Streaming Model Selection via Online Factorized Asymptotic Bayesian Inference

Chunchen Liu *, Lu Feng *
NEC Laboratories China
Beijing
liu_chunchen@nec.cn, feng_lu@nec.cn

Ryohei Fujimaki
NEC Knowledge Discovery Research Laboratories,
Cupertino, CA
rfujimaki@nec-labs.com

***Abstract*—Recent growing needs for real time data analytics have increased importance of streaming model selection. Real-world streaming observations are often obtained by dynamically-changing or heterogeneous data sources, and learning machines must identify the complexities of the data generation processes on the fly without prior knowledge. This paper proposes online FAB (OFAB) inference as a general framework for streaming model selection of latent variable models. The key idea in OFAB inference is degeneration, i.e. it intentionally considers a "redundant" latent space and dynamically derives a "non-redundant" latent sub-space using a FAB-unique shrinkage mechanism on demand. By integrating the idea of stochastic variational inference, OFAB automatically and dynamically selects the best dimensionality of latent variables in a streaming and Bayesian principled manner. Empirical results on two applications, density estimation and abnormal detection, show that online FAB (OFAB) outperformed the state-of-the-art online inference methods.**

I. INTRODUCTION

Recent growing needs for real time data analytics have increased importance of streaming learning machines. For example, in Internet of Things (IoT), infinite amount of sensor streaming data might be too huge to be uploaded to cloud. Thus learning as well as prediction has to be processed on the "edges" where the amount of memory and storage are quite limited. The streaming observations from the sensors are often governed by a variety of data generation processes due to deployed environment, type of sensors, non-stationary changes in data over time, etc. Further, it is often hard in the above scenarios to obtain supervised signals in real time. Due to these natures, unsupervised learning machines must identify the complexities of the data generation processes on the fly without prior knowledge.

Online learning of unsupervised learning machines, particularly latent variable models, have been actively studied for a past decade [1], [2], [3], [4], [5], [6]. Particularly, stochastic variational inference (SVI) [7] and related work [8], [9], [10], [11] have recently received significant attentions for a wide class of latent variable models. The SVI methods incorporate ideas in stochastic optimization [12] and natural gradient [13] with variational inference, and their estimates of the gradient of evidence lower bounds (ELBOs) by subsampling the data can be cheaply obtained and are unbiased under mild conditions. With a combination of Bayesian Non-Parametric models (BNPs), SVI enables us to determine model complexity (i.e. dimensionality of latent variables) as well as model parameters in a streaming fashion. However, Bayesian non-parametric priors tend to include superfluous latent variables to achieve a good estimate of the density [14], [15] and thus usually overestimate the dimensionality of latent variables as we will demonstrate in our experiments.

Meanwhile, factorized asymptotic Bayesian (FAB) inference has recently been proposed for model selection of various of latent variable models [16], [17], [18], [19], [20], [21], and its model selection has been empirically proven to outperform BNPs with much less computational cost. A unique nature of the FAB inference is a L0-regularization effect on latent variables induced from Bayesian marginalization, which automatically eliminates irrelevant latent variables through EM-like alternating optimization (so-called *shrinkage*). In contrast to BNPs which enables us to control model complexity by prior representation, the FAB inference is better at automation since its model selection is purely induced by Bayesian marginalization regardless of prior. Although FAB inference is promising for streaming model selection, previous FAB methods are batch learning algorithms and are not applicable to steaming scenarios. Further, the FAB regularization eliminates irrelevant latent variables but there is no principal mechanism to *generate* latent variables (e.g. a new topic appears over time in non-stationary topic clustering).

This paper proposes online FAB (OFAB) inference as a general framework for streaming model selection of latent variable models. Our key ideas and contributions that address the above issues of FAB inference are summarized as follows. First, we combine an idea of streaming trust-region SVI (strust-SVI) [11] with FAB inference and derive streaming update formula of OFAB inference. As with strust-SVI, OFAB inference alternatingly scratches a sample subset from data stream and performs trust-region updates of model parameters based on the captured subset via alternating co-ordinate ascent. In addition, in order to address the issue of "generating" latent variables, we employ a latest theoretical notion, *degeneration* on FAB inference introduced by Hayashi et al. [22]. Suppose we have two latent variables Z and Z' with $\dim Z > \dim Z'$ where $\dim Z$ is the dimensionality of Z. Roughly speaking, the idea of *degeneration* is that the values of marginal log-likelihood under Z and Z' are equivalent

* Equal contribution.

if there exists a projector Π such that $Z' = \Pi Z$, and the *shrinkage* of FAB inference finds such Z' from Z (and Π is chosen to minimize the dimensionality of Z'.). On the basis of *degeneration*, our OFAB inference learns Z with sufficiently large dimensionality in a streaming fashion (here we do not perform the *shrinkage* operation) and then returns $Z' = \Pi Z$ on the fly by performing the *shrinkage* operation while keeping Z in the back. This procedure enables dynamic adjustment of the realized dimensionality of Z' and realizes fully-streaming model selection of latent variable models. We derive OFAB inference for two basic models, Gaussian mixture models (GMMs) and mixtures of probabilistic principal component analyzers (MPPCA) [23], in applications to density estimation and outlier detection. Experimental results show the superiority of OFAB over state-of-the-art online inference methods.

II. Related Work

SVI [7] has been proposed for (batch) stochastic inference of latent variable models by incorporating the idea of stochastic optimization [12] and natural gradient [13] in [1], and SVI has been extended in various aspects such as combinations with BNPs [8], [24], adaptive learning rate [25], combinations with collapsed VB [26], etc. For streaming model selection, SVI with BNPs offer efficient online learning algorithms with model selection ability [3], [5], [8], [9]. However, it has been criticized due to its dependence on the learning rate and a pre-knowledge of a fixed data size. To address the issue, recently, the SVI has been improved by replacing the natural gradient steps with trust-region updates, which allows to initialize parameters arbitrarily in each iteration, and help to escape local optima [11]. Although this strategy is particularly suitable in the streaming setting, it hasn't been directly derived to BNPs.

Besides those stochastic variational inference (SVI) style methods, Lin [27] developed the sequential variational approximation (SVA) inference for DP mixture models, which has been generalized to a broader class of BNPs recently [28]. Streaming variational Bayesian (SVB) [6] handles unbounded data by exploiting the sequential nature of Bayes theorem to recursively update an approximation of the posterior using the previous posterior as a prior. Campbell et al. [10] extended SVB to BNPs and more importantly they proposed a computation framework for streaming, distributed, asynchronous data clustering.

Factorized asymptotic Bayesian inference has been proposed in [16] as a model selection framework and has been extended to various latent variables such as hidden Markov models [17], latent features models [18], hierarchical mixture of experts [19], binary matrix factorization [20], and principal component analysis [22]. They have shown the superiority of FAB inference w.r.t. model selection over BNP-based methods. Recently, Liu et al. [20] combines SVI with FAB for scalable model selection on very large network data. However, their methods are essentially batch-learning algorithms, and as far as we know there has been no work extending FAB inference for streaming scenarios.

III. Preliminary: FAB Inference and Degenration

Suppose we have observation data $X^N = x_1, \cdots, x_N$ where $x_n \in R^D$. Let us denote latent variables corresponding to X^N as $Z^N = z_1, \cdots, z_N$, where $z_n \in \{0,1\}^K$ is an indicator vector with $z_{nk} = 1$ denotes x_n is generated by the k-th latent state[1]. FAB inference considers the following alternative representation of the marginal log-likelihood,

$$\log p(X^N|K) = \max_q \left\{ \sum_{Z^N} q(Z^N) \log \frac{p(X^N, Z^N|K)}{q(Z^N)} \right\}, \tag{1}$$

$$p(X^N, Z^N|K) = \int p(X^N, Z^N|\Theta) p(\Theta|K) d\Theta, \tag{2}$$

where $q(Z^N)$ is a variational distribution on Z^N, K denotes the model complexity, and Θ is a model parameter, respectively. By adopting a prior of $\log p(\Theta|K) = \mathcal{O}(1)$, and by applying the Laplace method to $p(X^N, Z^N|K)$, we obtain an asymptotic approximation of (1) as follows[2]:

$$FIC(\Theta|X^N) \equiv \max_q \left\{ E_q \left(\log p(X^N, Z^N|\Theta) - \underline{\underline{\sum_{k=1}^{K} \frac{D_k}{2} \log \sum_{n=1}^{N} z_{nk}}} \right) - \frac{D_\alpha}{2} \log N + H(q) \right\}, \tag{3}$$

where D_α and D_k are the parameter dimensionality of $p(Z^N)$ and $p_k(X^N|Z_k^N)$, respectively, and $H(q)$ is the entropy of $q(Z^N)$. $FIC(\Theta)$ is referred to as *factorized information criterion* (FIC), and FAB inference optimizes q and Θ by EM-like alternating updates. FIC has been proven to be asymptotically accurate, i.e. with N goes infinity,

$$\log p(X^N|K) = FIC(\Theta|X^N) + O(1) \tag{4}$$

holds.

Hayashi et al. [22] have introduced the notion of *degeneration* that plays a central role in our idea of online FAB inference. Suppose we have two tuples: (Θ_K, K, Z_K) and $(\Theta_{K'}, K', Z_{K'})$ where $K' < K$. Here we denote Θ and Z given K by Θ_K and Z_K, respectively. Then, the model $p(X, Z_K|\Theta_K, K)$ is *degenerated* if there exists an equivalent likelihood as follows:

$$p(X^N, Z_K|\Theta_K, K) = p(X^N, Z_{K'}|\Theta_{K'}, K'). \tag{5}$$

One of key observations in [22] is that, if there exists a continuous onto mapping $(Z_K, \Theta_K) \to (Z_{K'}, \Theta_{K'})$ that satisfies (5) and $Z_{K'}$ is not degenerated[3], then

$$\log p(X^N|K) = \log p(X^N|K') + O(1) \tag{6}$$

[1]Here we consider multinomial latent variables but it is easy to extend it for factorial latent variables [18], [20] and continuous latent variables [22].

[2]The *generalized FIC* which is asymptotically more accurate than the standard FIC has been proposed by [22], which replace $log \sum_{n=1}^{N} z_{nk}$ by logarithmic determinant of Fisher information matrix.

[3]For more detailed theoretical conditions, see [22].

holds and, importantly, such mappings exist for many latent variable models like Gaussian mixture models (GMMs), principal component analysis (PCA), etc. For example, it may correspond to merge two identical mixed components for GMMs or it may correspond to eliminate singular eigen subspaces for PCA.

IV. Online FAB Inference

A. The Concept

The idea of degeneration states that FAB inference enables identification of the smallest (i.e. non-degenerated) subspace of latent variables for given data X from the degenerated space. In other words, if we have sufficiently large (degenerated) space of latent variables, we can recover a non-degenerated small subspace anytime. The key idea in online FAB inference is to update "sufficiently large" latent variables in a streaming fashion and identify dynamically changing model complexity on demand. Fig. 1 illustrates this key concept of online FAB inference. Let b denotes the mini-batch index and suppose we have the *master* latent variable $Z^{(b)}$ and the *master* parameter $\Theta^{(b)}$ where the dimensionality of $Z^{(b)}$ is set to be a sufficiently large fixed number K in advance. Further, suppose $X^{(b)} = x_{(b-1)T+1}, \ldots, x_{bT}$ is a streaming mini-batch observation where T is the batch size. For each $X^{(b)}$, we iteratively update $Z^{(b)}$ and $\Theta^{(b)}$. The key idea is that K is sufficiently large such that $Z^{(b)}$ is always degenerated (a practical setting of K under resource restriction will be discussed in the later sub-section) though the effective dimensionality given $X^{(1)}, \ldots X^{(b)}$ may change over b. Then, on the basis of requests from users, we duplicate $Z^{(b)}$ and $\Theta^{(b)}$ as $\hat{Z}^{(b)}$, and $\hat{\Theta}^{(b)}$ and perform the FAB shrinkage operation to $\hat{Z}^{(b)}$ and $\hat{\Theta}^{(b)}$ in order to find the non-degenerated latent variables and their corresponding parameters.

B. Online FAB Algorithm

The derivation of the online FAB algorithm has the following two key processes:

1) streaming updates of Θ and $q(Z)$,
2) streaming shrinkage operation.

Streaming FAB EM Updates: In the stochastic updates, we extend the idea of SVI [7], particularly the recently proposed strust-SVI [11], for the FAB inference. The (strust-)SVI stochastically approximates the ELBO (a.k.a. variational free energy) by replacing expectation over variational distributions on Θ and Z by sub-sampling. Such noisy estimates of a gradient not only are cheaper to compute than true gradient, but also empower an inference algorithm with sequential nature. Instead of the ELBO, our FAB inference considers the following stochastic approximation of FIC on the mini-batch $X^{(b)}$:

$$\mathcal{FIC}(\Theta,q(Z^{(b)})|X^{(b)}) = E_q\Bigg[\frac{N_{b+1}}{T}\sum_{i=1}^{T}\log\frac{p(x_{N_b+i},z_{N_b+i}|\Theta)}{q(z_{N_b+i})} \underline{\underline{- \sum_{k=1}^{K}\frac{D_k}{2}\log\frac{N_{b+1}}{T}\sum_{i=1}^{T}z_{N_b+i,k}}}\Bigg] - \frac{D_\alpha}{2}\log N_{b+1}. \tag{7}$$

where

$$N_b = (b-1)T. \tag{8}$$

It is easy to prove that (7) is an unbiased estimator of (3), as is the case of SVI. Then, the online FAB E-step updates $q(Z^{(b)})$ by:

$$\arg\max_{q(Z^{(b)})} \mathcal{FIC}(\Theta^{(b)}, q(Z^{(b)})|X^{(b)}) \tag{9}$$

The idea of trust-region updates in SVI [11] is to regularize the ELBO by the Kullback-Leibler (KL) divergence which prevents the joint distribution of model parameters from changing so much comparing with the last update, and such that trust region updates together with the arbitrarily initialization of mini-batch samples will help to make the learning algorithm more robust (less sensitive to initialization). More specifically, the online FAB M-step using the trust-region update is given by:

$$\arg\max_{\Theta} \mathcal{FIC}(\Theta, q^{(b)}(Z^{(b)})|X^{(b)}) - \varepsilon_b \mathcal{KL}(\Theta^{(b-1)}, \Theta), \tag{10}$$

$$\mathcal{KL}(\Theta^{(b-1)}, \Theta) = \frac{N_{b+1}}{T} E_p\left(\log\frac{p(X^{(b)}, Z^{(b)}|\Theta^{(b-1)})}{p(X^{(b)}, Z^{(b)}|\Theta)}\right), \tag{11}$$

where ε_b is a learning rate which is chosen such that $\sum_b (1+\varepsilon_b)^{-1} = \infty$ and $\sum_b (1+\varepsilon_b)^{-2} < \infty$ are satisfied. We set the dynamic learning rate for the b^{th} mini-batch by observed sample number N_{b+1} and two hyperparameters (τ, κ), which is:

$$\varepsilon_b = (\tau + N_{b+1})^{\kappa} - 1. \tag{12}$$

Streaming Shrinkage Operation: One of unique natures of the FAB inference is its regularization, caused by the double-underlined parts in (3) and (7), which automatically eliminates irrelevant and redundant latent variables as we will also see in the next subsection. Previous studies refer to the effect as *model shrinkage*. Roughly speaking, the smaller and more poorly-fitted latent variables are, the more strongly they will be regularized, and thus such latent variables are gradually diminished from the model through the EM updates (please see [16], [22] for details of the shrinkage mechanism).

In contrast to the other batch FAB methods which operate the shrinkage operation directly on Z and Θ, OFAB first duplicates Z and Θ as $\hat{Z}$ and $\hat{\Theta}$, and then performs the shrinkage operation on the duplicated model so that the duplicated one comes to be non-degenerated (while the master

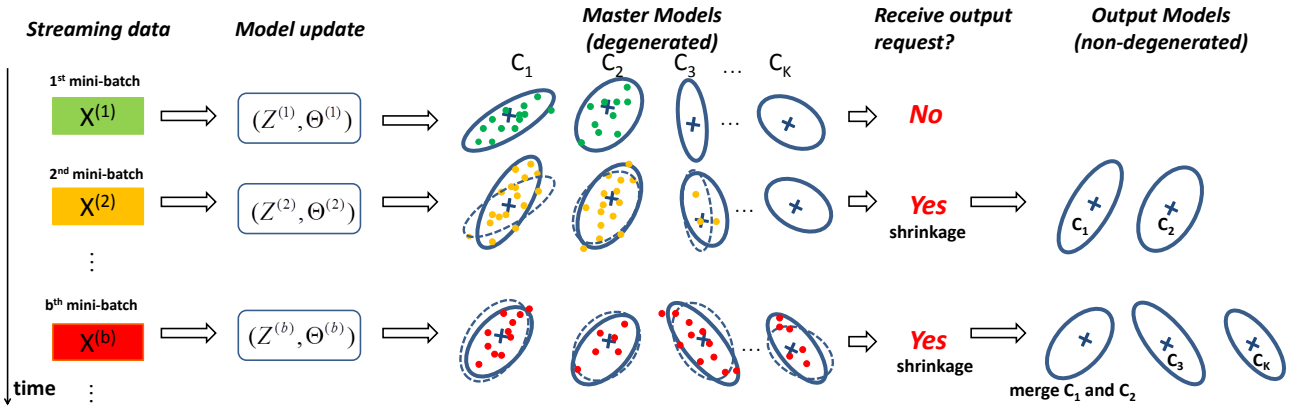

Fig. 1. One pass through the streaming data for online FAB inference, where C_K denotes the K^{th} component.

model is kept degenerated.) In practical implementation, the shrinkage operation eliminates "small" latent variables which satisfy:

$$Q_k^{(b-1)} + \sum_{i=1}^{T} q^{(b)}(z_{N_b+i,k}) < N_{b+1}\delta, \tag{13}$$

$$Q_k^{(b)} = \sum_{b'=1}^{b}\sum_{i=1}^{T} q^{(b')}(z_{N_{b'}+i,k}), \tag{14}$$

where δ is pre-fixed shrinkage parameter, and $Q_k^{(b)}$ is a scalar value and kept in memory as one of sufficient statistics. As [16] pointed, FAB inference theoretically tends to keep similar latent states (i.e. $p(x|\Theta_k)$ and $p(x|\Theta_{k'})$ resemble with each other). To obtain the model which maximizes FIC, we apply the same merge heuristics proposed in [18], which sequentially chooses two latent variables whose KL-divergence are the smallest and merges them if FIC increases by the merge.

C. Practical Implementation: Dynamic Memory Adjustment

One of important issues in OFAB is the choice of K, i.e. the dimensionality of the degenerated master latent variables. In theory, it can be any sufficiently large number. However in practice finding such a number might not be trivial particularly in situations explained in Section I, e.g. different sensors are governed by different data generation process. Suppose we have two data streams and set $K = 100$. If the true dimensionality of one stream is 10, then 100 might be too large in terms of memory consumption (memory is often limited on IoT edges). On the other hand, if that of another stream is 200, 100 is too small and we might fail to learn the data generation process.

In order to mitigate such situations and to eliminate pre-determination of K, we introduce a heuristics which dynamically expands dimensionality of latent variable *in memory*. As we have explained above, we assume that K is a very big number which exists only theoretically. Let us introduce two new notions, implemented dimensionality and realized dimensionality. The implemented dimensionality, denoted by $\bar{K}^{(b)} < K$, is dimensionality of the latent variables in memory with which the model is still degenerated. If we apply the shrinkage operation to the model with $\bar{K}^{(b)}$ latent variables, we obtain the non-degenerated model with $\hat{K}^{(b)} < \bar{K}^{(b)}$ latent variables, and we refer to $\hat{K}^{(b)}$ as the realized dimensionality.

As we observe more streaming data, $\hat{K}^{(b)}$ might approach to $\bar{K}^{(b)}$. If $\hat{K}^{(b)}$ becomes larger than $\bar{K}^{(b)} - \bar{K}_\epsilon$, we increase the implemented dimensionality by $\bar{K}^{(b)} + \bar{K}_i$ and reallocate the model in memory. This procedure guarantees that the implemented model always has more than $\bar{K}_i$ redundant latent variables and is degenerated. By this way and initializing $\hat{K}^{(1)} = 1$ and $\bar{K}^{(1)} = \bar{K}_i + 1$, we can eliminate K and dynamically adjust appropriate size of degenerated latent variables. Although we here introduce two new parameters $\bar{K}_\epsilon$ and $\bar{K}_i$, they are not sensitive and it's easy to determine them[4].

D. Examples: GMM and MPPCA

This section provides update equations for two standard models, GMMs and MPPCAs. For each mini-batch streaming observation, we iterate the OFAB EM steps for M times in practice where M is a small number. Below, Let $m \in \{1, \ldots, M\}$ denotes the index for the inner EM iterations.

1) Gaussian Mixture Models: Let $\Theta_k = \{\mu_k, \Sigma_k\}$ and $\alpha_k \in [0, 1]$ denote the parameters and the mixing ratio of the k-th component respectively, and let $\Theta = \{\Theta_1, \ldots, \Theta_K, \alpha = (\alpha_1, \ldots, \alpha_K)\}$. Then the joint distribution of GMMs over $X^{(b)}$ and $Z^{(b)}$ is described as follows:

$$p(X^{(b)}, Z^{(b)}|\Theta) = \prod_{i=1}^{T}\prod_{k=1}^{K} (\alpha_k \mathcal{N}(x_i|\mu_k, \Sigma_k))^{z_{ik}} \tag{15}$$

where $\mu_k \in \mathbb{R}^D$ and $\Sigma_k \in \mathbb{R}^{D\times D}$ are the mean vector and covariance matrix of a Gaussian distribution.

By introducing the above joint likelihood into the stochastic FIC (7) and solving the optimization problem in (9), we obtain

[4]As long as the implemented model is reasonably degenerated, the final results are not significantly affected.

the formula of E-step updates for GMMs on the mini-batch $\mathbf{X}^{(b)}$ as follows:

$$q^{(b,m)}(z_{N_b+i,k}) \propto \alpha_k^{(b,m)} p(x_{N_b+i}|\Theta_k^{(b,m)}) \underline{\underline{\exp\left(-\frac{D_k}{2Q_k^{(b,m-1)}}\right)}}, \tag{16}$$

$$Q_k^{(b,m-1)} = Q_k^{(b-1)} + \sum_{i=1}^{T} q^{(b,m-1)}(z_{N_b+i,k}), \tag{17}$$

where D_k represents the parameter dimensionality of $\Theta_k^{(b)}$.

The unique regularization term of FAB double-underlined in (3) and (7) turns to $\exp(-D_k/2Q_k^{(b,m-1)})$ in (16) when updating the latent variable distribution $q(z_{N_b+i,k})$. Intuitively speaking, the smaller a component is (the smaller $Q_k^{(b,m-1)}$ is), the less possible samples will be assigned to this component (the smaller $q^{(b,m)}(z_{N_b+i,k})$ is), and the more likely it is to be shrunk. Additionally, as the cumulative number of samples $N_{b+1} \to \infty$, the regularization term approaches 1, resulting in the attenuation of its regularizing effect. In practice for streaming data, the less samples following a latent state, the higher possibility that latent state is out-of-date or redundant, then it should be eliminated.

The OFAB M-step using the trust-region update in (10) has been derived for GMMs as follows:

$$\alpha_k^{(b,m)} = \frac{\sum_{i=1}^{T} q^{(b,m-1)}(z_{N_b+i,k}) + \varepsilon_b T \alpha_k^{(b,m-1)}}{(1+\varepsilon_b)T}, \tag{18}$$

$$\mu_k^{(b,m)} = \frac{\sum_{i=1}^{T} q^{(b,m-1)}(z_{N_b+i,k}) \mathbf{x}_{N_b+i}}{(1+\varepsilon_b)T\alpha_k^{(b,m-1)}} + \frac{\varepsilon_b \mu_k^{(b,m-1)}}{1+\varepsilon_b}, \tag{19}$$

$$\Sigma_k^{(b,m)} = \frac{\sum_{i=1}^{T} q^{(b,m-1)}(z_{N_b+i,k}) \left\langle \mathbf{x}_{N_b+i}, \mu_k^{(b,m)} \right\rangle}{(1+\varepsilon_b)T\alpha_k^{(b,m-1)}} + \frac{\varepsilon_b}{1+\varepsilon_b}\left(\left\langle \mu_k^{(b,m-1)}, \mu_k^{(b,m)} \right\rangle + \Sigma_k^{(b,m-1)} \right), \tag{20}$$

where we denote $(\mathbf{a}-\mathbf{b})(\mathbf{a}-\mathbf{b})^T$ as $\langle \mathbf{a}, \mathbf{b} \rangle$.

Algorithm 1 summarizes OFAB for GMMs with the heuristics explained in the previous subsection. For notational simplicity, we denote $q^{(b)} = q^{(b,M)}$ and the same rule is applied to $\alpha^{(b)}$, $\mu_k^{(b)}$ and $\Sigma_k^{(b)}$. Also, in line 9, $\Theta^{(b)}$ is denoted by:

$$\Theta^{(b)} = (\alpha^{(b)}, \Theta_1^{(b)}, \ldots, \Theta_{\bar{K}^{(b)}}^{(b)}). \tag{21}$$

It is worth noting that the dimensionality of $\hat{\Theta}$ changes from $\bar{K}^{(b)}$ in line 9 to $\hat{K}^{(b)}$ in line 10, and the dimensionality of the output model dynamically changes over the mini-batch iteration. Let us note a few points on hyper-parameter settings. For updating the learning rate in line 3, we fixed $\tau = 1.0$ and $\kappa = 0.5$ which stably worked. δ is not necessarily determined in advance and users can change it when they request models though we fixed it to $\delta = 0.001$ in our experiments for simplicity. In terms of the number of inner EM iterations, M, a small value performed better in our investigation from both viewpoints of computational efficiency and overfitting.

Algorithm 1 Online FAB for GMMs

Input:
Stream Mini-batch Data $\{X^{(b)}\}_{b=1}^{\infty}$;
Hyperparameters $\tau = 1.0$, $\kappa = 0.5$, $\delta = 0.001$, $\bar{K}_\epsilon = \bar{K}_i = 2$, $M = 10$.

Output: Model parameters α and Θ_k for $\forall k$.

1: Randomly initialize parameters with $\bar{K}_i + 1$ latent variables.
2: **for** $b = 1, 2, 3, \cdots$ **do**
3: Randomly initialize $q^{(b,0)}(Z^{(b)})$ and update ε_b by (12)
4: **for** $m = 1, \cdots, M$ **do**
5: Update $(\alpha^{(b,m)}, \mu_k^{(b,m)}, \Sigma_k^{(b,m)})$ by (18)-(20).
6: Update $q^{(b,m)}(z_{N_b+n,k})$ by (16).
7: **end for**
8: **if** Receive a call for exporting the model **then**
9: Copy $\Theta^{(b)}$ to $\hat{\Theta}$ by (21), and apply the shrinkage to $\hat{\Theta}$ by (13)
10: **yield** $\hat{\Theta} = (\hat{\alpha}, \hat{\Theta}_1, \ldots, \hat{\Theta}_{\hat{K}^{(b)}})$.
11: **end if**
12: Update $\hat{K}^{(b)}$ (the number of k that don't satisfy (13)).
13: **if** $\hat{K}^{(b)} \geq \bar{K}^{(b)} - \bar{K}_\epsilon$ **then**
14: Randomly initialize $\alpha_{\bar{K}^{(b)}+1}, \cdots, \alpha_{\bar{K}^{(b)}+\bar{K}_i}$ and $\Theta_{\bar{K}^{(b)}+1}, \cdots, \Theta_{\bar{K}^{(b)}+\bar{K}_i}$ and add them to $\Theta^{(b)}$.
15: **end if**
16: **end for**

2) Mixtures of Probabilistic PCA: MPPCA differs from GMM mainly on its decomposition of the full covariance matrix $\Sigma_k = W_k W_k^T + \sigma_k^2 I$ and offers a way to control the model complexity. Let $\Theta_k = \{\mu_k, W_k, \tilde{D}_k, \sigma_k\}$ denote the parameters of the k-th component. Then the joint likelihood over $X^{(b)}$ and $Z^{(b)}$ is given by:

$$p(X^{(b)}, Z^{(b)}|\Theta) = \prod_{i=1}^{T}\prod_{k=1}^{K} (\alpha_k \mathcal{N}(x_i|\mu_k, W_k W_k^T + \sigma_k^2 I))^{z_{ik}}, \tag{22}$$

where $\mu_k \in \mathbb{R}^D$, $W_k \in \mathbb{R}^{D\times\tilde{D}_k}$, $\tilde{D}_k \in \{1, \cdots, D\}$, and $\sigma_k \in \mathbb{R}$. It is worth noting that different components may have different values of intrinsic dimensionality $\tilde{D}_k$, and OFAB inference is able to select them automatically.

The formula of E-step updates on the mini-batch $X^{(b)}$ is

$$q(z_{N_b+i,k})^{(b,m)} \propto \alpha_k^{(b,m)} p(x_{N_b+i}|\mu_k^{(b,m)}, W_k^{(b,m)} W_k^{(b,m)T} + \sigma_k^{2(b,m)} I) \underline{\underline{\exp\left(-\frac{(D+1)^2 - (D-\tilde{D}_k^{(b,m)})^2 + \tilde{D}_k^{(b,m)} + 1}{4Q_k^{(b,m-1)}}\right)}}. \tag{23}$$

In the OFAB M-step, we first update $\alpha_k^{(b,m)}$, $\mu_k^{(b,m)}$, and $\Sigma_k^{(b,m)}$ using (18) - (20), and then decompose $\Sigma_k^{(b,m)}$ by:

$$\begin{aligned} &\Sigma_k^{(b,m)} = V_k \mathrm{Diag}(\lambda_1, ..., \lambda_D) V_k^T, \\ &\lambda_1 \geq \lambda_2 \geq ..... \geq \lambda_D, \\ &V_k = [v_{k1}, ..., v_{kD}] \in \mathbb{R}^{D\times D}, V_k^T V_k = I, \end{aligned} \tag{24}$$

and update $\tilde{D}_k^{(b,m)}$, $W_k^{(b,m)}$, and $\sigma_k^{2(b,m)}$ as follows,

$$\tilde{D}_k^{(b,m)} = \arg\max_d \left\{ -\frac{N_{b+1}(1+\varepsilon_b)\alpha_k^{(b,m)}}{2} \left(\sum_{j=1}^{d} \log \lambda_j + (D-d) \log \frac{\sum_{j=d+1}^{D} \lambda_j}{D-d} \right) - \frac{2Dd - d^2 + d}{4} \log \left(\frac{N_{b+1}}{T} \sum_{i=1}^{T} q^{(b,m-1)}(z_{N_b+i,k}) \right) \right\}, \tag{25}$$

$$\sigma_k^{2(b,m)} = \frac{\sum_{j=\tilde{D}_k^{(b,m)}+1}^{D} \lambda_j}{D - \tilde{D}_k^{(b,m)}}, \tag{26}$$

$$W_k^{(b,m)} = \left[\sqrt{\lambda_1 - \sigma_k^{2(b,m)}} v_{k1}, \ldots, \sqrt{\lambda_{\tilde{D}_k^{(b,m)}} - \sigma_k^{2(b,m)}} v_{k\tilde{D}_k^{(b,m)}} \right]. \tag{27}$$

The unique nature of (O)FAB is the selection of latent variables by the double underlined term which is yielded from the regularization term in FIC and corresponds to the double underlined term in (3). By this regularization effect, (O)FAB-MPPCA enables to select the intrinsic dimensionality for each latent variable.

V. EXPERIMENTS

We conducted experiments using two basic models, GMMs and MPPCAs, explained in Section IV-D.

A. Gaussian Mixture Model

1) Baseline Methods: We applied OFAB to GMM in application to artificial datasets, and compared four state-of-the-art methods with OFAB/GMM as baselines: (i) batch FAB inference for GMM (FAB [16]); (ii) GMM with streaming trust-region stochastic variational inference (strust-SVI [11]); (iii) streaming variational Bayesian inference for GMM (SVB [6]); (iv) DP Gaussian mixture with streaming, distributed variational Bayesian inference (SDA-DP [10]). We used the latest implementations provided by the authors for FAB, strust-SVI[5], and SDA-DP[6], and implemented SVB/GMM on our own. For model selection, strust-SVI and SVB employ an outer loop for cross validation and SDA-DP relies on its Dirichlet process prior to incorporate new components. It is worth noting that **only OFAB and SDA-DP are truly streaming model selection methods** since FAB is a batch algorithm and strust-SVI and SVB need outer-loop cross validation.

The two hyperparameters in OFAB, $\bar{K}_\epsilon$ and $\bar{K}_i$ as introduced in section IV-C, are set as $\bar{K}_\epsilon = \bar{K}_i = 2$ identically for all synthetic datasets. We used the default hyperparameters in the softwares for strust-SVI, SDA-DP, and FAB. Two hyperparameters should be adjusted for SVB, parameter of Dirichlet prior for mixing weights and parameter of normal prior for Gaussian mean value, for which we tried different combinations and selected the best one. For fair comparison, OFAB, strust-SVI, SVB, and SDA-DP all employ $M = 10$ as the maximum number of inner iteration on each streaming mini-batch. We also ran only 1 thread for SDA-DP to avoid loss of accuracy although it's a distributed one. In all the simulation experiments below, we initialized $\hat{K}^{(1)} = 1$ and $\bar{K}^{(1)} = \hat{K}^{(1)} + \bar{K}_i = 3$ for OFAB, adopted $K = 40$ for FAB, performed inference with $K = 2, ..., 40$ for strust-SVI and SVB for cross validation, and truncated mini-batch inference of SDA-DP to $K_{mini} = 20$ to balance between estimation accuracy and model selection accuracy. All of the results are average of three runs.

2) Evaluation Criteria: We compared the five methods on the basis of three evaluation metrics. The first one is the KL divergence between the true distribution and the estimated distribution for evaluating estimation error. The second one is the test log-likelihood for evaluating generalization error. The last one is $|\hat{K}^{(b)} - K^*|$ for evaluating model selection error where K^* is the true dimensionality.

3) Synthetic Dataset Generation: We generated the true models with different D (dimension), K^* (component number) and $\Theta_c^* = (\mu_c^*, \Sigma_c^*)$ (mean and covariance), where $*$ denotes the true model. The parameter values were randomly sampled as $\alpha_c^* \sim [0.4, 0.6]$ (before normalization), and $\mu_c^* \sim [-5, 5]^D$. Σ_c^* were generated as $a_c \bar{\Sigma}_c^*$, where $a_c \sim [0.5, 1.5]$ is a scale parameter and $\bar{\Sigma}_c^* \sim [0, 1]^{(D \times D)}$. Based on the true models, we generated both stationary stream (data are generated according to some stationary probability distributions) and non-stationary stream (data are generated according to drifting probability distributions) to simulate real data environments.

4) Case 1. Stationary Streaming: Assuming we have a true model with K^* components, then a stationary stream can be simulated by: (i) mixing all instances from the K^* components uniformly, and getting the sample set S; (ii) dividing S into several disjointed subsets; (iii) sending those subsets (batches) one by one to the learning framework to simulate streaming data.

We first investigated how the five methods work as streaming data arrive continuously using a true model whose $K^* = 10, D = 3, N = 50000$. When simulating the data stream, we set the size of each coming mini-batch as $T = 10$, and there are $N/T = 5000$ batches in total. Fig. 2 summarizes the comparisons on KL-divergence, test log-likelihood and component number. From these results, we observed:

- For KL-divergence (**A**), OFAB outperformed the others identically along the stream arriving process. Further, the convergence of OFAB was much faster than the others.
- For test log-likelihood, the performances of OFAB and strust-SVI were similar and significantly better than the others. Similar to KL-divergence, the convergence of OFAB was faster than strust-SVI.
- For model selection along with mini-batches, SDA-DP performed better than OFAB which required sufficient number of mini-batches to appropriately perform its shrinkage mechanism. However, as data increased, SDA-DP overestimated

[5]https://github.com/lucastheis/trmix

[6]https://github.com/trevorcampbell/sda-bnp

K as previous study theoretically shown [14]. On the other hands, OFAB came to stably and well estimate K^*.

- Although the performance of strust-SVI was comparative to OFAB, it needs outer-loop cross validations for model selection, and thus is unable to adjust the learnt structure as data streaming in incessantly.
- For KL-divergence and test log-likelihood, the performance of OFAB rapidly approached those of the batch FAB. It was rather surprising that OFAB achieved similar level of errors with batch FAB that repeatedly revisit large amount of samples, by looking at much less samples only once.

Next we evaluated the five methods further on datasets with different model complexity (K*=10,30), different feature scale (D=3,10,20), and different batch sizes ($T = 10, 50$) to investigate detailed behaviors and sensitivity of each method. Table I summarizes their performance on KL-divergence and component number on the last mini-batch $b = 5000$. From the results, we observed:

- Comparing two truly-streaming methods, OFAB and SDA-DP, OFAB performed better in KL-divergence for 6 settings out of 8. For model selection, SDA-DP significantly over-estimated the dimensionality in many settings and failed to select appropriate models though OFAB chose very close dimensionality to K^*.
- Comparing OFAB with strust-SVI and SVB, OFAB performed much better in KL-divergence for all settings. For model selection, with $K^* = 10$, both methods performed comparatively, but OFAB outperformed strust-SVI with $K^* = 30$.
- The batch FAB performed the best among all because it can finely tune the model by repeatedly visiting all samples. It is worth noting that OFAB performed very closely to the batch FAB and we confirmed the strong stream model selection ability of OFAB.

5) Case 2. Non-stationary Streaming: We next evaluate OFAB in cases where the stream is non-stationary. For a true model with K^* components, (i) divide the K^* components into several groups with or without overlap among them, and get component group set $S^k = \{S_1^k, S_2^k, \ldots\}$; (ii) for each component group S_i^k, mix instances from its components uniformly, to get its sample set S_i^s; (iii) divide S_i^s into disjointed groups $S_i^s = \{S_{i,1}^s, S_{i,2}^s, \ldots\}$; (iv) combine sample sets in the sequence of i increasing, and get $S = \{\ldots, S_{i,1}^s, S_{i,2}^s, \ldots, S_{i+1,1}^s, S_{i+2,2}^s, \ldots\}$; (v) send sample sets in S in sequence to the learning framework.

The configuration of the true model was $K^* = 10$, $D = 3$, $N = 5000$, and $T = 10$. Let us denote the j-th components in the true model by Kj. To simulate the drifting stream, data were divided into 500 mini-batches, where the first 150 batches comprised samples following distributions of components $K1 \sim K3$, batches $151 \sim 295$ contained samples belonging to components $K4 \sim K6$ only, and batches $296 \sim 500$ had samples from components $K7 \sim K10$ which is shown in Fig. 3-**B**.

Fig. 3 shows comparisons w.r.t. prediction and model selection (we here exclude KL-divergence because it is very similar to the results w.r.t. test-loglikelihood). We can see from (**A**) that sharp inflexions arose on batch 151 and batch 296 for all of the four methods. And these two mini-batches were just the points where samples following old distributions disappeared and the ones following new patterns arose. After these inflexion points, OFAB converged much faster than all the other methods, meaning that OFAB captured the drifting data generation processes the most quickly. That's because i) OFAB needs less data for convergence, and ii) the mechanism of OFAB for increasing latent variable dimensionality makes it catch features of fresh patterns timely. After stepping into stable convergence, OFAB and strust-SVI shared similar predictive accuracy, while SVB and SDA-DP performed inferiorly. (**B**) shows how the different online methods work when learning model structures changing over time. For strust-SVI/GMM and SVB/GMM, because of their lacking the ability to adjust model complexity in one pass which is the regular ways that stream arrive, thus they can't catch the dynamic changing model structures. For OFAB, after the inflexion points (batch $1, 151$, and 296), it gradually approached and converged to the true model structure, which benefited from: i) the shrinkage operation could automatically eliminate useless components. For the example case, as no more samples following old components arrived, the scale of these old components became smaller and smaller (Q_k got smaller), and were shrunk finally. ii) Components catching new features were added when OFAB found the implemented model lacked redundant latent variables. For SDA-DP, it looked sensitive to the emergence of new patterns because the component number learnt out increased sharply after the inflexion points, however, it obviously produced superfluous components comparing with the true model.

B. Mixtures of Probabilistic Principle Component Analyzers

We next applied OFAB to MPPCA. Firstly, we used a toy example to show how OFAB/MPPCA worked when data streamed in with drifting patterns. Then we compared OFAB/MPPCA with 1) batch FAB/MPPCA and 2) strust-SVI for MPPCA (strust-SVI/MPPCA) in terms of outlier detection on some real-world datasets. We derived trust-region stochastic variational inference to MPPCA and implemented it by ourselves.

1) Visual Demonstration: We generated a synthetic dataset comprising 2000 two-dimensional data points that compose characters 'I', 'C', 'D', 'M' with additive Gaussian noise as Fig. 4 (A) showed. To demonstrate how OFAB/MPPCA worked, we designed two streaming settings, 1) sequential pattern and 2) hybrid pattern. In the sequential pattern, the 500 points that aggregate into 'I' are divided into different batches and streamed in firstly, and then the same number of points that aggregate into 'C', 'D', 'M' streamed in sequentially. In the hybrid pattern, the points that compose 'I' and 'C' were mixed randomly and streamed in firstly, and then the points that compose 'I', 'C' and 'D' were mixed randomly and streamed in, at last the remained points that compose 'I', 'C', 'D', 'M' were mixed randomly and streamed in. Under both of the two

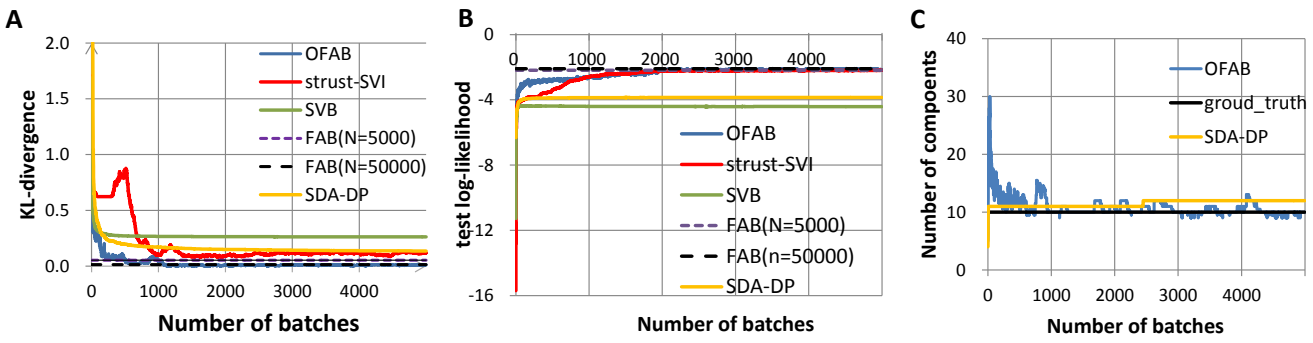

Fig. 2. Model selection accuracy comparison under stationary streaming setting. **A**, KL-divergence comparison among the five methods, where the batch FAB carries out two experiments by using $N = 5000$ and $N = 50000$ samples, respectively. **B**, test log-likelihood comparison, and we used a test set holding out a different 20% of the data to evaluate log-likelihood. **C**, the dynamic model structure estimation accuracy of OFAB along the stream arriving process.

TABLE I

COMPARISONS OF DENSITY ESTIMATION ABILITY (KL-DIVERGENCE, TOP) AND MODEL SELECTION ABILITY (BOTTOM) ON VARIOUS TYPES OF SYNTHETIC DATA.THE STANDARD DEVIATIONS WAS SHOWED IN PARENTHESES. THE BEST RESULT ON EACH DATASET WAS HIGHLIGHTED IN BOLD, AND THE SECOND BEST ONE WAS MARKED IN ITALIC, EXCLUDING THE BATCH FAB WHICH IS NOT A STREAM LEARNING ALGORITHM

K^*	D	T	FAB (batch)	OFAB	strust-SVI + CV	SVB + CV	SDA-DP
10	3	10	0.038(0.03)	**0.051(0.04)**	*0.058(0.05)*	0.291(0.21)	0.067(0.04)
10	3	50	0.038(0.03)	**0.039(0.02)**	0.112(0.09)	0.297(0.15)	*0.084(0.08)*
10	10	10	0.044(0.09)	*0.049(0.07)*	0.055(0.09)	1.816(0.82)	**0.044(0.03)**
10	10	50	0.044(0.09)	**0.047(0.09)**	0.460(0.26)	1.324(0.38)	*0.056(0.04)*
30	10	10	0.289(0.09)	*0.212(0.18)*	0.240(0.08)	1.435(0.49)	**0.112(0.04)**
30	10	50	0.289(0.09)	**0.161(0.08)**	1.367(0.21)	1.304(0.22)	*0.599(0.17)*
30	20	10	0.349(0.08)	**0.943(0.62)**	*1.663(0.28)*	3.039(0.68)	187.8(20.9)
30	20	50	0.349(0.08)	**0.396(0.20)**	2.648(0.49)	2.603(0.71)	*1.347(0.43)*

K^*	D	T	FAB (batch)	OFAB	strust-SVI + CV	SVB + CV	SDA-DP
10	3	10	10.0(2.45)	*9.3(1.18)*	**10.0(0.00)**	7.3(0.67)	11(1.67)
10	3	50	10.0(1.45)	*11.2(1.86)*	**9.2(1.03)**	7.4(1.20)	12(2.56)
10	10	10	9.9(0.35)	*10.2(1.60)*	**10.0(0.00)**	7.0(0.82)	282(99.9)
10	10	50	9.9(0.35)	**10.8(0.37)**	*9.0(0.76)*	8.83(0.69)	37(6.39)
30	10	10	29.1(1.12)	*31.6(2.57)*	**28.8(0.90)**	22.3(2.43)	653.1(300)
30	10	50	29.1(1.12)	**29.9(0.70)**	22.8(2.60)	*22.8(1.48)*	64.1(18.6)
30	20	10	30.4(1.49)	**32.7(5.18)**	*19.5(4.11)*	18.3(1.92)	6577.3(728.7)
30	20	50	30.4(1.49)	**29.4(1.42)**	*24.5(3.23)*	20.9(1.63)	474.7(67.7)

settings, in order to make the patterns visible, the batch size was set to 250 and resulted in 8 batches. An MPPCA model was fitted to the synthetic data points using OFAB inference method. At each iteration, an involving model was updated based on data points in the current batch, and then it was used to reconstruct those data points to verify our approach. In Fig. 4 (B)∼(I), we showed the reconstruction results at different iterations. The first line of Fig. 4 illustrated the data reconstruction in the sequential pattern, and the second line showed the data reconstruction in the hybrid pattern. The illustration provided some intuition that our online algorithm can flexibly adapt to drifting concepts in data stream and adjust the model structure dynamically to catch complicated data structure.

2) Outlier Detection: One of important applications of MPPCA is outlier detections where MPPCA detect data samples that have large reconstruction errors or high negative log-likelihood. Compared to GMM, MPPCA learns local low-dimensional coordinates and therefore can accurately detect outliers by taking locality of outliers into account.

We employed three open datasets from UCI machine learning repository. The first one was the log file of shuttle including 45586 normal records and 3022 anomaly records [7]. All 9 attributes were involved in our analyses. The second dataset was the log file of network connections used for KDD Cup 1999 [8]. Considering a large amount of memory overhead consumed by batch algorithm, we only extracted 20 percent of samples and 3 attributes (i.e. duration, src_bytes, dst_bytes). That is about 197K normal accesses and 2206 successful intrusions whose logged_in attributes were 1. The third dataset was composed of 2.4 million URLs and 3.2 million features [29] [9]. We extracted about 255K benign URLs and 1413 malicious URLs (spam, phishing, exploits, and so on). Originally, there were 64 real-valued features in total, and the number of features were further narrowed down to 5. Those features were discarded because they were owned by less than 10 percent of samples.

We used the same learning rate ($\tau = 1.0$ and $\kappa = 0.5$), the same batch size ($T = 10$), and the same number of

[7]https://archive.ics.uci.edu/ml/datasets/Statlog+%28Shuttle%29
[8]http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[9]http://archive.ics.uci.edu/ml/datasets/URL+Reputation

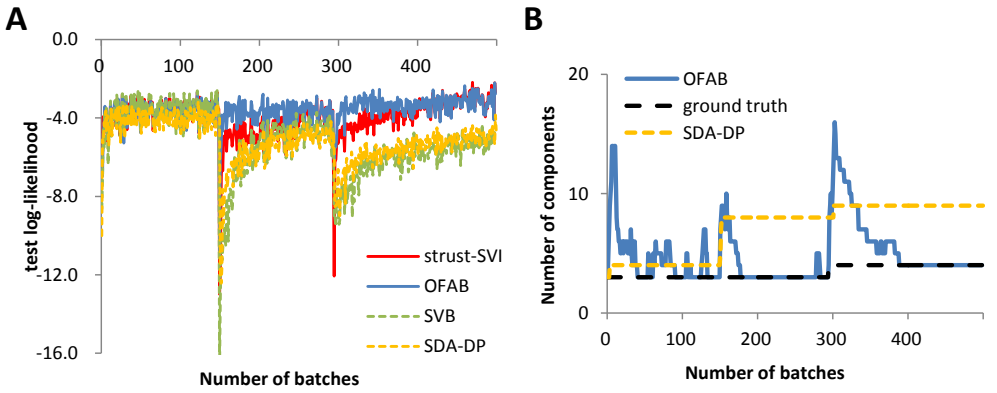

Fig. 3. Model selection accuracy comparison under non-stationary setting. **A**, test log-likelihood comparison, and we used the model at time $t-1$ to predict likelihood for the stream batch arriving at time t. **B**, the model structure estimation accuracy of OFAB along the stream arriving process.

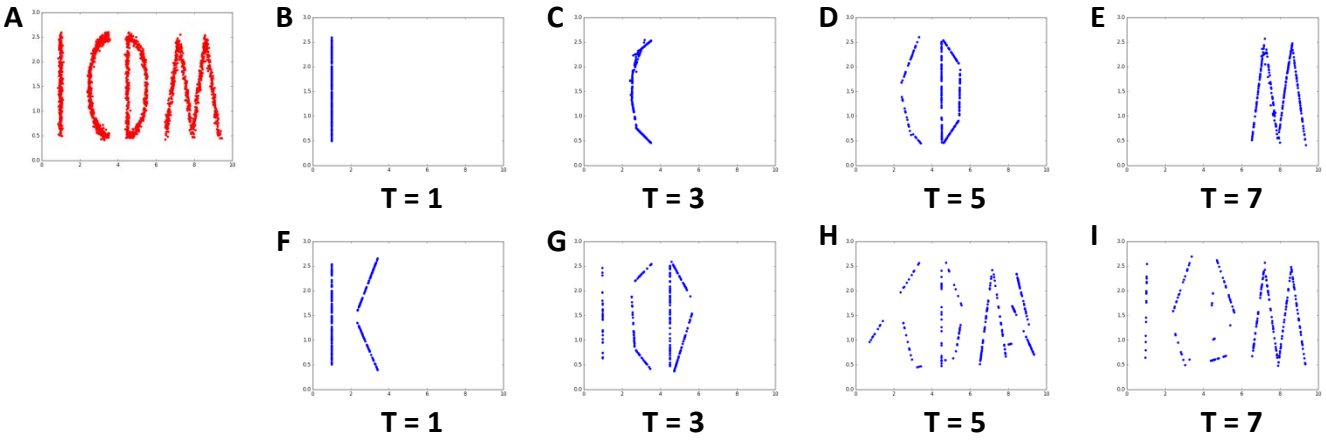

Fig. 4. Modeling noisy data that comprise characters 'I', 'C', 'D', and 'M'. (A) The synthetic data in a scatter plot. (B)∼(E) Data reconstructed from the fitted MPPCA model on different batches when data streamed in the sequential pattern. (F)∼(I) Data reconstructed from the fitted model on different batches when data streamed in the hybrid pattern.

inner iteration ($M = 10$) for both OFAB/MPPCA and strust-SVI/MPPCA in the following experiments. We gradually updated an outlier detector as soon as it streamed in, and used that detector to pick up outliers in the test set. Two metrics, precision and recall, were used to measure the performance of outlier detection, and they were plotted against the number of mini-batches as Fig. 5 showed. The accuracy of OFAB/MPPCA increased gradually and converged to that of FAB/MPPCA. By comparing among different datasets, we can see that as the scale of a dataset grew up, the speed of convergence of OFAB/MPPCA slowed down. It's easy to understand, the data patterns might grow as their quantity grew, and thus the algorithm needed more observations to completely catch the data distribution. In general, the performance of OFAB/MPPCA was much more attractive than that of strust-SVI/MPPCA.

To investigate the influence of the batch size on the performance of OFAB/MPPCA, we carried out additional experiments with three different batch sizes (i.e. $T = 10$, $T = 20$, and $T = 50$) on two open datasets. Since the number of samples N is a constant, the total number of batches N/T decreases when the batch size T becomes larger. In Fig. 6, we truncated the accuracy curves to the previous $N/50$ batches to facilitate demonstration for $T = 10$ and $T = 20$, and we found that the convergence speed of OFAB/MPPCA increased when the batch size got larger.

VI. Conclusions

This paper proposes a general framework, online factorized asymptotic Bayesian (OFAB) inference, for streaming model selection of latent variable models. The main contributions are applying degeneration for dynamic model selection which keeps a redundant latent space and degenerate it into a "non-degenerated" subspace on demand, an optimization of degeneration that starts from a simple complexity and gradually complicate it to catch new data generation processes that significantly saves memory and improve the efficiency, deriving OFAB for two basic models GMM and MPPCA in application to density estimation and outlier detection.

References

[1] M. Sato, "Online model selection based on the variational bayes," Neural Computation, 2001.

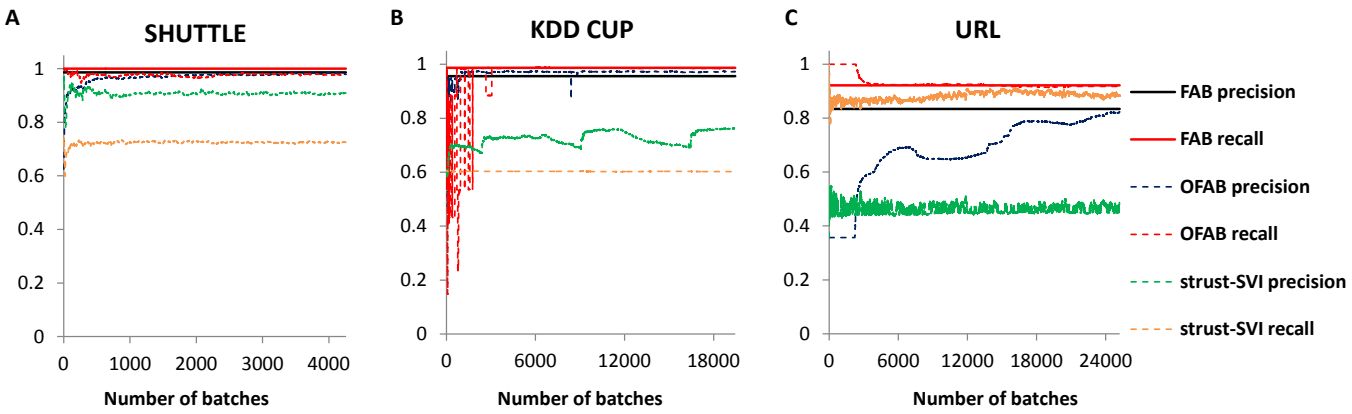

Fig. 5. Performance of outlier detection on (A) streamed logs of shuttle, (B) streamed logs of network connections, and (C) streamed URL.

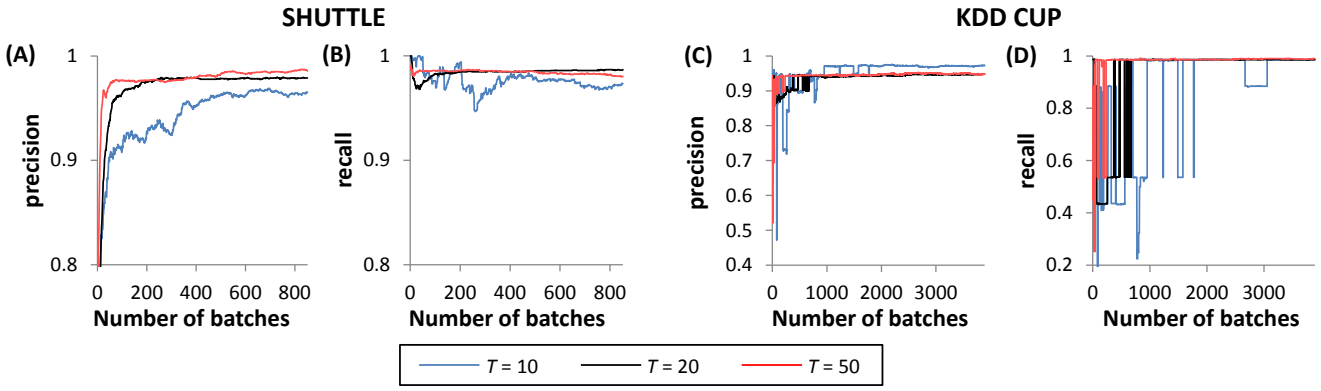

Fig. 6. Performance of OFAB/MPPCA with different batch sizes. (A) and (B) show the precision and recall on the "shuttle" dataset, (C) and (D) show the precision and recall on the "KDD CUP" dataset.

[2] M. Hoffman, D. Blei, and F. Bach, "Online learning for latent dirichlet allocation," in NIPS, 2010.

[3] C. Wang, J. Paisley, and D. Blei, "Online variational inference for the hierarchical dirichlet process," in AISTATS, 2011.

[4] R. Fujimaki, Y. Sogawa, and S. Morinaga, "Online heterogeneous mixture modeling with marginal and copula selection," in KDD, 2011.

[5] M. Bryant and E. Sudderth, "Truly nonparametric online variational inference for hierarchical dirichlet process," in NIPS, 2012.

[6] T. Broderick, N. Boyd, A. Wibisono, A. Wilson, and M. Jordan, "Streaming variational bayes," in NIPS, 2013.

[7] M. Hoffman, D. Blei, C. Wang, and J. Paisley, "Stochastic variational inference," JMLR, 2013.

[8] C. Wang and D. Blei, "Truncation-free stochastic variational inference for bayesian nonparametric models," in NIPS, 2012.

[9] M. Hughes and E. Sudderth, "Memoized online variational inference for dirichlet process mixture models," in NIPS, 2013.

[10] T. Campbell, J. Straub, J. Fisher, and J. How, "Streaming, distributed variational inference for bayesian nonparametrics," in NIPS, 2015.

[11] L. Theis and M. Hoffman, "A trust-region method for stochastic variational inference with applications to streaming data," in ICML, 2015.

[12] H. Robbins and S. Monro, "A stochastic approximation method," The Annals of Mathematical Statistics, 1951.

[13] S. Amari, "Natural gradient works efficiently in learning," Neural Computation, 1998.

[14] J. Miller and M. Harrison, "A simple example of dirichlet process mixture inconsistency for the number of components," in NIPS, 2013.

[15] ——, "Inconsistency of pitman-yor process mixtures for the number of components," JMLR, 2014.

[16] R. Fujimaki and S. Morinaga, "Factorized asymptotic bayesian inference for mixture modelling," in AISTATS, 2012.

[17] R. Fujimaki and K. Hayashi, "Factorized asymptotic bayesian hidden markov models," in ICML, 2012.

[18] K. Hayashi and R. Fujimaki, "Factorized asymptotic bayesian inference for latent feature models," in NIPS, 2013.

[19] R. Eto, R. Fujimaki, S. Morinaga, and H. Tamano, "Fully-automatic bayesian piecewise sparse linear models," in AISTATS, 2014.

[20] C. Liu, L. Feng, R. Fujimaki, and Y. Muraoka, "Scalable model selection for large-scale factorial relational models," in ICML, 2015.

[21] Y. M. J. Wang, R. Fujimaki, "Trading interpretability for accuracy: Oblique treed sparse additive models," in KDD, 2015.

[22] K. Hayashi, S. Maeda, and R. Fujimaki, "Rebuilding factorized information criterion: asymptotic accurate marginal likelihood," in ICML, 2015.

[23] M. Tipping and C. Bishop, "Mixtures of probabilistic principal component analyzers," Neural Computation, 1999.

[24] J. Hensman, N. Fusi, and N. D. Lawrence, "Gaussian processes for big data," in UAI, 2013.

[25] R. Ranganath, C. Wang, D. M. Blei, and E. P. Xing, "An adaptive learning rate for stochastic variational inference," in ICML, 2013.

[26] J. R. Foulds, L. Boyles, C. DuBois, P. Smyth, and M. Welling, "Stochastic collapsed variational bayesian inference for latent dirichlet allocation," in KDD, 2013.

[27] D. Lin, "Online learning of nonparametric mixture models via sequential variational approximation," in NIPS, 2013.

[28] A. Tank, N. J. Foti, and E. B. Fox, "Streaming variational inference for bayesian nonparametric mixture models," in AISTATS, 2015.

[29] J. Ma, L. Saul, S. Savage, and G. Voelker, "Identifying suspicious urls: an application of large-scale online learning," in ICML, 2009.