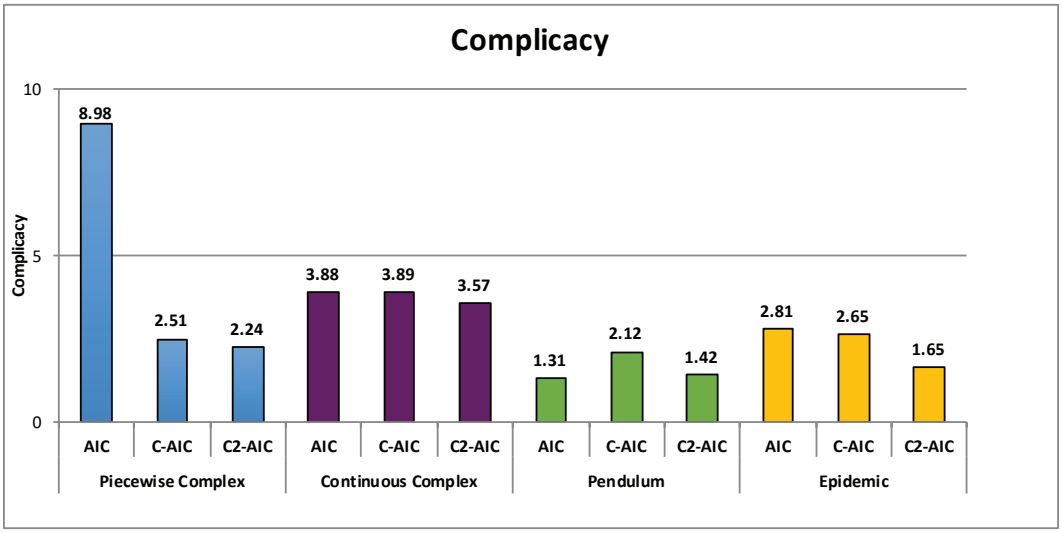
In the experiments presented in this section, we set the default value for total simulation budget B_t to 900 and default partition threshold $rs_{\perp}$ to 0.97. We also set application specific complicacy W_e, W_l, and W_s to 4. Each time a partition is selected for further study, we schedule 30 new simulations for that partition.

Given an ensemble, in order to to identify a best-fitting parameter instantiation for each model in the dictionary, we rely on MatLab functions `fit`() and `fitnlm`().

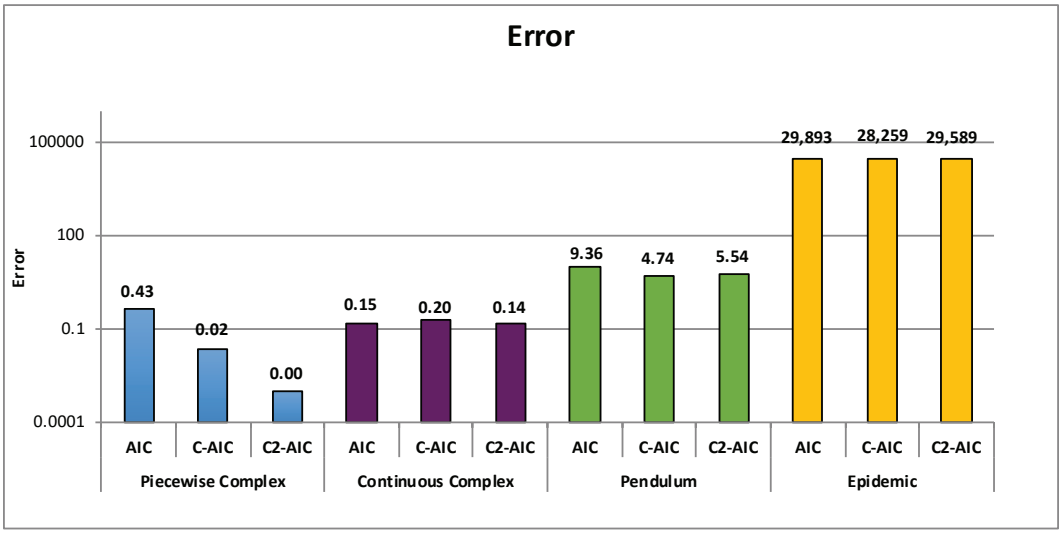
6.4.3 Evaluation Measures

As we see in Figure 6.4, different simulation ensemble creation strategies sample the parameter space differently, potentially leading to different degrees of fit and complicacy. Thus, we use the following measures of fit and complicacy:

- *Error:* For each partition, we (a) consider an average error, measuring the absolute difference between the ground truth and the predicted output of the system, (b) multiply this average error with the volume of the partition, and (c) finally sum up all the volume-weighted errors to measure the complete `error` for the whole parameter space.



(a) `complicacy` results



(b) `error` results

Figure 6.5: (a) Complicacy and (b) error results for pure AIC, complicacy-guided AIC (C-AIC), and coverage-weighted, complicacy-guided AIC (C2-AIC) measures for different data sets

Table 6.1: Experiment Results

Algorithm	Comp.	P-Comp.	Error	Num. of regions
NR	13.3	995	0.14	75
CNR	5.97	602	0.15	101
NR w/LA	5.4	562	0.19	104
CNR w/ LA	3.8	470	0.16	124
AIC	3.92	481	0.15	124
C-AIC	3.82	481	0.20	124
C2-AIC	3.49	439	0.14	124

- *Complicacy:* We also assess the complicacy of the resulting models. For this purpose, we (a) first compute the complicacy for each partition, (b) multiply this value with the volume of the partition, and (c) sum these up to obtain the complete `complicacy` for the entire parameter space. In addition, in order to ensure that the sampling strategies do not return a very large number of low-complicacy models, we also use a partition weighted complicacy, `p_complicacy` measure where we multiply the complicacy with the number of partitions the sampling strategy generates. A low `p_complicacy` indicates a small number of partitions with low complicacies.

Note that all strategies considered in this paper are top-down and, thus, fully cover the parameter space.

6.4.4 Discussion of the Results

Overview Results of Different Data Sets

Figure 6.5 provides an overview of the `complicacy` and `error` results for the three variants of proposed CPSS algorithm: AIC, complicacy-guided AIC (C-AIC), and coverage-weighted, complicacy-guided AIC (C2-AIC) measures, under default configuration for different data.

As we see in Figure 6.5, the general trend, except for the pendulum data set, is that

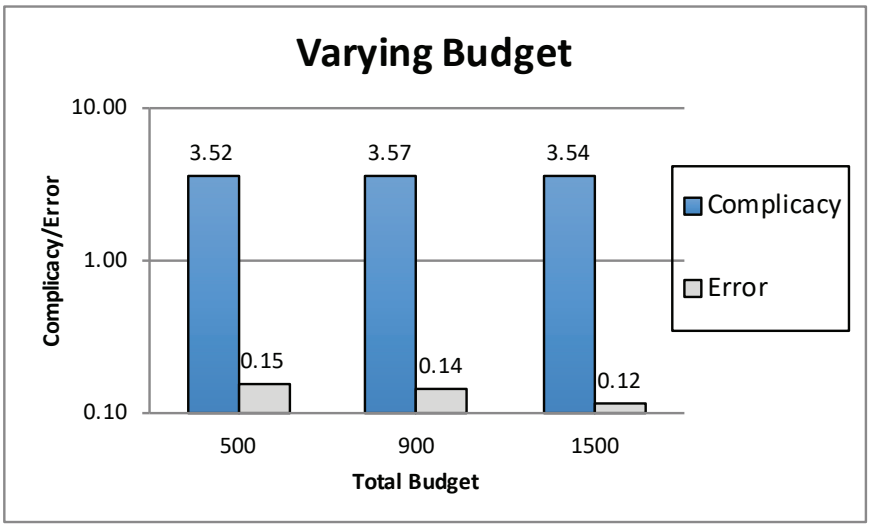


Figure 6.6: Varying budget (continuous complex data, C2-AIC measure, default parameters)

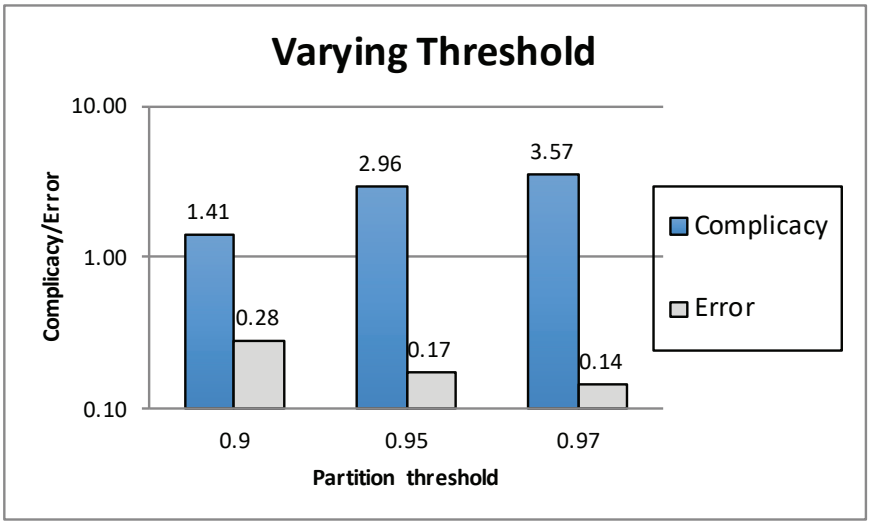


Figure 6.7: Varying rank stability threshold (continuous complex data, C2-AIC measure, default parameters)

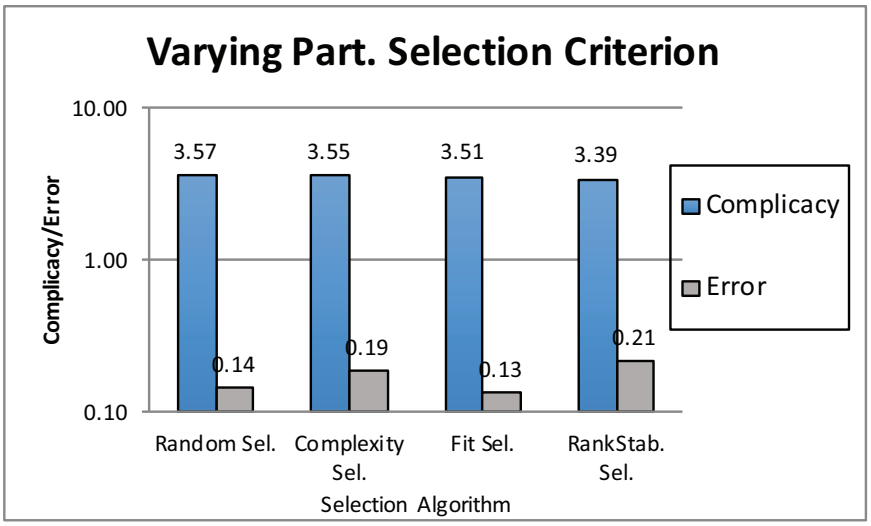


Figure 6.8: Varying partition selection criterion (continuous complex data, C2-AIC measure, default parameters)

as we leverage *complicacy-guidance* and *coverage-weighting* strategies, the complicacy of the model returned by the simulation ensemble drops; moreover, C2-AIC (which leverages both of these strategies) leads to the best overall complicacy among the three alternatives. Figure 6.5(b) studies the degree of fit for the models returned by the three strategies. As we see here, the *complicacy-guidance* and *coverage-weighting* strategies, not only lead to lower complicacies, but also lower error. Note that, the reason why, for the pendulum data set, *complicacy-guidance* and *coverage-weighting* strategies appear to lower the error, rather than focusing on complicacy, is that for that data set, model complicacies are already low (close to linear; i.e., ~ 1). Therefore, C2-AIC focuses on correcting the error, which is very poor for the AIC strategy.

Overall, *complicacy-guidance* and *coverage-weighting* strategies, together, help improve ensemble quality in terms of fitness, simplicity.

Results for Different Penalty Measures and Split-strategies

Table 6.1 shows the `complicacy`, `p_complicacy`, and `error` results for different penalty measures and split strategies: (a) normalized R^2 (NR) as the baseline; and

three variants of the baseline: (b) complicacy-guided NR (CNR); (c) NR with look-ahead; (d) CNR with look-ahead; the proposed CPSS using: (e) AIC; (f) complicacy-guided AIC (C-AIC); and (g) complicacy-guided and coverage-weighted AIC (C2-AIC). As we see, the baseline (pure normalized R^2 (NR) based approach) leads to a very high model complicacy. Combining NR with complicacy-guidance and look-ahead based split (CNR w/LA) significantly improves the model complicacy. We see that the last three approaches (AIC, C-AIC, and C2-AIC), where we apply rank-stability and look-ahead by default, match the complicacy performance of the CNR w/LA. As expected, we obtain the best complicacy outcomes using C2-AIC, which leverages complicacy-guiding and coverage-weights, along with rank-stability and look-ahead for split decisions. The 99% confidence interval on mean model complicacy for AIC, C-AIC, and C2-AIC are [3.73, 4.11], [3.65, 3.99] and [3.36, 3.63] respectively, this confirms that the complicacy results are statistically significant.

Also in terms of the partition-weighted complicacy (`p_complicacy`), the complicacy-guiding and coverage-weights strategies with rank-stability and look-ahead, lead to better models, despite the fact that the number of resulting partitions is generally higher than the pure NR strategy.

Results for Different Parameter Settings

Figure 6.6 studies the impact of the total simulation budget B that can be allocated. We select two alternatives, namely, 500 and 1500. As we see in the figure, even in cases where increasing the budget does not necessarily help reduce complicacy (because a low complicacy model is already found), the C2-AIC strategy leverages the additional samples to push down the error and improve fit.

Figure 6.7 studies the impact of the rank stability threshold $rs_{\perp}$. We vary the threshold to be 0.9 and 0.95. The threshold can be used to control the trade-off

between complicacy and fit. When $rs_{\perp}$ is low, the C2-AIC strategy stabilizes with models that have low complicacies; however, as we increase the threshold, it becomes harder to reach a complicacy-based stability and, thus, the strategy leads to smaller partitions, which tend to have lower errors.

Figure 6.8 studies the impact of the different strategies on the degrees of fitness of the resulting models. As we see here, the pure NR strategy leads to a low error rate (0.14); however, it also leads to low (0.43) error fairness. It is interesting to see that, this low error rate, 0.14, is also matched by the C2-AIC strategy and that C2-AIC also provides a significantly higher degree of fairness (0.68), indicating that, the proposed strategies are also effective in terms of the degrees of fits of the resulting models.

6.5 Conclusion

In this work, I developed a knowledge discovery framework to create simulation ensembles to support decision making for complex systems. Noting that simulation ensembles are often extremely sparse due to the size of the potential simulation parameter space, we proposed a top-down complicacy-guided parameter space sampling (CPSS) scheme to help obtain simple, yet accurate models. As the experiments showed, CPSS employs fair and effective strategies to decide (a) which partition to pick for additional simulations, (b) whether to further partition a given region of the simulation space, and (c) whether to stop adding instances to a given partition.

As also discussed in Section 1.3, when the simulation system itself is evolving, the samples that we selected not only need to not only fit the current observations but also be useful in the future state of the system. This implies that the sampling algorithm has to provide sufficient diversity to help adapt to different potential futures in other words, the sampling algorithm implicitly predicts the likely futures and select

samples that are potentially useful also in the future. Therefore, realizing this special challenge for a dynamic simulation system, I further develop a sampling strategy to tackle it, I will describe it in detail in the next chapter.

Algorithm 6 Overview of Complicacy-Guided Parameter Space Sampling

Input:

A complex dynamic system, S, with N input parameters; total simulation budget, B_t; partitioning threshold, $rs_\perp$; model dictionary $\mathcal{M}$.

Output:

Sparse tensor $\boldsymbol{\mathcal{X}}$ with R partitions and their descriptive model $\boldsymbol{M_R}$.

Initialize $\boldsymbol{\mathcal{X}}$ as an empty sparse tensor,
Partition Queue $PQ \leftarrow \boldsymbol{\mathcal{X}}$,
Simulation budget left $B_l = B_t$
while $PQ \neq \emptyset$ **and** $B_l > 0$ **do**
    Current partition $P_i \leftarrow select(PQ)$, using partition selection methods described in Section 6.3.4,
    Allocate additional B_r simulations to P_i, fill $\boldsymbol{\mathcal{X}}$ with simulation output accordingly.
    Compute model ranking using model dictionary $\mathcal{M}$ and rank stability rs as described in Section 6.3.4,
    if $rs \leq rs_\perp$ **then**
        $PQ \leftarrow Par(P_i)$, where $Par()$ is a space partition scheme.
    else
        Virtually split P_i into sub-partitions,
        A model ranking is obtained for each sub-partition based on the corresponding simulation instances,
        For each sub-partition, its rank stability rs_j is computed relative to P_i.
        if $\forall rs_j \leq rs_\perp$ **then**
            $PQ \leftarrow Par(P_i)$,
        else
            No further partition is needed, use the best model with highest rank M_k as the descriptive model for partition P_i,
            $R \leftarrow P_i$
            $\boldsymbol{M_R} \leftarrow M_k$
        end if
    end if
    $B_l = B_l - B_r$.
end while
while $B_l \leq 0$ **and** $PQ \neq \emptyset$ **do**
    $P_i \leftarrow pop(PQ)$,
    use the best model with highest rank M_k as the descriptive model for partition P_i,
    $R \leftarrow P_i$
    $\boldsymbol{M_R} \leftarrow M_k$
end while

Chapter 7

PGSS: INCREMENTAL PIVOT GUIDED PARAMETER SPACE SAMPLING FOR SIMULATION ENSEMBLE GENERATION IN DYNAMIC CONTEXTS

7.1 Introduction

As also mentioned previously, data- and model-driven computer simulation ensembles are increasingly critical in many application domains that support knowledge discovery and decision making. However, obtaining these simulation ensembles is expensive, and moreover, it is costly to keep these simulation ensembles up to date as the simulations are evolving or to identify new simulation instances that predict diverse futures of the simulation system. In order to tackle these challenges, I develop a pivot guided parameter space sampling mechanism, PGSS, for maintaining simulation ensembles for dynamic systems. PGSS relies on a novel *pivot guided* mechanism to identify simulation instances to execute next as the state of the simulation changes. The experiments show that PGSS is able to achieve both better global fit and lower local errors in dynamic scenarios, compared to conventional way to explore the parameter space.

7.2 Problem Formulation

In this section, I formulate the continuous sampling problem that we aim to solve in this paper.

7.2.1 Dynamic Simulation Ensembles

Recall that, as described in Section 6.2.1, the simulation ensemble tensor $\boldsymbol{\mathcal{X}}$ is constructed by a set of simulation instances $X = \{x_j = \langle\langle v_{j,1}, \ldots, v_{j,N}\rangle, S(v_{j,1}, \ldots, v_{j,N})\rangle \parallel 1 \leq j \leq B\}$. However, as the simulation evolves, one may need to obtain more simulation instances, better aligned with the current state of the system. To achieve this, one can allocate additional budget B^t and identify a new set of simulation instance $X^t = \{x_r = \langle\langle v_{r,1}, \ldots, v_{r,N}\rangle, S(v_{r,1}, \ldots, v_{r,N})\rangle \parallel 1 \leq r \leq B^t\}$. This leads to an updated simulation ensemble tensor, $\boldsymbol{\mathcal{X}}^t$ at simulation time stamp t.

7.2.2 Desiderata for Model Learned

Let $D()$ be a decision function [1] , which takes a complex system as input and outputs a decision. Naturally, the simulation instances included in the simulation ensemble should be selected in such a way that the sparse tensor, $\boldsymbol{\mathcal{X}^t}$, would lead to similar decisions as if the complete tensor, $\boldsymbol{\mathcal{Y}^t}$, was available to the decision maker: $D(\boldsymbol{\mathcal{X}^t}) \sim D(\boldsymbol{\mathcal{Y}^t})$. However, given that in practice we neither have access to the complete tensor, $\boldsymbol{\mathcal{Y}^t}$, nor the decision function, $D()$, we need to replace this criterion with a similar, but less strict condition: assuming that the decision function, $D()$, will rely on an intermediate ensemble model, M_D, learned from the available data, we can replace the above criterion with $M_D(\boldsymbol{\mathcal{X}^t}) \sim M_D(\boldsymbol{\mathcal{Y}^t})$. This revised criterion eliminates the need to directly access the decision function, D, to assess the simulation ensemble and, instead, needs only to consider the data driven model used for decision making. However, the criterion still needs access to the complete system, $\boldsymbol{\mathcal{Y}^t}$, which in practice is not available [2] . To avoid this, we restate the desiderata only in terms

[1] The decision function, $D()$, might represent the user's domain expertise, subjective preferences, rules, and regulations.

[2] In certain situations, partial information about the system may be available, either through existing observations or through validation experiments one can execute. Without loss of generality,

of the properties of the model, $M_D(\boldsymbol{\mathcal{X}^t})$, as follows:

- *Fit/Error:* The model, $M_D(\boldsymbol{\mathcal{X}^t})$, needs to provide a good explanation of the ensemble, $\boldsymbol{\mathcal{X}^t}$. In other words, given a fit/error function $fit(\cdot,\cdot)$ (or $error(\cdot,\cdot)$) measuring how well a given model explains given data, we would desire that $fit(M_D(\boldsymbol{\mathcal{X}^t}), \boldsymbol{\mathcal{X}^t})$ is high (or $error(M_D(\boldsymbol{\mathcal{X}^t}), \boldsymbol{\mathcal{X}^t})$ is low).

- *Coverage:* Optimizing only for a good fit may not be a good strategy, as it might be possible to select a simulation ensemble with an easy to fit the model (for example, by selecting simulation instances at the flat region), but that does not accurately represent the complete system, $\boldsymbol{\mathcal{Y}}$. We, therefore, extend the above criteria as follows: Let $\boldsymbol{\mathcal{X}} \in (\mathbb{R} \cup \perp)^{I_1 \times I_2 \times \ldots \times I_N}$ be a simulation ensemble constructed with B simulation instances and $coverage(\boldsymbol{\mathcal{X}})$ be a function denoting the ratio of the simulation space $I_1 \times I_2 \times \ldots \times I_N$ covered by the simulations ensemble. We desire that $coverage(\boldsymbol{\mathcal{X}})$ is high ($\simeq 1$).

- *Diversity:* Since the system is evolving over time, it is also important that the current model, $M_D(\boldsymbol{\mathcal{X}^t})$, has sufficient inherent diversity to capture possible futures. More specifically, we would like $diversity(M_D(\boldsymbol{\mathcal{X}^t}))$, which denotes the inherent diversity of the model, to be high.

7.2.3 Problem Statement

Let us be given a complex system, S, with N input parameters, such that the i^{th} input parameter can take I_i distinct values. Let us also be given a simulation budget of $B^t \ll I_2 \times \ldots \times I_N$ simulations to execute for each time stamp t. Our goal is to construct a simulation ensemble, $\boldsymbol{\mathcal{X}^t}$, that at any time stamp t, leads to a

we ignore these partial knowledge scenarios in this paper.

data-driven model, $M_D(\boldsymbol{\mathcal{X}}^t)$, that (a) provides a good degree of explanation of the current simulation instances, (b) has a large coverage, and (c) has large diversity.

7.3 Incremental Pivot Guided Space Sampling

In this section, we propose a novel incremental *pivoted guided parameter space sampling* (PGSS) method to address the above problem. PGSS relies on (a) incremental maintenance of factorization of the simulation ensemble $\boldsymbol{\mathcal{X}}^t$ to ensure maximum ability to predict diverse future evolutions of the simulation tensor and (b) a pivot guided sample selection mechanism where the budgeted simulation samples are incrementally spent to maximize the overall model quality. In particular, we start by sampling from the complete parameter space as a whole and, recursively identify new simulation instances to be executed. This ensures that the selected simulation instances cover the entire parameter space. Then, the previously sampled instances are used as pivots to guide the selection of the next instances.

To achieve these, PGSS relies on several innovative techniques, including (a) an efficient method to incrementally maintain a High Order Singular Value Decomposition (HO-SVD) model of the simulation ensemble space as the system evolves; (b) a method to define pivots in the simulation ensemble space and their *criticalities*; and (c) strategies to decide where to allocate new simulation instances. We discuss each in detail in the following subsections.

7.3.1 Ensemble based Incremental Model Maintenance

Given a limited simulation budget, one cannot practically obtain a complete simulation ensemble and therefore, the resulting simulation ensemble tensor, $\boldsymbol{\mathcal{X}}$, is sparse. One way to obtain representative models from such sparse tensors is to seek principal patterns through tensor decomposition techniques, such as CANDECOMP [19],

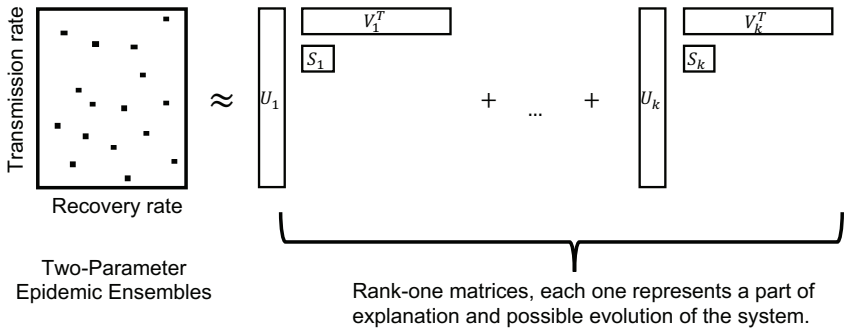


Figure 7.1: An example of two-parameter simulation scenario od the factorization based model learned

PARAFAC [42] (a.k.a. CP decomposition), and Tucker decomposition [83]. In CP decomposition, for example, a tensor $\mathcal{A}$ is approximated by the sum of k rank-one tensors such that the summation minimizes the differences from the original tensor, and these rank one components form the factor matrices of the tensor. To solve this optimization problem, one usually uses an Alternating Least Square (ALS) method; ALS estimates one factor matrix while keeping the other factor matrices fixed.

Model Redundancy and Diversity

As we discussed earlier, one of our desiderata is to maximize the model diversity or minimize the model redundancy: this is especially important as our goal is not only to select the fewest number of patterns that explain the current samples well, but also to make sure that these few patterns can adapt well to (so far unknown) alternative futures in a dynamic setting. Consider for example, the simple 2-D epidemic tensor (matrix) shown in Figure 7.1: here, we have a two-parameter epidemic simulation ensemble space represented as a 2D-tensor, $\boldsymbol{\mathcal{X}}$. This tensor is factorized into k rank-one matrices, where each can be considered as an alternative model; these models all together make up the complete model, $M_D(\boldsymbol{\mathcal{X}})$, for the ensemble.

It is easy to see that having redundancies among these models is wasteful; a less

redundant decomposition could represent the same simulation ensembles with fewer alternative models. This problem is compounded in a dynamic scenario: since we do not know how the future will evolve (including the unpredictable actions of agents external to the system), having the ability to adapt to these unpredictable futures requires us to maintain as diverse (i.e., as least redundant) k alternative models as possible. In other words, eliminating redundancies not only enables one to better explain the underlying structure of the ensemble space, but also to identify diverse patterns which could potentially help adapt to diverse potential futures of the system.

Model Orthogonality

One way to minimize the underlying model redundancy is to seek factors that are orthogonal. While this can be easily achieved for 2D-tensors (i.e. matrices) using singular-valued decomposition (SVD), maintaining orthogonality is non-trivial for tensors with higher modality. CP decomposition, for example, **does not** force its factors to be orthogonal to each other. An alternative decomposition, HO-SVD [28], however, is able to create orthogonal factors: In HO-SVD, a tensor $\mathcal{A}$ (e.g. a 3-mode tensor of size $I_1 \times I_2 \times I_3$) is approximated by its orthogonal factor matrices $\mathcal{U} = \{U_1, U_2, U_3\}$ and a core tensor $\mathcal{C}$. To be more specific, $\mathcal{A} = \mathcal{C} \times_1 U_1 \times_2 U_2 \times_3 U_3$, where U_i is the left singular vectors of mode-i unfolding of $\mathcal{A}$. Note that, while we can rely on HO-SVD to help create a diverse model $M_D(\boldsymbol{\mathcal{X}})$, as the system evolves, we also need to be able to **update** this model: a decomposition based on old simulations may not be able to accurately explain current samples and, at the same time, the old patterns are not sufficient to predict the diverse futures. Recall that at time stamp t, the simulation ensembles lead to a ensemble tensor $\boldsymbol{\mathcal{X}}^t$ with $M_D(\boldsymbol{\mathcal{X}}^t)$ which consists a set of orthogonal factor matrices $\mathcal{U}^t = \{U_i^t\}$. However, at the next time stamp $t+1$, one may identify a new set of instances to be executed, or revisit and

update some of the previous ensembles that may have become outdated. Therefore, it is critical that one selects a new set of simulation instances to execute such that the resulting simulation tensor, $\boldsymbol{\mathcal{X}}^{t+1}$ at time stamp $t+1$, is better aligned with the updated simulation instances.

Maintaining Model Orthogonality

Unfortunately, conventional HO-SVD [28] is not designed for dynamic scenarios. In this paper, we propose to incrementally maintain the HO-SVD factors of $\boldsymbol{\mathcal{X}}$ by maintaining the SVD decompositions themselves in an incremental manner. Let us assume at time stamp t, the resulting simulation ensemble tensor is $\boldsymbol{\mathcal{X}}^{t}$ of size $I_1 \times I_2 \times I_3$, which has already been factorized into a core tensor $\boldsymbol{\mathcal{C}}^{t}$ and three factor matrices $\mathcal{U}^t = \{U_1^t, U_2^t, U_3^t\}$ which consist $M_D(\boldsymbol{\mathcal{X}}^{t})$. At time stamp $t+1$, the simulation ensemble tensor will be updated to $\boldsymbol{\mathcal{X}}^{t+1}$ to include the newly executed simulation instances, we are seeking for an update factorization model, which consists of the updated core tensor core tensor $\boldsymbol{\mathcal{C}}^{t+1}$ and three updated factor matrices $\mathcal{U}^t = \{U_1^{t+1}, U_2^{t+1}, U_3^{t+1}\}$. In order to update the core tensor and factor matrices, the following update rules are employed:

$$U_i^{t+1} = incSVD(\Delta_i(\boldsymbol{\mathcal{X}}^{t+1}), U_i^t) \quad \forall i \in \{1,2,3\}$$

$$\boldsymbol{\mathcal{C}}^{t+1} = \mathcal{X}^{t+1} \times_1 U_1^{t+1} \times_2 U_2^{t+1} \times_3 U_3^{t+1}$$

where $incSVD()$ indicates the incremental update framework and $\Delta_i(\boldsymbol{\mathcal{X}}^{t+1})$ represents the update on the i^{th} mode unfolding from the updated tensor $\boldsymbol{\mathcal{X}}^{t+1}$. In particular, for this task, we rely on LWISVD [25], which incrementally maintains SVD factors in the presence of augmentation, shrinkage, and revisions to the input matrix. LWISVD is also equipped with a matrix reservoir sampling component to detect and adapt to sudden major changes in data. Such close-to-instantaneous structural

changes are commonly seen in many applications. In epidemics, for example, new interventions, such as vaccination, can significantly increase the population immunity which may significantly impact the evolution of the disease.

7.3.2 Sample Selection via Critical Pivots

Intuitively, the sample selection problem is related to tensor/matrix sketching. For instance, [90] proposed a cascade compression sampling method for matrix sketching, by first randomly sampling columns and rows from the input matrix and then computing the sketch of the intersection of these rows and columns. A second round of sampling helps further identify rows and columns that will be the "representatives" in the low-rank space. The process finishes by computing the sketch of the newly identified rows and columns. This method has been shown to work well in terms of creating a low rank approximation of a given matrix. However, it also has several drawbacks that make it unsuitable for simulation ensemble sampling: in particular, the algorithm needs to sample complete rows and columns from the input matrix. This is not practical in the simulation ensemble space, as sampling obtaining complete rows and columns may already exhaust all computational resources. In addition, it is non-trivial to adapt the algorithm to dynamically evolving scenarios. In this section, we describe a pivot based sampling selection method which overcomes these challenges through a novel concept of *critical pivots*.

Pivots

We refer to the members of the most recently executed set of simulation instances, at time t, as pivots, P^t. In general, a pivot may come from the following two scenarios:

- it may be an entirely new, freshly sampled simulation instance whose execution just completed, or

- it may be a refresh of a simulation instance which had been previously executed, but which became outdated as the system evolves.

Note that we consider only the simulated instances that have most recently been completed as pivots, as they most accurately represent the most recent system state and thus more faithfully guide selection of future samples.

Criticality of a Pivot

Maximizing fit, which quantifies how well the explanations provided by the model align with the true observations, is a common optimization criteria in many applications. When evaluating simulation ensembles, however, we may not have access to real-world data; instead, we can evaluate an ensemble $\mathcal{X}^t$ in terms of how well it is aligned with the model, $M_D(\boldsymbol{\mathcal{X}^t})$, obtained from the ensemble:

$$fit(M_D(\boldsymbol{\mathcal{X}^t})) = \frac{norm(recon(M_D(\boldsymbol{\mathcal{X}^t})) - \boldsymbol{\mathcal{X}^t})}{norm(\boldsymbol{\mathcal{X}^t})}. \tag{7.1}$$

Here, the function $norm()$ represents the Frobenius norm, and function $recon()$ represents the reconstruction of the simulation tensor via data-driven model $M_D(\boldsymbol{\mathcal{X}^t})$.

Note that, given the above definition of fit, we can also associate to each pivot a *degree of fit*, measuring the contribution of the pivot to the above term. Considering the desiderata reported in Section 7.2.2, however, a high fit is not necessarily the only criterion to consider when selecting the next pivot. In fact,

- a pivot with a good fit indicates a low approximation/reconstruction error and may lie on the "representative" location of the low rank space of the simulation ensemble [90] – thus, choosing additional pivots that are aligned with well-fitting pivots may help build on the representative region defined by these pivots to help explore the parameter space with higher effectiveness;

- on the other hand, a pivot with bad fit indicates the current data-driven model $M_D(\boldsymbol{\mathcal{X}}^t)$ does not explain the current ensemble at time t well, and, in this case, one may want to allocate more resources around the pivot to obtain an ensemble that can be better explained by a revised model.

Note that the above discussion indicates that we can measure how *critical* a pivot is (for informing us where to place the next sample) in terms of the corresponding degree of fit. However, the discussion also indicates that defining criticality is not trivial and pivots with either high or low fits, as opposed to those with mid-range fits, may be critical in further sampling. Given this, we define alternative sampling strategies below based on the degrees of fit of the pivots.

Sample Selection Strategies

Assume that we have identified a set of pivots $\mathcal{P} = \{p_1^t, p_2^t, ..., p_s^t\}$ at t. Let f_i^t denote the degree of fit for pivot p_i^t, computed by comparing its value against the corresponding value in the tensor reconstructed from $M_D(\boldsymbol{\mathcal{X}}^t)$.

Bad Fit First(BFF): In the BFF strategy, it is assumed that the criticality of a pivot p_i^t is inversely proportional to its fit f_i^t, i.e. $cr(p_i^t) = \frac{1/f_i^t}{\sum_i 1/f_i^t}$. Given this, we can define the *sampling weight* of any instance x^t as

$$w(x^t) = \sum_i cr(p_i^t) \quad \forall p_i^t \; where \; mode(x^t, p_i^t) = true, \tag{7.2}$$

where $mode()$ indicates whether the pivot p_i^t and x^t are in the same row in mode m of the tensor (e.g. in a matrix scenario, $mode()$ will check whether the pivot p_i^t and x^t will be on the same row or column).

Given all the computed weights, we can then obtained the sampling probability

of a simulation instance as

$$Prob(x^t) = \frac{w(x^t)}{\sum_{\forall x^t \in \boldsymbol{\mathcal{X}^t}} w(x^t)}. \tag{7.3}$$

Intuitively, those simulation instances that are aligned along the modes of the tensor, with pivots having low degrees of fit, will have higher probability of being sampled. As we discussed above, this can potentially improve the degrees of fit for these simulation instances that are poorly explained by the current model.

Good Fit First (GFF): In contrast, in the GFF strategy, the criticality of pivot p_i^t is assumed to be directly proportional to its fit f_i^t, i.e. $cr(p_i^t) = \frac{f_i^t}{\sum_i f_i^t}$ and the sampling probabilities are computed under this assumption. As discussed above, choosing additional samples along with the well-fitting pivots can help build on the well-fitting parts of the current low-rank approximation to more effectively explore the parameter space.

GFF and BFF with Random Sampling (GFF-RS, BFF-RS): As discussed above, using BFF and GFF, we can improve model $M_D(\boldsymbol{\mathcal{X}^t})$ by either *exploiting* around its weak regions (where bad fit occurs) or its well-fitting pivots (which may be representatives on the low-rank space). However, these completely ignore the opportunities for *exploring* the untouched areas of the simulation ensemble. In order to address this issue, we propose random sampling (RS) to complement BFF and/or GFF. In this case, we are allowing samples at random locations in parts of the parameter space that are not aligned with any of the pivots. Under BFF-RS and GSS-RS, we revise the sample selection probability as follows:

$$Prob(x^t) = \begin{cases} \frac{(1-\alpha)\times w(x^t)}{\sum_{\forall x^t \in \boldsymbol{\mathcal{X}^t}} w(x^t)} & \text{if } mode(x^t, P^t) = true \\ \frac{\alpha}{|\mathcal{X}^t| - num(P^t)} & \text{otherwise} \end{cases} \tag{7.4}$$

where α is the random sampling (i.e., exploration) rate and $num()$ is the number of potential simulation instances that are not aligned with any pivot; i.e. $mode(x^t, P^t) = false$.

Hybrid Sampling Strategy Since BFF and GFF can both improve model $M_D(\boldsymbol{\mathcal{X}^t})$ through different aspects, a natural way to combine the goodness of them is to create a hybrid sampling strategy, which allocates half of the sampling budget to samples indicated by pivots criticality using GFF and the other half of the sampling budget to samples indicated by pivots criticality using BFF, i.e. we first create sample weight using Equation 7.2, where the criticality of a pivot is defined by GFF, we identify half budget simulation instances, and then, we recreate sample weight using Equation 7.2 again, where the criticality of a pivot is defined by BFF, we identify the rest half budget simulation instances. In this way, we not only will select simulation instance that exploits around the weak regions, where the fit is bad, but also around the well-fitting pivots, which may be representatives on the low-rank space.

7.4 Experiments

In this section, we present experimental evaluations under various settings and simulation scenarios.

7.4.1 Setup

To evaluate PGSS, we use several simulation systems with varying properties and complexities.

Continuous Complex System

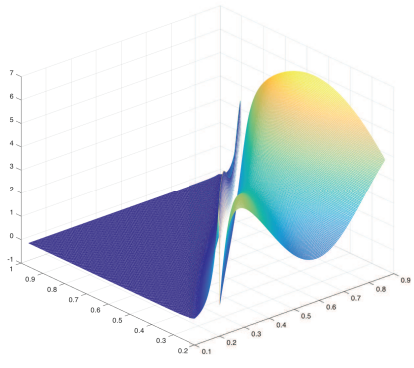
First, we consider two *2-parameter complex systems* (referred as Cont. and Cont.2 in experiments) depicted in Figure 7.2(a), where $f(x_1, x_2) = \Lambda(x_1, x_2)\big(2 \times sin(8x_1) + 5sin(3x_2)\big)$ when $x_1 < x_2$ and $f(x_1, x_2) = (1 - \Lambda(x_1, x_2))\big(2 \times sin(8x_1) + 5sin(3x_2)\big)$ when $x_1 \geq x_2$, and Figure 7.2(b), where $f(x_1, x_2) = \Lambda(x_1, x_2) \times 2sin(x_1) + 5x_2^2$, when $x_1 < x_2$ and $f(x_1, x_2) = (1 - \Lambda(x_1, x_2)) \times 2sin(x_1) + 5x_2)$ when $x_1 \geq x_2$ Here, $\Lambda(x_1, x_2)$ is an exponential smoothing term to smooth the transition, where $x_1 \sim x_2$. We create 50 iterations of synthetic update, drawn from a Gaussian distribution of $(\mu = 0, \sigma^2 = 5)$.

Piecewise Complex System

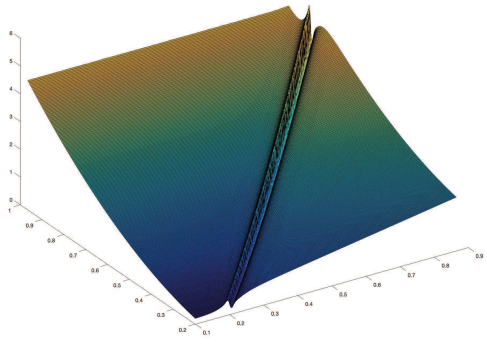
We also consider two *piecewise complex systems* (referred as Piece. and Piece.2 in experiments), shown in Figure A.1 and Figure A.2 in Appendix A: the parameter space is split into 14 regions of different sizes and shapes (linear, polynomial, exponential, or hybrid). Again, we create 50 iterations of updates, drawn from a Gaussian distribution of $(\mu = 0, \sigma^2 = 5)$.

Dynamic Pendulum System

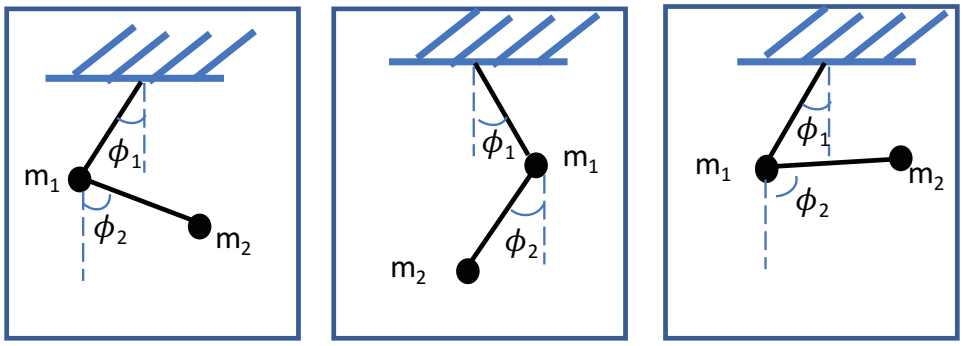
As the third simulation system, we consider a double-pendulum system, shown in Figure 7.2(c), with five variables: gravity, g, initial angles, θ_1 and θ_2, and masses, m_1 and m_2, of the two pendulums. We created two subsets of dynamic systems, in one set, we fixed θ_2 and m_2 and varied the rest, e.g. we get simulation dynamics for the first pendulum (we refer this as Pendulum1 in the experiments). In other set, we fixed θ_1 and m_1 and varied the rest, so in this way, we get simulation dynamics for the second pendulum (we refer this as Pendulum2 in the experiments). We select a random initialization of these variables as the ground truth, for each simulation



(a) continuous complex 1



(b) continuous complex 2



(c) double-pendulum system

Figure 7.2: Dynamic Simulation Scenarios

instance, we consider the Euclidean distance from this ground truth as the output of the system. We slice the total simulation time into 50 time stamps and consider each as a different simulation state.

Dynamic Epidemic Simulations

Finally, we use epidemic simulations created using STEM [1]: we consider a simulation scenario, where an epidemic is occurring in the U.S. for a full year. We use the SEIR model with three input parameters (transmission, recovery, and mortality rates), where each parameter is a continuous real number ranging from 0 to 1, and we

consider 40 distinct values for each parameter, i.e., they all vary from 0.025 to 1 with increments of 0.025. For the simulation outcome, we focus on the number of deaths over time. We select a random initialization of these variables as the ground truth and, for each simulation, we output the Euclidean distance from the ground truth as the system output. We slice the whole simulation into 12 time stamps, corresponding to distinct months of the simulated year.

7.4.2 Default Parameters

In our experiments, we set the default per time stamp sampling budget B^t to 0.02% if the volume of the space. The target rank for HO-SVD is set to 25% of the size of the parameter space. The default exploration probability for BFF-RS and GFF-RS is set to 0.15. All experiments were run for 50 times and averages are reported.

7.4.3 Evaluation Measures

In order to evaluate the sampling strategies we consider the following error/accuracy measures:

- *GF (Global Fit):* GF measures the overall quality of the model as it quantifies the total prediction residual, which is defined in Equation 7.1.
- *SLE (Sum of Local Errors):* SLE is analogous to sum squared error (SSE), but is evaluated on the pivots. Consequently, this indicates how well the selected pivots are explained by the models refined using them.

In addition to these, we also consider *diversity* measured through angular penalty that sums up the cosine similarity of the vectors in each factor [43]: the lower the angular penalty, the more diverse the corresponding models.

Table 7.1: Percent *increase* in Global Fit (GF) over purely random sampling for different sampling strategies

Strategy	Piece.	Piece.2	Cont.	Cont.2	Pendulum1	Pendulum2	Epidemic
PGSS/GFF-RS	13.6%	13.4%	16.4%	18.9%	6.4%	5.9%	15.6%
PGSS/BFF-RS	11.8%	11.4%	10.9%	12.3%	9.4%	8.8%	21.1%
PGSS/Hybrid-RS	14.5%	14.2%	17.2%	20.3%	8.2%	8.5%	16.5%
Latin	0.9%	0.9%	1.4%	1.9%	0.6%	0.6%	2.7%

Table 7.2: Percent *reduction* in Sum of Local Errors (SLE) over purely random sampling for different sampling strategies

Strategy	Piece.	Piece.2	Cont.	Cont.2	Pendulum1	Pendulum2	Epidemic
PGSS/GFF-RS	83.6%	81.9%	82.0%	83.5%	2.4%	1.6%	6.5%
PGSS/BFF-RS	70.7%	68.3%	56.6%	61.6%	-17.9%	-17.4%	6.0%
PGSS/Hybrid-RS	76.5%	74.1%	73.9%	78.3%	-3.5%	-5.3%	6.4%
Latin	-4.9%	-5.1%	6.3%	6.5%	-19.1%	-19.5%	0.8%

7.4.4 Discussion of the Results

Accuracy

Tables 7.1, 7.2 and Figure 7.3, 7.4 provide an overview of the accuracy results for the GFF-RS, BFF-RS and Hybrid-RS based pivot selection strategies (with 0.15 random exploration probability) and Latin hypercube sampling (Latin) [62] compared against purely random sample selection (pure RS). As we see in the first table, on all four

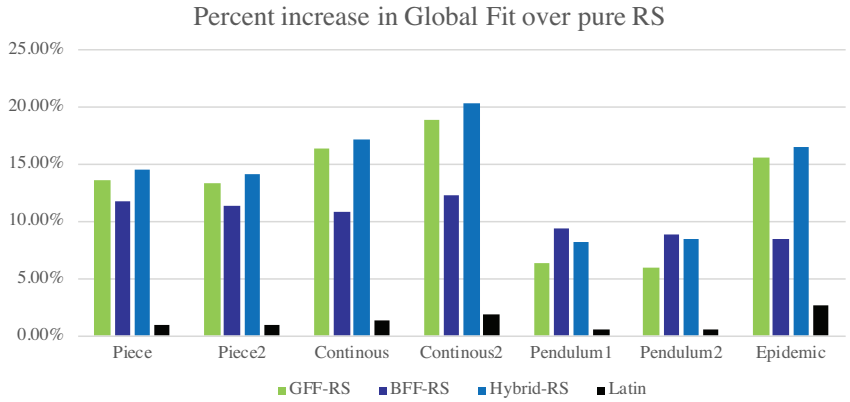


Figure 7.3: Percent *increase* in Global Fit (GF) over purely random sampling for different sampling strategies

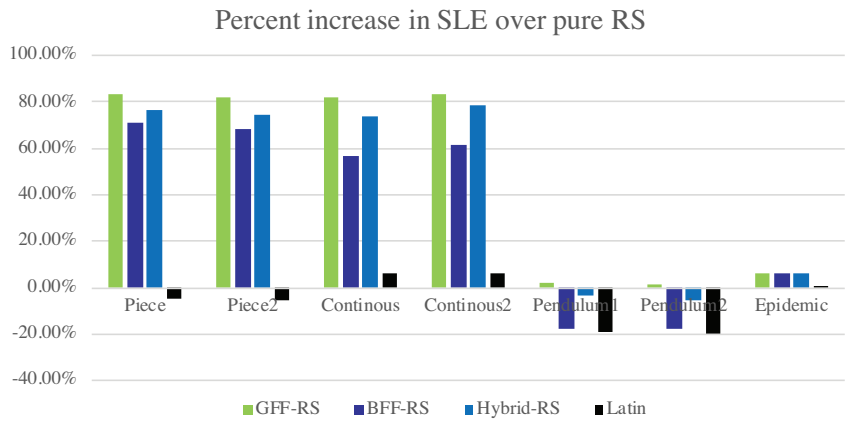


Figure 7.4: Percent *reduction* in Sum of Local Errors (SLE) over purely random sampling for different sampling strategies

data sets, the proposed PGSS algorithm with pivot based sampling leads to much better global fits than purely random sampling, while the Latin hypercube sampling only achieves minimal improvement over purely random sampling. It is interesting to note that **both** GFF, BFF and Hybrid strategies outperform pure RS, confirming our hypothesis that **both** higher-than-average and lower-than-average degrees of fit point toward pivots that can signal where to obtain future simulation samples, furthermore, by selecting from both of them combines the advantages and even boost the accuracy. Table 7.2 and Figure 7.4 shows that, also in terms of SLE, the GFF, BFF and hybrid strategies of PGSS lead to significant improvements over purely random sampling, with the only exception being the BFF strategy for the *pendulum* scenario, which leads to $\sim 18\%$ increase in local errors. This, along with the observation (in Table 7.1 and Figure 7.3) that the BFF strategy leads to a higher global fit than GFF for the *pendulum* scenario, indicates that, for this highly complex system, selecting samples along poorly fitting samples helps to better explore the parameter space by selecting samples at *difficult to fit* portions of the parameter space. By using Hybrid strategy, one can trade off between the gain in global fit and sum of local errors. Again, this confirms that Latin has minimal improvement over pure RS.

Table 7.3: Impact of diversity maximization – PGSS vs. CP vs. Cp-ortho [2] (dynamic epidemic system)

Strategy	**Ang.Pen./Max.Ang.Pen.**	**GF Improvement over CP**
PGSS (GFF-RS)	0/135	5.8%
Cp-ortho (GFF-RS)	22.4/135	-16.1%
CP (GFF-RS)	48.3/135	N/A

Table 7.4: Impact of random sampling rate – Percent *increase* in Global Fit over pure RS (continuous complex system).

Random sampling rate α	**GFF (0%)**	**GFF-RS (5%)**	**GFF-RS (15%)**	**GFF-RS (85%)**	**pure RS (100%)**
Fit gain	7.4%	7.7%	16.4%	1.8%	N/A

Orthogonality

Table 7.3 show the impact of diversity maximization strategy used in PGSS. In this table, we compare PGSS and Cp-ortho developed in [2] against CP-based decomposition – both leveraging GFF-RS for sample selection – for the epidemic system.

As we see in the table, under CP, the angular penalty of the factors sum up to 48.3, while the maximum possible angular penalty for this scenario is 135, resulting in a redundancy ratio of $\sim$ 36% for CP. In contrast, the proposed PGSS scheme, presented in Section 7.3.1, leads to perfectly orthogonal factors, with 0 redundancy, while in Cp-ortho, there is still a redundancy ratio of $\sim$ 16%. The last column, then, assesses how much improvement in global fit we are able to obtain relative to CP: as we see here, PGSS is able provide 5.8% improvement in global fit by (incrementally) maintaining orthogonal factors with zero redundancy, while Cp-ortho provides even worse global fit due to its optimization to balance fit and angular penalty. [3]

[3]In Cp-ortho, the algorithm requires setting data-specific hyperparameters for the optimizations, and the parameters suggested in [2] did not provide satisfactory results on epidemic data, we performed a grid search and reported the best results so far.

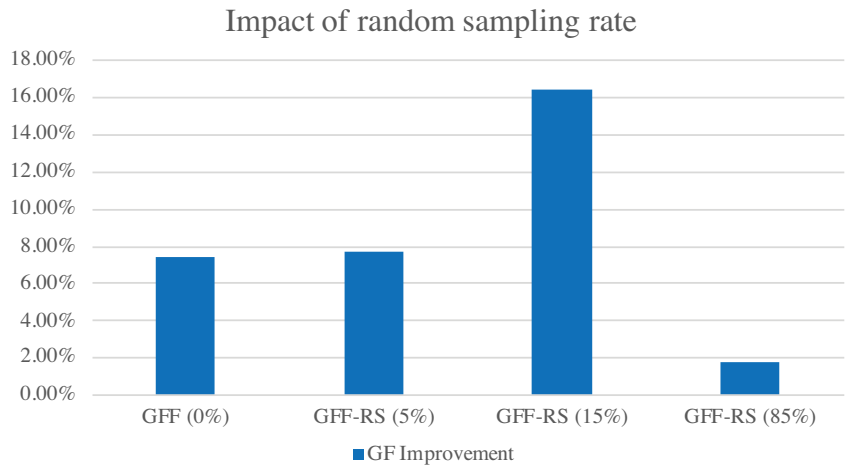


Figure 7.5: Impact of random sampling rate – Percent *increase* in Global Fit over pure RS (continuous complex system).

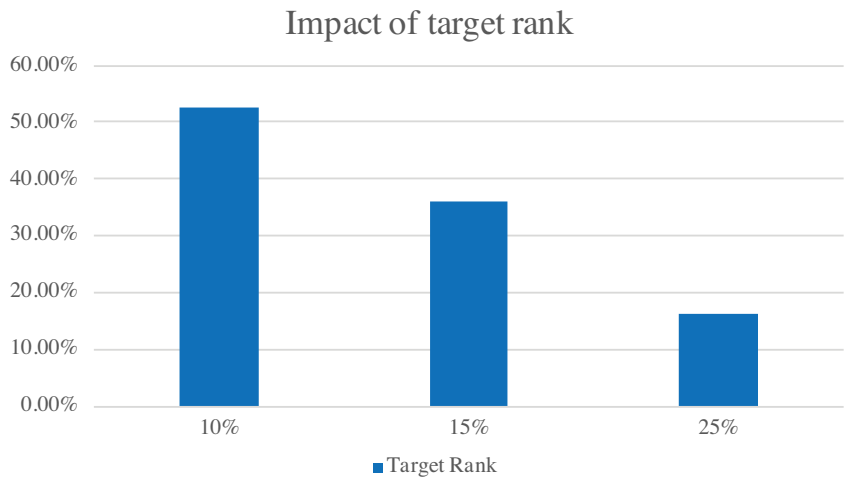


Figure 7.6: Impact of target rank – Percent *increase* in Global Fit over pure RS (continuous complex system).

Table 7.5: Impact of target rank – Percent *increase* in Global Fit over pure RS (continuous complex system).

Target Rank	**10%**	**15%**	**25%**
Fit gain	52.3%	35.9%	16.4%

Table 7.6: Impact of sampling budget – Percent *increase* in Global Fit over pure RS (continuous complex system).

Sampling Budget	**0.01%**	**0.02%**	**0.03%**
Fit gain	8.2%	16.4%	19.1%

Varying Random Exploration Rates

Table 7.4 and Figure 7.5 study the impact of the random sampling rate α over the global fit. As we see, neither GFF (pure exploitation), nor pure RS (pure exploration) lead to best global fits. Instead, the GFF-RS with 15% random sampling rate leads to best overall accuracy, indicating that GFF-based exploitation is indeed effective, and is further strengthened by some limited random exploration.

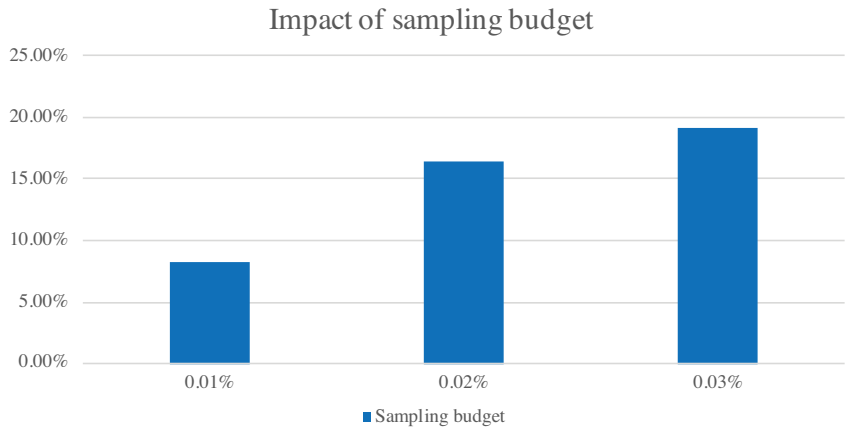


Figure 7.7: Impact of sampling budget – Percent *increase* in Global Fit over pure RS (continuous complex system).

Varying Target Ranks

Table 7.5 and Figure 7.6 study the impact of different target ranks of the decomposition. Note that, the higher the target rank, the more complex the discovered model. As we see in the figure, both GFF-RS and pure RS achieve improved global fits as the target rank increases. On the other hand, GFF-RS provide better accuracy than pure RS for all target ranks considered; moreover, the relative accuracy of GFF-RS over RS increases as we seek for models that are less complex (and thus likely to be easier to interpret).

Varying Simulation Budgets

Table 7.6 and Figure 7.7 study the impact of different sampling budgets. As we expect, large budgets lead to higher global fit for both GFF-RS and pure RS. However, as the available budget increases, GFF-RS is able to leverage the extra samples more effectively, boosting the global fit faster than pure RS.

7.5 Conclusion

In this work, I developed a knowledge discovery framework to create simulation ensembles to support decision making for complex, dynamic systems. The proposed pivot guided simulation sampling (PGSS) algorithm is able to incrementally maintain a diverse model of the simulation ensemble and iteratively identify new simulation instances to execute, as the system evolves. The experiments have also shown that, using PGSS, one can maximize the diversity of the models learned from simulation ensembles, while obtaining better local and global fits.

Chapter 8

CONCLUSIONS

The main goal of this dissertation is to design and develop efficient model learning algorithms that work in a streaming environment. Particularly, I look at three different models as mentioned in Chapter 1: a) Latent Factor Model, b) Probabilistic Generative Model and, c) Blackbox Model. Different model makes different assumptions are requires different level of domain knowledge of the data. Latent factor model assumes that every observation is of a degree of membership to a cluster, instead of assigning clusters explicitly in the probabilistic model. The probabilistic model usually assumes that the observation is generated from a set of latent variables with different distributions and sampling methods can be applied to maximize the posterior probability given this assumption and thus a probabilistic model can be learned from the data. In Blackbox model, one cannot directly infer the parameters from the observations since the underlying model is not accessible or too complex and thus sampling has to be involved when exploring it.

In Chapter 3, realizing that SVD is computationally costly and a naive implementation does not scale to the needs of scenarios where data evolves continuously, I developed a *Low-rank, Windowed, Incremental SVD* (LWI-SVD) algorithm, which leverages efficient and accurate *low-rank approximations* to speed up incremental SVD updates. In addition, LWI-SVD also uses a *window-based* approach to aggregate multiple incoming updates (insertion or deletions) and, thus reduces online costs. Furthermore, to reduce the error accumulated in the continuous decomposition, I designed the LWI2-SVD algorithm, which leverages a novel partial reconstruction based change detection technique to support timely refreshing of the decompositions

to prevent accumulation of errors.

In Chapter 4, I realize that NMF is another way of obtaining low rank approximations of a given data matrix and however, it is still computationally costly which makes it hard to scale to many online applications. Moreover, many applications include integration of multiple data streams (with potential overlaps) and/or involves tracking of multiple similar (but different) queries To tackle these challenges, I developed *Group Incremental Non-Negative Matrix Factorization* (GI-NMF) which leverages redundancies across multiple NMF tasks over data streams. I derive *group multiplicative update rules* (G-MUR) to reuse the factorization results from the redundancies and significantly improve the efficiency with negligible error overhead. Moreover, the GI-NMF algorithm is able to incrementally update the factors in a streaming environment so that it scales to the need of online applications.

In Chapter 5, realizing that latent topics from the data may have varying spans and topics of multiple scales can co-exist in a single web or social media data stream, I designed a Multi-Scale Dynamic Topic Model, which capture the latent topics and their dynamics simultaneously, at different scales, i.e. both "past" and "now" in multiple scales. In addition, I developed a multi-scale incremental Gibbs sampling mechanism to efficiently leverage the overlaps among various scales to speed up the online inference steps.

In Chapter 6, to tackle the three main challenges in black box models, i.e. a) limited ensemble simulation budgets, b) inherent data sparsity of simulation ensembles and c) post-simulation interpretation and knowledge discovery, I developed a complicacy-guided parameter space sampling method to select from a large space of a potential simulations, a subset to be executed and to optimize learned models descriptive/predictive power, i.e. to maximize the ease-of-interpretation (in addition to its descriptive power). More specifically, CPSS splits the simulation parameter space

in a top-down fashion, by incrementally and adaptively assigning simulation instances to different parts of the input space, in such a way that each resulting partition of the parameter space is explained with a model with good fit and low complicacy.

In Chapter 7, realizing that there are additional challenges when the simulation system itself is evolving and existing simulation ensembles may become outdated, and be insufficient to accurately describe and predict future events. I further designed a novel incremental *pivot guided parameter space sampling* (PGSS) method to tackle the challenge. PGSS is to able to maintain an orthonormal basis for the simulation ensemble space as new simulation instances arrive incrementally, and it has a pivot guided sample selection mechanism to incrementally and adaptively select the simulation instances to execute. I also experimentally showed that using PGSS can maximize the diversity and obtain good global and local fits.

REFERENCES

[1] "Stem. the spatiotemporal epidemiological modeler project", `http://www.eclipse.org/stem` (2018).

[2] Afshar, A., J. C. Ho, B. Dilkina, I. Perros, E. B. Khalil, L. Xiong and V. Sunderam, "Cp-ortho: An orthogonal tensor factorization framework for spatio-temporal data", in "Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems", p. 67 (ACM, 2017).

[3] Agarwal, D. and B. Chen, "Regression-based latent factor models", in "Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009", pp. 19–28 (2009).

[4] Ahmed, A., Y. Low, M. Aly, V. Josifovski and A. J. Smola, "Scalable distributed inference of dynamic user interests for behavioral targeting", in "Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011", pp. 114–122 (2011).

[5] Akaike, H., "Information theory and an extension of the maximum likelihood principle", in "Breakthroughs in statistics", pp. 610–624 (Springer, 1992).

[6] Berry, M. W., Z. Drmac and E. R. Jessup, "Matrices, vector spaces, and information retrieval", SIAM review **41**, 2, 335–362 (1999).

[7] Berry, M. W. and D. I. Martin, "Principal component analysis for information retrieval", in "Handbook of parallel computing and statistics", pp. 415–430 (Chapman and Hall/CRC, 2005).

[8] Berry, M. W., S. A. Pulatova and G. Stewart, "Algorithm 844: Computing sparse reduced-rank approximations to sparse matrices", ACM Transactions on Mathematical Software (TOMS) **31**, 2, 252–269 (2005).

[9] Bhadury, A., J. Chen, J. Zhu and S. Liu, "Scaling up dynamic topic models", in "Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016", pp. 381–390 (2016).

[10] Blei, D. M., M. I. Jordan *et al.*, "Variational inference for dirichlet process mixtures", Bayesian analysis **1**, 1, 121–143 (2006).

[11] Blei, D. M. and J. D. Lafferty, "Dynamic topic models", in "Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006", pp. 113–120 (2006).

[12] Blei, D. M. and J. D. Lafferty, "Topic models", in "Text Mining", pp. 101–124 (Chapman and Hall/CRC, 2009).

[13] Blei, D. M. and J. D. McAuliffe, "Supervised topic models", in "Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007", pp. 121–128 (2007).

[14] Blei, D. M., A. Y. Ng and M. I. Jordan, "Latent dirichlet allocation", Journal of Machine Learning Research **3**, 993–1022 (2003).

[15] Brand, M., "Fast low-rank modifications of the thin singular value decomposition", Linear algebra and its applications **415**, 1, 20–30 (2006).

[16] Bucak, S. S. and B. Gunsel, "Video content representation by incremental non-negative matrix factorization", in "2007 IEEE International Conference on Image Processing", vol. 2, pp. II–113 (IEEE, 2007).

[17] Candan, K. S. and M. L. Sapino, *Data management for multimedia retrieval* (Cambridge University Press, 2010).

[18] Canini, K. R., L. Shi and T. L. Griffiths, "Online inference of topics with latent dirichlet allocation", in "Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS 2009, Clearwater Beach, Florida, USA, April 16-18, 2009", pp. 65–72 (2009).

[19] Carroll, J. D. and J.-J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of eckart-young decomposition", Psychometrika **35**, 3, 283–319 (1970).

[20] Chandrasekaran, S., B. S. Manjunath, Y. Wang, J. Winkeler and H. Zhang, "An eigenspace update algorithm for image analysis", CVGIP: Graphical Model and Image Processing **59**, 5, 321–332 (1997).

[21] Chang, J., J. L. Boyd-Graber, S. Gerrish, C. Wang and D. M. Blei, "Reading tea leaves: How humans interpret topic models", in "Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada.", pp. 288–296 (2009).

[22] Chen, C.-h. and L. H. Lee, *Stochastic simulation optimization: an optimal computing budget allocation*, vol. 1 (World scientific, 2011).

[23] Chen, X. and K. S. Candan, "Gi-nmf: Group incremental non-negative matrix factorization on data streams", in "Proceedings of the 23rd ACM international conference on conference on information and knowledge management", pp. 1119–1128 (ACM, 2014).

[24] Chen, X. and K. S. Candan, "LWI-SVD: low-rank, windowed, incremental singular value decompositions on time-evolving data sets", in "The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014", pp. 987–996 (2014).

[25] Chen, X. and K. S. Candan, "LWI-SVD: low-rank, windowed, incremental singular value decompositions on time-evolving data sets", in "The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014", pp. 987–996 (2014).

[26] Chen, X., K. S. Candan and M. L. Sapino, "Ims-dtm: Incremental multi-scale dynamic topic models", in "Thirty-Second AAAI Conference on Artificial Intelligence", (2018).

[27] Crawley, D. B., L. K. Lawrie, F. C. Winkelmann, W. F. Buhl, Y. J. Huang, C. O. Pedersen, R. K. Strand, R. J. Liesen, D. E. Fisher, M. J. Witte *et al.*, "Energyplus: creating a new-generation building energy simulation program", Energy and buildings **33**, 4, 319–331 (2001).

[28] De Lathauwer, L., B. De Moor and J. Vandewalle, "A multilinear singular value decomposition", SIAM journal on Matrix Analysis and Applications **21**, 4, 1253–1278 (2000).

[29] Deerwester, S., S. T. Dumais, G. W. Furnas, T. K. Landauer and R. Harshman, "Indexing by latent semantic analysis", Journal of the American society for information science **41**, 6, 391–407 (1990).

[30] Deng, H., J. Han, B. Zhao, Y. Yu and C. X. Lin, "Probabilistic topic models with biased propagation on heterogeneous information networks", in "Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining", pp. 1271–1279 (ACM, 2011).

[31] Donoho, D. and V. Stodden, "When does non-negative matrix factorization give a correct decomposition into parts?", in "Advances in neural information processing systems", pp. 1141–1148 (2004).

[32] Fisher, R. A., "The design of experiments", (1935).

[33] Gaber, M. M., A. Zaslavsky and S. Krishnaswamy, "Mining data streams: a review", ACM Sigmod Record **34**, 2, 18–26 (2005).

[34] Gao, D., Y. Lou, D. He, T. Porco, Y. Kuang, G. Chowell and S. Ruan, "Prevention and control of zika as a mosquito-borne and sexually transmitted disease: A mathematical modeling analysis", Cambridge University Press (2010).

[35] Garofalakis, M., J. Gehrke and R. Rastogi, "Querying and mining data streams: you only get one look a tutorial", in "Proceedings of the 2002 ACM SIGMOD international conference on Management of data", pp. 635–635 (ACM, 2002).

[36] Gelman, A., H. S. Stern, J. B. Carlin, D. B. Dunson, A. Vehtari and D. B. Rubin, *Bayesian data analysis* (Chapman and Hall/CRC, 2013).

[37] Golub, G. H. and C. Reinsch, "Singular value decomposition and least squares solutions", in "Linear Algebra", pp. 134–151 (Springer, 1971).

[38] Gonzalez, J., Y. Low, A. Gretton and C. Guestrin, "Parallel gibbs sampling: From colored fields to thin junction trees", in "Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011", pp. 324–332 (2011).

[39] Griffiths, T. L. and M. Steyvers, "Finding scientific topics", Proceedings of the National academy of Sciences **101**, suppl 1, 5228–5235 (2004).

[40] Gu, M. and S. C. Eisenstat, "A stable and fast algorithm for updating the singular value decomposition", (1994).

[41] Gu, M. and S. C. Eisenstat, "Downdating the singular value decomposition", SIAM J. Matrix Analysis Applications **16**, 3, 793–810 (1995).

[42] Harshman, R. A., "Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis", (1970).

[43] Henderson, J., J. C. Ho, A. N. Kho, J. C. Denny, B. A. Malin, J. Sun and J. Ghosh, "Granite: Diversified, sparse tensor factorization for electronic health record-based phenotyping", in "Healthcare Informatics (ICHI), 2017 IEEE International Conference on", pp. 214–223 (IEEE, 2017).

[44] Hmedeh, Z., H. Kourdounakis, V. Christophides, C. Du Mouza, M. Scholl and N. Travers, "Subscription indexes for web syndication systems", in "Proceedings of the 15th International Conference on Extending Database Technology", pp. 312–323 (ACM, 2012).

[45] Hong, L., B. Dom, S. Gurumurthy and K. Tsioutsiouliklis, "A time-dependent topic model for multiple text streams", in "Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining", pp. 832–840 (ACM, 2011).

[46] Huang, S., K. S. Candan and M. L. Sapino, "BICP: block-incremental CP decomposition with update sensitive refinement", in "Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016", pp. 1221–1230 (2016).

[47] Iwata, T., T. Yamada, Y. Sakurai and N. Ueda, "Online multiscale dynamic topic models", in "Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, July 25-28, 2010", pp. 663–672 (2010).

[48] Järvelin, K. and J. Kekäläinen, "Cumulated gain-based evaluation of IR techniques", ACM Trans. Inf. Syst. **20**, 4, 422–446 (2002).

[49] Johnson, N., "Sequential analysis: A survey", Journal of the Royal Statistical Society. Series A (General) pp. 372–411 (1961).

[50] Kawamae, N., "Trend analysis model: trend consists of temporal words, topics, and timestamps", in "Proceedings of the Forth International Conference on Web Search and Web Data Mining, WSDM 2011, Hong Kong, China, February 9-12, 2011", pp. 317–326 (2011).

[51] Kiskowski, M. and G. Chowell, "Modeling household and community transmission of ebola virus disease: epidemic growth, spatial dynamics and insights for epidemic control", Virulence **7**, 2, 163–173 (2016).

[52] Kolda, T. G. and B. W. Bader, "Tensor decompositions and applications", SIAM review **51**, 3, 455–500 (2009).

[53] Lee, D. D. and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization", Nature **401**, 6755, 788 (1999).

[54] Lee, D. D. and H. S. Seung, "Algorithms for non-negative matrix factorization", in "Advances in neural information processing systems", pp. 556–562 (2001).

[55] Levey, A. and M. Lindenbaum, "Sequential karhunen-loeve basis extraction and its application to images", IEEE Transactions on Image processing **9**, 8, 1371–1374 (2000).

[56] Li, X., K. S. Candan and M. L. Sapino, "ntd: Noise-profile adaptive tensor decomposition", in "Proceedings of the 26th International Conference on World Wide Web", pp. 243–252 (International World Wide Web Conferences Steering Committee, 2017).

[57] Li, X., K. S. Candan and M. L. Sapino, "M2td: multi-task tensor decomposition for sparse ensemble simulations", in "2018 IEEE 34th International Conference on Data Engineering (ICDE)", pp. 1144–1155 (IEEE, 2018).

[58] Li, X., S. Huang, K. S. Candan and M. L. Sapino, "Focusing decomposition accuracy by personalizing tensor decomposition (ptd)", in "Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management", pp. 689–698 (ACM, 2014).

[59] Lin, J., E. J. Keogh, L. Wei and S. Lonardi, "Experiencing SAX: a novel symbolic representation of time series", Data Min. Knowl. Discov. **15**, 2, 107–144 (2007).

[60] Liu, S., Y. Garg, K. S. Candan, M. L. Sapino and G. Chowell-Puente, "Notes2: Networks-of-traces for epidemic spread simulations", in "Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence", (2015).

[61] Liu, S., S. Poccia, K. S. Candan, G. Chowell and M. L. Sapino, "epidms: data management and analytics for decision-making from epidemic spread simulation ensembles", The Journal of infectious diseases **214**, suppl_4, S427–S432 (2016).

[62] McKay, M. D., R. J. Beckman and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code", Technometrics **42**, 1, 55–61 (2000).

[63] McLachlan, G. J. and D. Peel, "Mixtures of factor analyzers", in "Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000", pp. 599–606 (2000).

[64] Minka, T., "Estimating a dirichlet distribution", (2000).

[65] Montgomery, D. C., E. A. Peck and G. G. Vining, *Introduction to linear regression analysis*, vol. 821 (John Wiley & Sons, 2012).

[66] Nallapati, R., A. Ahmed, E. P. Xing and W. W. Cohen, "Joint latent topic models for text and citations", in "Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008", pp. 542–550 (2008).

[67] Paisley, J., C. Wang, D. M. Blei and M. I. Jordan, "Nested hierarchical dirichlet processes", IEEE Transactions on Pattern Analysis and Machine Intelligence **37**, 2, 256–270 (2015).

[68] Papadimitriou, S., J. Sun and C. Faloutsos, "Streaming pattern discovery in multiple time-series", in "Proceedings of the 31st international conference on Very large data bases", pp. 697–708 (VLDB Endowment, 2005).

[69] Pedrielli, G. and S. H. Ng, "G-star: a new kriging-based trust region method for global optimization", in "Proceedings of the 2016 Winter Simulation Conference", pp. 803–814 (IEEE Press, 2016).

[70] Pedrielli, G., Y. Zhu and L. H. Lee, "Single-run simulation optimization through time dilation and optimal computing budget allocation", in "Proceedings of the 10th Conference on Stochastic Models of Manufacturing and Service Operations", pp. 187–194 (2015).

[71] Pedrielli, G., Y. Zhu, L. H. Lee and H. Li, "Empirical analysis of the performance of variance estimators in sequential single-run ranking & selection: the case of time dilation algorithm", in "Proceedings of the 2016 Winter Simulation Conference", pp. 738–748 (IEEE Press, 2016).

[72] Petersen, K. B., M. S. Pedersen *et al.*, "The matrix cookbook", Technical University of Denmark **7**, 15, 510 (2008).

[73] Petrovic, M., H. Liu and H.-A. Jacobsen, "Cms-topss: efficient dissemination of rss documents", in "Proceedings of the 31st international conference on Very large data bases", pp. 1279–1282 (VLDB Endowment, 2005).

[74] Phan, A. H. and A. Cichocki, "Advances in parafac using parallel block decomposition", in "International Conference on Neural Information Processing", pp. 323–330 (Springer, 2009).

[75] Phan, A. H. and A. Cichocki, "Parafac algorithms for large-scale problems", Neurocomputing **74**, 11, 1970–1984 (2011).

[76] Poccia, S. R., M. L. Sapino, L. Sicong, C. Xilun, G. Yash, H. Shengyu, H. K. Jung, L. Xinsheng, N. Parth, K. Selcuk Candan *et al.*, "Simdms: Data management and analysis to support decision making through large simulation ensembles", in "20th International Conference on Extending Database Technology (EDBT'17)", pp. 582–585 (OpenProceedings. org, 2017).

[77] Porteous, I., D. Newman, A. T. Ihler, A. U. Asuncion, P. Smyth and M. Welling, "Fast collapsed gibbs sampling for latent dirichlet allocation", in "Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008", pp. 569–577 (2008).

[78] Rebhan, S., W. Sharif and J. Eggert, "Incremental learning in the non-negative matrix factorization", in "International Conference on Neural Information Processing", pp. 960–969 (Springer, 2008).

[79] Silberstein, A., J. Terrace, B. F. Cooper and R. Ramakrishnan, "Feeding frenzy: selectively materializing users' event feeds", in "Proceedings of the 2010 ACM SIGMOD International Conference on Management of data", pp. 831–842 (ACM, 2010).

[80] Stewart, G., "Four algorithms for the the efficient computation of truncated pivoted qr approximations to a sparse matrix", Numerische Mathematik **83**, 2, 313–323 (1999).

[81] Sun, J., S. Papadimitriou and C. Faloutsos, "Online latent variable detection in sensor networks", in "21st International Conference on Data Engineering (ICDE'05)", pp. 1126–1127 (IEEE, 2005).

[82] Sun, J., D. Tao, S. Papadimitriou, P. S. Yu and C. Faloutsos, "Incremental tensor analysis: Theory and applications", ACM Transactions on Knowledge Discovery from Data (TKDD) **2**, 3, 11 (2008).

[83] Tucker, L. R., "Some mathematical notes on three-mode factor analysis", Psychometrika **31**, 3, 279–311 (1966).

[84] Van den Broeck, W., C. Gioannini, B. Gonçalves, M. Quaggiotto, V. Colizza and A. Vespignani, "The gleamviz computational tool, a publicly available software to explore realistic epidemic spreading scenarios at the global scale", BMC infectious diseases **11**, 1, 37 (2011).

[85] Vitter, J. S., "Random sampling with a reservoir", ACM Transactions on Mathematical Software (TOMS) **11**, 1, 37–57 (1985).

[86] Wang, C., D. M. Blei and D. Heckerman, "Continuous time dynamic topic models", in "UAI 2008, Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence, Helsinki, Finland, July 9-12, 2008", pp. 579–586 (2008).

[87] Wang, Y. and G. Mori, "Human action recognition by semilatent topic models", IEEE Trans. Pattern Anal. Mach. Intell. **31**, 10, 1762–1774 (2009).

[88] Xu, W., X. Liu and Y. Gong, "Document clustering based on non-negative matrix factorization", in "Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval", pp. 267–273 (ACM, 2003).

[89] Yao, L., D. Mimno and A. McCallum, "Efficient methods for topic model inference on streaming document collections", in "Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining", pp. 937–946 (ACM, 2009).

[90] Zhang, K., C. Liu, J. Zhang, H. Xiong, E. Xing and J. Ye, "Randomization or condensation?: Linear-cost matrix sketching via cascaded compression sampling", in "Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining", pp. 615–623 (ACM, 2017).

[91] Zhao, Z. A. and H. Liu, *Spectral feature selection for data mining* (Chapman and Hall/CRC, 2011).

APPENDIX A

SYNTHETIC SIMULATION DATASETS

In this Appendix, I present sample data used in Chapter 6 and Chapter 7.

Piecewise Function 1 The first piecewise function is depicted in Figure A.1 and each function is described below:

Linear function 1: $2 \cdot X - 3 \cdot Y$
Linear function 2: $10 \cdot X + 10 \cdot Y$
Linear function 3: $10 \cdot X - 2 \cdot Y$
Linear function 4: $-X + 5 \cdot Y$
Polynomial function 1: $X^3 + Y^2$
Polynomial function 2: $5 \cdot X^3 + Y$
Polynomial function 3: $-X^2 + Y^2$
Sine function 1: $8 \cdot \sin(X) + 3 \cdot \sin(Y)$
Sine function 2: $8 \cdot \sin(X) - 2 \cdot \cos(Y)$
Exponential function 1: $\exp(\frac{X+Y}{10})$
Exponential function 2: $\exp(X^2 - Y)$
Exponential function 3: $\exp(X + \frac{Y}{5})$
Hybrid function 1: $X \cdot \sin(X) + Y \cdot \sin(Y) + X^2 + Y^2$
Hybrid function 2: $X^2 \cdot Y + X \cdot Y^2 + \sin(X) \cdot \cos(Y)$

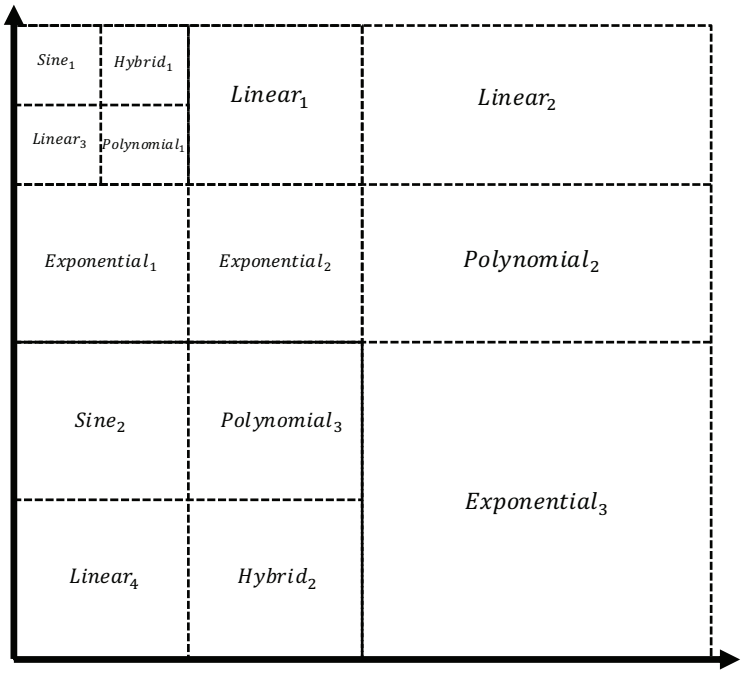


Figure A.1: Piecewise Functions 1

Piecewise Function 2 The second piecewise function is depicted in Figure A.2 and each function is described below:

Linear function 1: $6 \cdot X + 2 \cdot Y$

Linear function 2: $3 \cdot X + 5 \cdot Y$

Linear function 3: $2 \cdot X - 8 \cdot Y$

Polynomial function 1: $3 \cdot X^2 + 2 \cdot Y^2$

Polynomial function 2: $5 \cdot X^3 + 10 \cdot Y$

Polynomial function 3: $2 \cdot X^2 - 5 \cdot Y^2$

Polynomial function 4: $-5 \cdot X^2 + 3 \cdot Y^2$

Sine function 1: $2 \cdot \sin(X) + 3 \cdot \sin(Y)$

Exponential function 1: $\exp(\frac{X}{5} + \frac{Y}{2})$

Exponential function 2: $\exp(2 \cdot X - Y)$

Exponential function 3: $\exp(2 \cdot X + \frac{Y}{5})$

Hybrid function 1: $\exp(X) \cdot \frac{\sin(X)}{3} + Y \cdot \sin(Y) + X \cdot Y$

Hybrid function 2: $-X \cdot Y + \exp(X) \cdot Y^2 + \sin(X) \cdot \cos(Y)$

Hybrid function 3: $X^2 \cdot Y^2 + \frac{X}{Y} + X + Y^2$

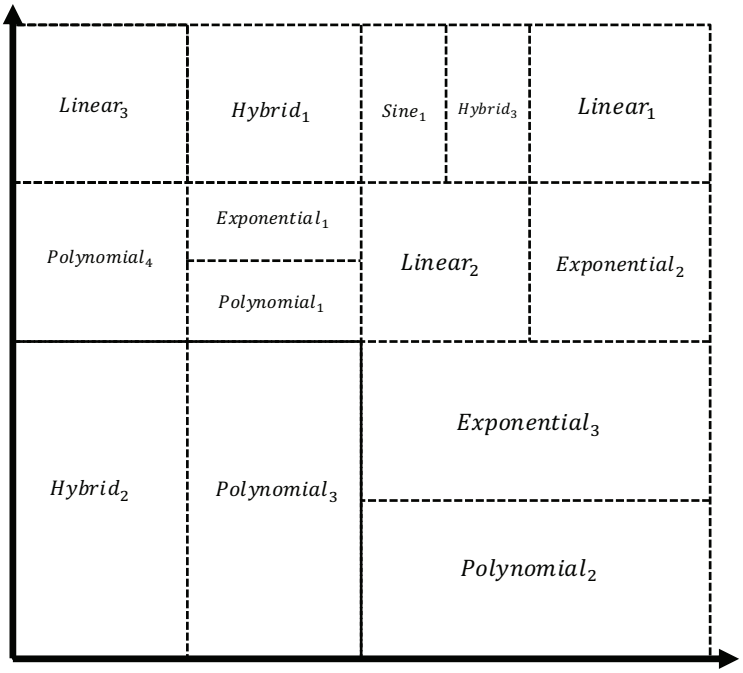


Figure A.2: Piecewise Functions 2