Variational Inference for Non-Stationary Distributions

by

Arsen Mamikonyan

Submitted to the Department of Electrical Engineering and Computer Science on February 7, 2017, in partial fulfillment of the requirements for the degree of Master of Engineering in Electrical Engineering and Computer Science

Abstract

In this thesis, I look at multiple Variational Inference algorithm, transform Kalman Variational Bayes and Stochastic Variational Inference into streaming algorithms and try to identify if any of them work with non-stationary distributions. I conclude that Kalman Variational Bayes can do as good as any other algorithm for stationary distributions, and tracks non-stationary distributions better than any other algorithm in question.

Thesis Supervisor: Samuel Madden
Title: Professor
Acknowledgments

I would like to thank Sam Madden and Peter Bailis for mentoring me through the work I did for my thesis. I am very grateful for your time, support and guidance.

I would like to thank Tamara Broderick for sparking my interest in Bayesian Inference, and being my academic adviser during this year.

I would like to thank my family and friends for always being there for me.
# Contents

1 Introduction 13
   1.1 Motivation ........................................ 13
   1.2 Defining the Problem .............................. 14
   1.3 Contributions ..................................... 15

2 Inference Techniques for Graphical Models 17
   2.1 Variational Inference .............................. 18
   2.2 Mean-Field Variational Inference ................ 20
      2.2.1 Mean-Field Variational Inference for Gaussian Mixture . . . 21
      2.2.2 Predictive Distribution ....................... 23
   2.3 Stochastic Variational Inference ................ 23
   2.4 Streaming Variational Bayes ................. 24
   2.5 Kalman Variational Bayes ...................... 26
      2.5.1 Internals of Kalman Filter ................. 26

3 Online Variational Inference Algorithms 29
   3.1 Removing dependence on $N$ ..................... 29
   3.2 Streaming Kalman Variational Bayes .......... 30
   3.3 Streaming Stochastic Variational Inference .... 31

4 Implementation Details 33
   4.1 Algorithm Initialization ....................... 33
      4.1.1 Weakly informative priors .................. 33
4.1.2 Components ........................................... 34
4.1.3 Mixing Coefficients ................................. 35
4.2 Hyperparameter Tuning ................................. 36

5 Evaluation .................................................. 37
  5.1 Stationary Synthetic Data ............................... 37
     5.1.1 Effect of Minibatch Approximation ............ 40
     5.1.2 Adding Noise ................................... 41
  5.2 Non-stationary Synthetic Data ......................... 43

6 Conclusion ................................................. 47
List of Figures

2-1  Graphical Model for Gaussian Mixture Model             21
2-2  Graphical Model for Kalman filter. The nodes continue both to to the
     left and to the right.                                     28
5-1  Heatmaps of example 3 Gaussian mixtures                  38
5-2  Integrated Squared Error between true distribution and predictive dis-
     tributions as more data is observed.                      39
5-3  Final Integrated Squared Error for 3 Gaussians dataset, averaged over
     10 trials.                                                 40
5-4  Effect of approximating dataset size to minibatch size      41
5-5  Integrated Squared Error for predictive distributions with 10/1 signal
     to noise ratio.                                          42
5-6  Converged clusters as ellipses on data with 10/1 signal to noise ratio. 43
5-7  Using different number of maximum mixtures for Kalman Variational
     Bayes when speed=1e-4                                    44
5-8  Log likelihood of last 10 minibatches averaged over 10 trials for different
     moving speeds.                                            45
## List of Algorithms

2.1 Coordinate Ascent Mean-Field Variational Inference .......................... 20
2.2 Stochastic Variational Inference .............................................. 25
2.3 Streaming Variational Bayes .................................................. 26
2.4 Kalman Variational Bayes ..................................................... 27
3.1 Streaming Kalman Variational Bayes ........................................ 31
3.2 Streaming Stochastic Variational Inference ................................. 32
4.1 Component Center Initialization using Repulsion ............................ 35
Chapter 1

Introduction

1.1 Motivation

With the increase of automation in our everyday life, there are more and more devices generating data. We can gain insights into systems by analysing the data they are generating. One of the problems that people encounter frequently when looking at such big datasets is understanding if the data that is being generated is similar to the old data or not. Weather there was a recent change in the system that made it behave differently.

Machine generated data is usually large and comes directly from the system that is generating it, i.e. there are no humans overseeing the data quality while it is being collected. The algorithm that is analysing the data is the first step in the analysis process, thus for this task we have to use unsupervised learning techniques.

Examples of problems of this type are estimating delays of airplanes in the airports based on historic data; understanding if any of a set of stores has anomalous electricity consumption when factoring its size, weather, opening hours, etc. and figuring out if anomalies could be explained by some of the properties of the data itself.

For most of this cases data is large, sometimes so large that it won’t fit in the memory of computers analysing it, so we have to rely on techniques that could work with streaming data.

In this thesis I focus on techniques for understanding underlying structure of the
datasets, and building distributions that describe the data. Specifically, I focus on the problem of detecting regions of unlikely points, or clusters of data that could have interesting explanations.

1.2 Defining the Problem

MacroBase\cite{1} is a framework that tries to address the need of people who are analysing machine generated data. One step of the process is to find outliers in the user-specified dataset. Work presented in this thesis was done to augment the MacroBase system with a density estimator that can work with variety of datasets, most of which are multimodal, so we need a very expressive model that can capture the underlying structure of many different datasets.

MacroBase is designed to be used by people with limited Computer Science knowledge, so it is paramount to have an algorithm that requires minimal human intervention to run on a new dataset. To achieve that we need an algorithm that is self tuning, and has few parameters.

Problem Statement

Our goal is to develop a model that will track the distribution of data as it is produced by some process, and will tag outliers, defined as points that are very unlikely to occur given the model.

We want the model to be easy and fast to train. It should also be possible to incrementally update it with new data and it should have the ability to track distributions even where there is some underlying drift in the distributions.

Potential Solutions

One way of finding outliers of changing distributions is to use time series analysis, but most of the algorithms that do this do not have very rich descriptive power and work with only certain types of distributions, most of which are unimodal.
Instead in this thesis I have adapted inference techniques for graphical models that were designed to work with stationary distributions, and made them work for changing distributions.

Graphical models have had recent success because mostly of their ability to provide very good unsupervised learning results for complex use cases, e.g. topic modeling [3]. Graphical models have very rich expressive power, and could model almost any kind of distribution, so they are well suited for working with complex datasets, and could capture huge number of use cases that MacroBase is trying to address.

Mixture models are a type of hierarchical graphical models, that model data as coming from a mixture of different distributions, which allows users to capture multimodal distributions. In this thesis I have picked a Gaussian Mixture Model to study inference techniques for mixture models. A Gaussian Mixture Model models the data as coming from $K$ different components, where each of the components is a Gaussian distribution with center $\mu_k$ and precision matrix $\Lambda_k$. Figure 2-1 shows the graphical model, and Section 2.2.1 explains the details of the Gaussian Mixture Model.

Most common techniques for graphical models are sampling MCMC techniques, which work for all types of graphical models but are usually slow and require starting from scratch when new data is introduced. Luckily there has been recent work in Variational Inference techniques that make the inference faster [7, 8], and streaming well [4].

1.3 Contributions

In this thesis I present Kalman Variational Bayes - a Variational Inference algorithm inspired by Stochastic Variational Inference, that uses Kalman Filters to smoothen the convergence of the algorithm. Devise a strategy for turning Kalman Variational Bayes and Stochastic Variational Inference into an online inference algorithm.

I have implemented Kalman Variational Bayes, Stochastic Variational Inference, and Streaming Variational Bayes for Gaussian Mixture Model I chose Gaussian Mixture Models because they can approximate many multimodal distributions, and they
show good experimental results for the datasets we have worked on. Since graphical models have many hyperparameters I have implemented Hyperband [9] to do automated hyperparameter tuning.

I evaluate the implemented inference algorithms on some real and synthetic data, to understand the feasibility of using them for non-stationary distributions. I conclude that both Stochastic Variational Inference, and Kalman Variational Bayes can be used for non-stationary data if the parameters are tuned correctly, but Kalman Variational Bayes is the best algorithm for automatically tuned hyperparameters because it performs the best on datasets with unpredictably changing distributions.

**Thesis Structure**

The rest of this thesis is organized in the following structure: Chapter 2 talks about inference techniques for Gaussian Mixture Models and gives extensive background on Variational Methods used in experiments. Chapter 3 talks about the modifications made to existing techniques in order to make them work in an online setting. Chapter 4 talks about implementation details, Chapter 5 goes through the evaluation of the introduced algorithms on different datasets, and Chapter 6 concludes this thesis.
Chapter 2

Inference Techniques for Graphical Models

Graphical models, such as Gaussian Mixture Model, provide a convenient and simple way of describing probability models. There are multiple algorithms for running inference on graphical models such as Expectation Maximization which works well for mixture models, sampling methods that work with variety of different models as well as multiple forms of approximate inference algorithms such as Variational Bayes and Expectation Maximization.

Bayesian inference algorithms have the advantage of providing not only estimates for the parameters in the graphical models but also giving distribution over those parameters. They have the downside of needing a prior distribution for all the parameters, but they tend to provide more robustness and better results once appropriate prior distributions are specified. For these reasons I have opted for Bayesian inference algorithms.

Most common Bayesian inference algorithms used in practice are sampling algorithms that do not require much prior setup from the user, but they have the downside of being slower compared to approximate algorithms, and a need to be restarted after new data has been observed. So for these reasons they do not suite our requirements.

Variational Inference on the other hand is designed to be fast, but has a drawback of introducing bias and sometimes is frowned upon because of this. But it perfectly
suits our needs, because there are already many Variational Inference algorithms that work with streaming data and usually they require only one pass over the dataset, which is advantageous if the entire dataset does not fit in the memory or if we add data over time. For this reason I focus on these algorithms in this thesis.

2.1 Variational Inference

Variational Inference transforms the inference problem into an optimization problem. Variational Inference restricts the probability distribution that we are trying to find to a family of distributions that have nice computational properties and tries to find analytical solution for the optimal distribution from that family. For complex graphical models this analytical solution is usually intractable to solve, but could be simplified to a point where usual optimization strategies such as gradient descent can yield a good solution.

We now discuss in more detail how Variational Inference works. Let $X$ be the set of all observed variables, and let $Z$ be the set of all latent variables, i.e., latent variables associated with each observed variables that are hidden in the model. Let $\lambda$ be the set of all not hidden model variables, e.g. cluster center locations, cluster proportions, etc. And let $\alpha_0$ be the set of fixed model parameters, for example, the number of mixtures.

Because each latent variable is associated with only a single data point, and all model variables govern the overall model behavior, sometimes I will refer to model variables as global variables and to latent variables as local variables.

Graphical models specify a joint distribution $p(X, Z, \lambda|\alpha_0)$. And the goal of the inference is to maximize the likelihood of the data given the model, which is given by the following formula

$$p(X|\alpha_0) = \sum_{Z, \lambda} p(X, Z, \lambda|\alpha_0)$$  \hspace{1cm} (2.1)

Where the sum is replaced with an integral for continuous variables.

Variational Inference assumes that the likelihood function is too hard to optimize
directly and that optimizing the complete-data likelihood function \( p(X, Z, \lambda | \alpha_0) \) is much easier. For the rest of this thesis, in order to simplify the notation, I will omit conditioning distributions on model parameters \( \alpha_0 \), and the conditioning will be implied for the rest of distributions where appropriate.

Let \( q(Z, \lambda) \) be some distribution defined over all model variables (latent and global). Notice that for any choice of \( q(Z, \lambda) \) the following decomposition holds. (See Bishop Chapter 10.1[2] for derivation).

\[
\ln p(X) = \int_{Z, \lambda} q(Z, \lambda) \ln \left( \frac{p(X, Z, \lambda)}{q(Z, \lambda)} \right) dZd\lambda - \int_{Z, \lambda} q(Z, \lambda) \ln \left( \frac{p(X | Z, \lambda)}{q(Z, \lambda)} \right) dZd\lambda
\]

The second term (with the minus sign) is Kullback-Leibler divergence of \( q \) and \( p \). And since Kullback-Leibler divergence is always non-negative it means that the first term is a lower bound on the log likelihood function \( \ln p(X) \), so it is referred to as the Evidence Lower Bound, which we denote \( \mathcal{L}(q) \).

Instead of maximizing the log likelihood function \( \ln p(X) \) directly, Variational Inference tries to maximize the lower bound, i.e.

\[
\mathcal{L}(q) = \int_{Z, \lambda} q(Z, \lambda) \ln \left( \frac{p(X, Z, \lambda)}{q(Z, \lambda)} \right) dZd\lambda
\]  

(2.3)

Since the Evidence Lower Bound has an unknown distribution \( q(Z, \lambda) \) in it, Variational Inference restricts \( q \) to some family of distributions and tries to find the optimal distribution from that family that maximizes \( \mathcal{L}(q) \).

The disadvantage of Variational Inference is that for different Graphical models the family of distributions \( q \) is usually different and one has to go through many derivations to find the family that yields a nice optimization strategy. Since I am aiming to find a model that will work adequately for a wide range of applications, this derivations are justified and give desired level of speed improvements.
Algorithm 2.1 Coordinate Ascent Mean-Field Variational Inference.

**input** data $X = \{x_i\}$ of size $N$; desired number of clusters $K$

Initialize $\lambda^0$

**repeat**

for $n = 1$ to $N$ do

update local variables

$$\phi_{nj}^{(t)} = \mathbb{E}_{\lambda^{(t-1)}}[\ln p(x_n, z_{n,-j}, \lambda^{(t-1)})]$$

end for

update global variables

$$\lambda^{(t)} = \mathbb{E}_{\phi^{(t)}}[\ln p(X, Z)]$$

**until** Evidence Lower Bound converges

2.2 Mean-Field Variational Inference

Mean-Field Variational Inference restricts the family of distributions $q(Z, \lambda)$ to distributions that factor into distributions over local and global parameters, i.e.

$$q(Z, \lambda) = q(Z)q(\lambda) \quad (2.4)$$

This factorization allows calculating tractable solutions for each of the factors of the optimal distribution $q^*$ based on the other factor, i.e.

$$\ln q^*(Z) = \mathbb{E}_\lambda[\ln p(X, Z, \lambda)] + \text{const}$$

$$\ln q^*(\lambda) = \mathbb{E}_Z[\ln p(X, Z, \lambda)] + \text{const} \quad (2.5)$$

After this we can perform coordinate ascent on the space of variables. The most straightforward way of doing coordinate ascent is to initialize global variables, and then iteratively update local and global variables until we are satisfied with the convergence. Algorithm 2.1 summarizes this approach. I’ll talk more about different ways of initializing global variables in Section 4.1.

Factorizing distributions introduces bias into the inference because we are only looking at the subset of possible distributions, but as we will see in the experiments section, for the applications that we are interested, this bias is manageable.
Mean-Field Variational Inference has a major drawback. It requires iterating over all local variables in order to make a single update to global variables. When the number of local variables is huge this may take a long time, especially because each iteration of coordinate ascent can be slow. I’ll discuss some solutions in Sections 2.3, 2.4, and 2.5, and I’ll propose ways of converting these algorithms into online algorithms in Chapter 3.

2.2.1 Mean-Field Variational Inference for Gaussian Mixture

Figure 2-1: Graphical Model for Gaussian Mixture Model

Figure 2-1 shows the graphical model for a Gaussian Mixture Model which we will use the most in this thesis. Graphical models show dependencies between variables, but not specify the distributions, so just specifying the graphical model is not enough and one needs to provide the distributions separately.

In Figure 2-1 circles represent variables, e.g. $\mu$, $\Lambda$, filled circles represent observables, i.e., data that we are using. Arrows show dependencies, i.e., $x_n$ is parametrised by $z_n$, $\mu$ and $\Lambda$. Plates show replication, i.e., there are $N$ different observables $x_n$ and that each of them has a variable $z_n$ associated with it, but all of them are dependent on global variables $\mu$ and $\Lambda$. Also all local variables $z_n$ are parametrised by global mixing coefficients $\pi$.

For Gaussian Mixture Model, each of $z_n$’s has a Multinomial distribution with parameter $\pi$, And $x_n$ are distributed by a mixture of $K$ Gaussian distributions (we
call this clusters) each with center $\mu_k$ and a precision $\Lambda_k$ and normalizing coefficient $\pi_k$.

For Gaussian Mixture Model we try to find a distribution $q^*$ that has two factors, first factor is for variables governing the Gaussian components, which have a Gaussian-Wishart distribution for the optimal solution, given by the following probability function:

$$q^*(\mu_k, \Lambda_k|\mu_0, \beta_k, W_k, \nu_k) = \mathcal{N}(\mu_k|\mu_0, (\beta_k\Lambda_k)^{-1}) \mathcal{W}(\Lambda_k|W_k, \nu_k) \quad (2.6)$$

Where $\mathcal{N}$ is a multivariate Gaussian distribution, and $\mathcal{W}$ is a Wishart distribution.

And the second factor is for mixing variables $\pi$, which have Dirichlet distribution as the optimal solution, given by the distribution

$$q^*(\pi) = Dir(\pi|\alpha) \quad (2.7)$$

So we have the following set of global variables that we need to optimize for:

$$\lambda = \{\alpha, \mu_0, \beta, W, \nu\} \quad (2.8)$$

For each of the data points $x_n$ we have a corresponding latent variable $z$ that is a binary $K$ (number of clusters) vector, that indicates which cluster the data point belongs to.

$$p(z|\alpha) = \prod_{k=1}^{K} \pi_k^{z_{nk}} \quad (2.9)$$

Each $z_n$ is parametrized by a Multinomial distribution with parameters $\phi_{nk}$, such that

$$q^*(z) = \prod_{n=1}^{N} \prod_{k=1}^{K} \phi_{nk}^{z_{nk}} \quad (2.10)$$

So the set of all local parameters is $Z = \{\phi_{nk}\}$
2.2.2 Predictive Distribution

After we have an estimates for global variables, in order to calculate the predictive distribution for a new sample \( \hat{x} \) with latent variables \( \hat{z} \), we have to marginalize the overall probability function, i.e.

\[
p(\hat{x}|X) = \sum_{\hat{z}} \int p(\hat{x}|\hat{z}, \lambda)p(\lambda|X)d\lambda
\]  

(2.11)

Predictive Distribution for Gaussian Mixture Model

For a Gaussian Mixture Model when we carry out the integrations in Equation 2.11 we are left with the following predictive distribution [2]:

\[
p(\hat{x}|X) = \frac{1}{c} \sum_{k=1}^{K} c_k St(\hat{x}|\mu_0, L_k, \nu_k + 1 - D)
\]  

(2.12)

Which is a Multivariate Student T distribution with sample centers as Gaussian Mixture components, but with a slightly modified precision matrix, which is given by equation 2.13 (\( D \) is the dimensionality of data points).

\[
L_k = \frac{(\nu_k + 1 - D)\beta_k}{1 + \beta_k} W_k
\]  

(2.13)

When talking about predictive log likelihood of data in the future, I will be referring to evaluating newly seen samples using equation 2.12.

2.3 Stochastic Variational Inference

Stochastic Variational Inference(SVI) [7] makes two more approximations to make Mean-Field Variational Inference run faster. Instead of going through the entire dataset to get the next estimate of global parameters, which could be very time consuming and lead to very insignificant improvements, SVI calculates a stochastic natural gradient treating the data as consisting of copies of a randomly chosen datapoint. So we calculate local variables only for one point.
The stochastic gradient calculation leads to an estimate for the global variables \( \hat{\lambda} \), which will be very noisy because the points will be coming from different clusters. In order to smooth this noise, SVI employs the Robbins-Monro algorithm (equation 2.14) which has theoretical convergence guarantees if the step sizes are chosen according to conditions in equation 2.15 [7].

\[
\lambda^{(t)} = (1 - \rho_t)\lambda^{(t-1)} + \rho_t \hat{\lambda}
\]

\[
\sum_{t=0}^{\infty} \rho_t = \infty \text{ and } \sum_{t=0}^{\infty} \rho_t^2 < \infty
\]

SVI chooses step-size according to equation 2.16, where \( \kappa \in (0.5, 1) \) is called the forgetting rate of Stochastic Variational Inference, and \( \tau \) is set to be 1.

\[
\rho_t = (t + \tau)^{-\kappa}
\]

SVI speeds up the convergence speed, but because the step-size equation 2.16 has two new parameters \( \tau \) and \( \kappa \), it leads to more complexity of overall model, because the user now needs to specify more parameters. I will talk more about figuring out right values for these parameters in Section 4.2. Algorithm 2.2 summarises Stochastic Variational Inference.

### 2.4 Streaming Variational Bayes

Instead of making step-size decay over time like SVI, Streaming Variational Bayes (SVB) [4] builds up the model gradually by calculating an estimate for global variables after evaluating local variables for each new datapoint.

SVB does not need to use a smoother, because in the begining of the algorithm only a few points are used to calculate global variables. Since all the points have equal weight, once more points are added when the estimate of global variable is good, it will improve newer estimates, because the old points with bad contributions
Algorithm 2.2 Stochastic Variational Inference

**input** data $X = \{x_i\}$ of size $N$; desired number of clusters $K$; Robbins-Monro parameters $\kappa$ and $\tau$.
Initialize $\lambda^0$
Set up step-size schedule for $\rho_t$

repeat
   for $n$ randomly in $\{1, \ldots, N\}$ do
      update local variables
      $$\phi_{nj}^{(t)} = \mathbb{E}_{\lambda^{(t-1)}}[\ln p(x_n, z_{n,-j}, \lambda^{(t-1)})]$$
      calculate global variables estimate, treating as if $x_n$ is replicated $N$ times
      $$\hat{\lambda} = \mathbb{E}_{\phi^{(t)}}[\ln p(X_n^{(N)}, Z_n^{(N)})]$$
      update global variables, using the estimated value
      $$\lambda^{(t)} = (1 - \rho_t)\lambda^{(t-1)} + \rho_t \hat{\lambda}$$
   end for
until Evidence Lower Bound converges

will have smaller and smaller weights. See Algorithm 2.3 for an overview of Streaming Variational Bayes.

The main advantage of Streaming Variational Bayes over Stochastic Variational Inference is that SVB can update local variables in parallel using the best available estimate of global variables at the time. And because of the update equation of global variable involves summation over local variables, it is easy to combine results from different processes running in parallel because local variables do not depend on each other when conditioned on global variables.

I have not tried the parallel feature of SVB, but since it is a streaming model by nature, it provides us with a nice performance benchmark that we will use in the Chapter 5 when evaluating different algorithms.
Algorithm 2.3 Streaming Variational Bayes

input data $X = \{x_i\}$ of size $N$; desired number of clusters $K$
Initialize $\lambda^0$
Initialize $Z$ to zero
repeat
    for $n = 1$ to $N$ do
        update local variables, potentially asynchronously
        $$\phi_{nj}^{(t)} = \mathbb{E}_{\lambda(\cdot-1)}[\ln p(x_n, z_{n,-j}, \lambda^{(\cdot-1)})]$$
        update global variables
        $$\lambda^{(t)} = \mathbb{E}_{\phi^{(t)}}[\ln p(X_{0:n}, Z_{0:n})]$$
    end for
until Evidence Lower Bound converges

2.5 Kalman Variational Bayes

Kalman Variational Bayes is very similar to Stochastic Variational Inference, but instead of a Robbins-Monro smoother it uses a Kalman filter\[8\] with moving speed fixed at 0. Kalman filter is a more expressive smoother than Robbins-Monro, and Kalman filters can be reduced to Robbins-Monro with $\kappa = 1$ if process noise is set to zero, (more on this in Subsection 2.5.1). Algorithm 2.4 present an overview of Kalman Variational Bayes.

The motivation for using a Kalman filter is that in practice it is used to track moving objects, so we hope hope that it will be effective in in tracking moving non-stationary distributions, which is the main class of distributions for which we would like to have an inference algorithm.

2.5.1 Internals of Kalman Filter

The Kalman filter is an algorithm that takes a series of measurement over time containing noise, and produces estimates that are more accurate than the observed data. The Kalman filters are themselves graphical models, and have the same structure as the Hidden Markov Models, but deals with continuous variables instead of discrete ones. Figure 2-2 shows the graphical model.
Algorithm 2.4 Kalman Variational Bayes

**input** data $X = \{x_i\}$ of size $N$; desired number of clusters $K$; Kalman filter parameters $\sigma_a, \sigma_z$  
Initialize $\lambda^0$  
Initialize a Kalman Filter $f(\sigma_a, \sigma_z)$  
repeat  
  for $n$ randomly in $\{1, \ldots, N\}$ do  
    update local variables for clusters $j = 1$ to $K$  
    \[ \phi_{nj}^{(t)} = \mathbb{E}_{\lambda^{(t-1)}}[\ln p(x_n, z_{n-j}, \lambda^{(t-1)})] \]  
    calculate global variables estimate, treating as if $x_n$ is replicated $N$ times  
    \[ \hat{\lambda} = \mathbb{E}_{\phi^{(t)}}[\ln p(X_{n}^{(N)}, Z_{n}^{(N)})] \]  
    update global variables, using the estimated value  
    \[ f.\text{update}(\hat{\lambda}) \]  
    \[ \lambda^{(t)} = f.\text{estimate()} \]  
  end for  
until Evidence Lower Bound converges

The Kalman filter can be thought as a particle moving in space that has Gaussian distributed acceleration $\mathcal{N}(0, \sigma_a^2)$ which we call process noise, and the measurement of the position is also noisy with noise distributed as a Gaussian $\mathcal{N}(0, \sigma_z^2)$, this noise is called measurement noise.

This two parameters govern the behavior of the Kalman filter together with the initial conditions of the filter, that I will talk about in Chapter 4. I will focus my attention to the Kalman filter with constant process and measurement noise variances to make the filter as simple as possible and keep the ability of global parameters to gain and maintain velocity during the inference process.

In this thesis I will use a fixed interval Kalman filter which exposes 2 methods to the user - `update(observation)` and `estimate()`. The `update` method takes a new measurement as an argument and updates the internal state based on the observation. The `estimate` method returns the current estimate of the system.
Extensions of Kalman filter

If allowed to make multiple passes over the data, Kalman filter could also infer the noise parameters it needs to use to optimize the filter, so for batch algorithms this is a possible direction to investigate further which is beyond the scope of this thesis.

As I said previously if we fix process and measurement noise variances, the Kalman filter becomes a filter that is very much like Robbins-Monro but with slightly different function for a step-size over epochs $t$. If we set $\sigma_z$ to 0 it reduces to a Robbins-Monro filter with $\kappa = 1$. 

Figure 2-2: Graphical Model for Kalman filter. The nodes continue both to the left and to the right.
Chapter 3

Online Variational Inference Algorithms

3.1 Removing dependence on $N$

So far, when considering Stochastic Variational Inference and Streaming Variational Bayes algorithms refine the global parameter estimates using a single point. Similar to many stochastic optimization algorithms [7], [4], we use several examples at a time to refine this estimates. We call a set of points used to make a single update to global variables a minibatch.

For Stochastic Variational Inference and Kalman Variational Bayes from previous chapter, if we use a minibatch of size $M$ with indices $\{i_1, \ldots, i_M\}$ the update equation for global parameter estimate $\hat{\lambda}$ becomes

$$
\hat{\lambda} = \frac{N}{M} \mathbb{E}_{q(\cdot)}[\ln p(X_{i_1:i_M}, Z_{i_1:i_M})]
$$

(3.1)

Equation 3.1 has an explicit dependency on $N$ (number of points in the dataset).

In order to remove the dependency on $N$ from equation 3.1, I make an assumption that there are total $M$ points in the dataset, instead of $N$. This assumption transfer the equation 3.1 into
\[ \hat{\lambda} = \mathbb{E}_{\mathcal{g}(\xi)}[\ln p(X_{i_1:i_M}, Z_{i_1:i_M})] \]  

(3.2)

When we choose \( M \gg \beta_B \) and \( M \gg \nu_B \), the effect on the predictive distribution for Gaussian Mixture Model (see equation 2.12) is that \( \nu_k, \beta_k \) become \( N/M \) times smaller and \( W_k \) become \( N/M \) times larger and \( \mu_{0k} \) don’t change.

Looking at equation 2.13, we see that \( L_k \) will stay the same as long as \( M \gg \beta_B \) and \( M \gg \nu_B \). And the only major change in the predictive distribution is that concentration parameter of Student T distribution becomes \( N/M \) times smaller, so the belief is no longer as strong as it was before. We will see in Section 5.1.1 that this has very small effect on the performance, as long as we pick minibatch size to be large.

### 3.2 Streaming Kalman Variational Bayes

First step of making Kalman Variational Bayes into a streaming algorithm is to adapt the minibatching approach from Section 3.1 and calculate \( \hat{\lambda} \) for every minibatch and feed it into the Kalman filter.

Even though we are trying to track changing distributions, I fix \( \sigma_a \) and \( \sigma_z \) for the entire duration of the algorithm, and estimate those values either by using grid search or Hyperband, see Section 4.2 for more details on estimating hyperparameters.

We will use Streaming Kalman Variational Bayes for evaluations section, here is an overview of the algorithm. Assuming we have values for hyperparameters \( K, M, \sigma_a, \sigma_z \) we first initialize the prior Normal-Wishart distribution (Section 4.1), and our initial guess for \( \lambda \). Then for every minibatch we calculate new estimate \( \hat{\lambda} \) of global variables based on previous value \( \lambda^{(t-1)} \) and points in the \( t \)-th minibatch. Then we feed \( \hat{\lambda} \) into the Kalman filter and set \( \lambda^{(t)} \) to be the estimate of the new state from Kalman filter. (For summary see Algorithm 3.1)
Algorithm 3.1 Streaming Kalman Variational Bayes

**input** data $x_i$; desired clusters $K$; Kalman filter parameters $\sigma_a, \sigma_z$; desired minibatch size $M$;
Initialize $\lambda^0$
Initialize a Kalman Filter $f(\sigma_a, \sigma_z)$

for $\{i_1, \ldots, i_M\}$ sliding over $\{1, \ldots, N\}$ do
update local variables for points in the minibatch
for $n = i_1$ to $i_M$ do
update local variables for clusters $j = 1$ to $K$

$$\phi_{nj}^{(t)} = \mathbb{E}_{\lambda^{(t-1)}}[\ln p(x_n, z_{n,j}, \lambda^{(t-1)})]$$
end for

calculate global variables estimate, treating that there are only $M$ points in the dataset

$$\hat{\lambda} = \mathbb{E}_{\phi^{(t)}}[\ln p(X_{i_1:i_M}, Z_{i_1:i_M})]$$

update global variables, using the estimated value

$$f.\text{update}(\hat{\lambda})$$

$$\lambda^{(t)} = f.\text{estimate}()$$
end for

### 3.3 Streaming Stochastic Variational Inference

The first step of converting Stochastic Variational Inference into a streaming algorithm, is to use the minibatch approach from Section 3.1 similar to Streaming Kalman Variational Bayes.

The second step is to remove dependence of $\rho$ on time $t$. Remember that $\rho_t = (t + \tau)^{-\alpha}$ for SVI. If we want our algorithm to be streaming, and have the ability to track changes in the distribution we can’t allow $\rho$ to go to zero at any point of time. And explicit dependence of $\rho$ on $t$, will make stepsize $\to 0$ for large values of $t$. To address this issue, we fix $\rho$ to be a constant and be a user-specified parameter of the algorithm. Now the update equation for the global variables becomes:

$$\lambda^{(t)} = (1 - \rho)\lambda^{(t-1)} + \rho \hat{\lambda} \quad (3.3)$$

We will use Streaming Stochastic Variational Inference for evaluations section,


**Algorithm 3.2** Streaming Stochastic Variational Inference

**Input** data $x_i$; desired clusters $K$; desired minibatch size $M$; Learning rate $\rho$

Initialize $\lambda^0$

for $\{i_1, \ldots, i_M\}$ sliding over $\{1, \ldots, N\}$ do

update local variables for points in the minibatch

for $n = i_1$ to $i_M$ do

update local variables for clusters $j = 1$ to $K$

$$\phi_{n}^{(t)} = \mathbb{E}_{\lambda^{(t-1)}}[\ln p(x_n, z_{n}, -j, \lambda^{(t-1)})]$$

end for

calculate global variables estimate, treating that there are only $M$ points in the dataset

$$\hat{\lambda} = \mathbb{E}_{\phi^{(t)}}[\ln p(X_{i_1:i_M}, Z_{i_1:i_M})]$$

update global variables, using the estimated value

$$\lambda^{(t)} = (1 - \rho)\lambda^{(t-1)} + \rho \hat{\lambda}$$

end for

Here is an overview of the algorithm. Assuming we have values for hyperparameters $K$, $M$, and $\rho$ we first initialize the prior Normal-Wishart distribution (Section 4.1), and our initial guess for $\lambda$. Then for every minibatch we calculate new estimate $\hat{\lambda}$ of global variables based on previous value $\lambda^{(t-1)}$ and points in the $t$-th minibatch. Then we use equation 3.3 to calculate $\lambda^{(t)}$. (For summary see Algorithm 3.2)
Chapter 4

Implementation Details

4.1 Algorithm Initialization

In order to start all the algorithms described in previous chapters, we have to initialize $\lambda^{(0)}$, which for Gaussian Mixture Model consists of mixing coefficients and Gaussian-Wishart distribution parameters.

During the Bayes update step when calculating new estimates for global variables $\lambda$, we also need to maintain the prior state of the system before the algorithm started. To be able to converge to many possible distributions we should start with as weak of a believe as possible.

4.1.1 Weakly informative priors

Some algorithms start with a probability distribution that is fully non-informative, e.g. fitting a single Gaussian [2], to be able to capture all possible states of the system. For mixture models having a fully non-informative prior and obtaining proper posterior predictive distributions is impossible because there is always a possibility that there are no points observed from on of the components[5]. Instead I follow the approach from [10] and pick weakly informative priors, using some statistics available during the initialization of the inference algorithm.

Variational Inference requires us to pick a conjugate prior, i.e. a prior distribution
that has the same functional form as the distributions we are trying to infer, so I have to initialize a Gaussian-Wishart distribution. I will use a subscript $B$ (Base) to indicate parameters for the prior distribution. I initialize them using Equations 4.1-4.4.

\[
\begin{align*}
\mu_0B &= \text{mean} \quad (4.1) \\
\beta_B &= R^{-2} \quad (4.2) \\
W_B &= I_d \quad (4.3) \\
\nu_B &= \text{dimension} \quad (4.4)
\end{align*}
\]

Here $R$ is the largest spread of the first minibatch points. $\mu_0B$ is set to the mean of the first minibatch points, using minibatch width instead of the entire data width because the aim is to come up with an algorithm that is truly online.

### 4.1.2 Components

Initializing locations of mixing components uniform randomly from data points leads to sub-optimal conversion of components. This can result in, for example, local maxima for distribution functions, e.g. merging of multiple components into a single one, if components start very close to each other. To avoid this type of behavior I initialize location of clusters using a slightly modified technique from [6]. The main idea is to pick points that are far from each other, in order to cover the majority of the space.

First I pick a point at random from the points provided. Then I pick every following point by taking a sum of inverse Mahalanobis-like distance squared, and take the point that has the lowest sum. I define Mahalanobis-likes distances between points $x$ and $y$ as

\[
d(x, y) = [(x - y)^T S^{-1}(x - y)]^{1/2} \quad (4.5)
\]
Algorithm 4.1 Component Center Initialization using Repulsion

**input** data $x_i$; desired clusters $K$

$c_0 := \text{random}(\{x_i\})$

**for** $k = 1$ to $K-1$ **do**

$c_k = \arg\min_x \sum_i^k \frac{1}{(x-c_i)^T S^{-1}(x-c_i)}$

**end for**

**return** $\{c_0, \ldots, c_K\}$

Where $S$ is a diagonal matrix with diagonal elements equal to the spread of data in that coordinate.

The intuition behind this approach is that if each point was repelling other points, I take the points that minimize the overall energy (after the first one was picked at random). The use of $S$ instead of the covariance matrix of all the points saves time by reducing amount of computation, at the same time providing with a good estimate of variance in each of the coordinates. Algorithm 4.1 given an overview of cluster location initialization algorithm used.

After cluster means are initialized, the rest of parameters of the starting distribution are chosen to be equal to those of the prior.

$$\beta_k = 1 \quad (4.6)$$

$$\nu_k = \nu_B = \text{dimension} \quad (4.7)$$

$$W_k = W_B = I_d \quad (4.8)$$

### 4.1.3 Mixing Coefficients

Finite Gaussian Mixture Models use a Multinomial distribution for mixing coefficients. I initialize all weights of the multinomial distribution to be equal to $1/K$.

For infinite Gaussian Mixture Models, which use a Dirichlet Process, I have to initialize a prior distribution concentration parameter, which I have chosen to be 0.3 from experimental results. This number is chosen empirically, and might differ from experiment to experiment, but in our experiments, its effects did not seem particularly
significant so I won’t go into the details of setting it.

For infinite Gaussian Mixture Model we also have to choose a truncation parameter, to bound the amount of computation. Because performance a comparison of different methods is beyond the scope of this thesis, I have chosen the truncation parameter to be parameter to be 20, well above potential number of clusters (see Chapter 5 for datasets).

4.2 Hyperparameter Tuning

For batch inference algorithms, it is typical to run a grid search with a range of different hyperparameters in order to find the optimal set of hyperparameters for the given dataset. This strategy does not work for streaming inference, especially with non-stationary distributions, because the set of optimal hyperparameters changes over time.

To address the problem of finding hyperparameters in a streaming setting we use Hyperband[9]. Hyperband starts multiple instances of the algorithm in parallel, all with different sets of configuration hyperparameters. After each minibatch Hyperband drops the instances that do not perform well. If we start with \(2^H\) instances of a model with different sets of hyperparameters each, in only \(H\) steps (which we take to be either a minibatch, or a small multiple of minibatches) we are left with only one instance, that can be used to calculate predictions and evaluate against other methods.

I also use Hyperband to start multiple instances of the inference algorithm with different random seeds. This leads to more computation in the first couple of minibatches, but in the long run makes the instance chosen by Hyperband converge faster, so we see meaningful results after only couple of minibatches.
Chapter 5

Evaluation

In this chapter we compare different Streaming Variational Inference algorithms presented in previous chapters under different conditions to demonstrate that even though Variational Inference assumes that the distributions are stationary, the algorithms can do quite well for non-stationary distributions. Section 5.1 explores how well algorithms perform when data has a stationary distribution, with and without noise. Section 5.2 explores how well the different algorithms can track moving distributions at different speeds.

5.1 Stationary Synthetic Data

To compare different algorithm performance for stationary data, we generate points from the following mixture of 3 Gaussians

\[
\begin{align*}
0.5\mathcal{N}\left(\begin{bmatrix} 1.5 \\ 2 \end{bmatrix}, \begin{bmatrix} 0.5 & 0.4 \\ 0.4 & 0.5 \end{bmatrix}\right) + 0.3\mathcal{N}\left(\begin{bmatrix} 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 0.3 & 0 \\ 0 & 0.6 \end{bmatrix}\right) + 0.2\mathcal{N}\left(\begin{bmatrix} 4.5 \\ 1 \end{bmatrix}, \begin{bmatrix} 0.9 & 0.2 \\ 0.2 & 0.3 \end{bmatrix}\right)
\end{align*}
\]

(5.1)

Figure 5-1a shows a heatmap of the data. This dataset provides with a good basic test case, because it has overlapping distributions with different weights and each of the distributions has different shape of a covariance matrix.
Stationary Comparison Experiment

For the first experiment I compare 4 streaming algorithms - Kalman Variational Bayes, Streaming Variational Bayes, Stochastic Variational Inference, and SVI with learning rate fixed at 1, which I call forgetful because it does not use any kind of filtering on the global variable estimates, i.e. has the maximum possible learning rate throughout the inference algorithm. Therefore, I expect forgetful to perform well with non-stationary data because it quickly adapts to changing distributions.

We train all models using the same configuration settings, e.g., a random seed to initialize clusters, and evaluate goodness of fit using Integrated Squared Error\(^1\) between predicted distributions of different models and the true distribution from equation 5.1.

In this experiment we fix minibatch size to 1000, and choose the optimal learning rate of SVI and \(\sigma_z\) and \(\sigma_a\) of Kalman Variational Bayes using grid search.

As you can see from Figure 5-2a SVB is very slow to converge, and even after 200,000 points it is still not as close to the true distribution as other algorithms. This

\(^1\)Integrated Squared Error is the integral of the difference squared of two functions, in this case distribution functions.
is as expected, because the strength of SVB is the fact that it can run in parallel effectively dividing the convergence time. Thus SVB is ill suited for non-stationary distributions because it is very slow to converge.

Figure 5-2a shows that all forgetful, Stochastic Variational Inference and Kalman Variational Bayes produce predictive distributions that are very close to true distributions. Forgetful is also very noisy, i.e. it oscillates a lot, as is expected because it does not do any kind of smoothing of global variables.

Figure 5-2: Integrated Squared Error between true distribution and predictive distributions as more data is observed.

Figure 5-3 shows the spread of final Integrated Squared Error for different algorithms taken over multiple trials. It confirms our previous observation that Kalman Variational Bayes performs as well as Stochastic Variational Inference for stationary data, but it has the problem that for some trials it converges to something that is very different. I use Hyperband to address some of this inconsistency. I initialize parallel instances that use different random seeds for cluster initialization and drop the instances that perform poorly over time. This issue becomes larger when we add noise to the system, so we will revisit investigation when noisy data is added in Section 5.1.2.
5.1.1 Effect of Minibatch Approximation

In the next experiment we look at how performance of Stochastic Variational Inference is affected when we approximate the size of the data to be equal to that of a minibatch, i.e. we measure the difference in performance of SVI in batch and streaming modes. We have chosen SVI, because it is slightly simpler than Kalman Variational Bayes and has been studied more extensively.

Figure 5-4a shows the Integrated Squared Error of batch SVI ($N = 200,000$) and streaming SVI ($N = 1,000$). You can see that for a fixed learning rate $\rho = 0.15$ (the same learning rate as the one found in previous experiment), online SVI converges even faster than the batch SVI.
Both batch and online SVI converge to the same value of Integrated Squared Error. Figure 5-4b shows Maximum a Posteriori (MAP) estimate of covariances and locations of clusters from both online and batch SVIs, together with true values. We see that even if MAP estimates are a little different from the true values, MAP estimates from both versions overlap, so we are not losing noticeable accuracy because of the approximation from Section 3.1.

Figure 5-4: Effect of approximating dataset size to minibatch size

5.1.2 Adding Noise

In this experiment we add noise to the 3 Gaussians from equation 5.1. To simulate anomaly detection situation, I add Uniform noise $\mathcal{U}([-2,9] \times [-3,5])$. Figure 5-1b shows an example of a noisy dataset that has 10/1 signal to noise ratio.

Using the same hyperparameters that we had found for noiseless data, we run the Stationary Comparison Experiment again using Hyperband with multiple initialization random seeds. We also skip running SVB, because it is very slow to converge, and is not very well suited for non-stationary distributions.

We instantiate all algorithms with 16 seeds, and after training on every minibatch starting from 2nd we eliminate half of the candidates that have the worst predictive log likelihood on the following minibatch, until we are left with only one candidate. We use predictive log likelihood to ensure that algorithms don’t use knowledge of the
Figure 5-5: Integrated Squared Error for predictive distributions with 10/1 signal to noise ratio.

Figure 5-5a shows how Integrated Squared Error changes over number of points processed. Note that now ISE values are higher than in noiseless case, but the behavior of different algorithms is almost the same, and they have closer results. Forgetful is performing as well as SVI and Kalman Variational Bayes, but is still more noisy even when converged.

Figure 5-5b shows the final convergence ISE from multiple trials. Notice that Kalman Variational Bayes now performs slightly better than SVI, but within the margin of error, but still both of SVI and Kalman Variational Bayes still have runs when they don’t converge to the correct distribution.

Figure 5-6 shows what converged distributions look like for a typical trial (Figure 5-6a), and for a trial that leads to abnormally high ISE (Figure 5-6b).

We can see that the problem is that Kalman Variational Bayes sometimes merges clusters. To make it easier to split the clusters again, we can use a Dirichlet Prior Gaussian Mixture Model, or simply specify higher number of clusters for finite Gaussian Mixture Model. Figure 5-1a shows final convergence on an example dataset using Dirichlet Prior Gaussian Mixture Model. You can see that there are more than 3 clusters with non-zero weights, but the resulting predictive distribution is almost
identical to the true distribution.

(a) Typical example of converged clusters  
(b) Bad convergence probably because bad initialization of the algorithms

Figure 5-6: Converged clusters as ellipses on data with 10/1 signal to noise ratio.

5.2 Non-stationary Synthetic Data

In this section we try to assess how well SVI and Kalman Variational Bayes can track moving distributions. For a moving distribution we pick the following mixture of 2 Gaussians whose centers move with constant velocities.

\[
.5\mathcal{N} \left( \begin{bmatrix} 0 \\ 0 \end{bmatrix} + v_1 t, \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix} \right) + .5\mathcal{N} \left( \begin{bmatrix} 1 \\ 1 \end{bmatrix} + v_1 t, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right)
\] (5.2)

Here \( v_1 \) and \( v_2 \) are constant for the duration of each trial, but change from trial to trial. We fix the magnitude of \( v_1 \) and \( v_2 \) to value \( s \), and vary the directions uniformly at random.

We tune Kalman Variational Bayes and SVI on stationary data (e.g. use the hyperparameters from previous section) and run it on non-stationary data; we see that Kalman Variational Bayes is actually performing quite well still.

Infinite Mixture

Before going through results for different speeds, we want to eliminate the performance anomalies that we saw in previous section (see Figure 5-6b).
That type of behavior happens even more when clusters are moving, when clusters get close together Kalman Variational Bayes sometimes merges the clusters, and it’s hard to split them once they are further apart, so instead of looking at a mixture of 2 Gaussians, we look at Dirichlet Prior Gaussian Mixture Model (see Section 4.1.3 for more details). When we use different truncating parameters we get slightly different results, sometimes clusters merge or split, but overall we can see from Figure 5-7 that in the end even if we have potentially more clusters we converge to the same solution.

In this section we use predictive log likelihood of the next minibatch as a comparison metric instead of ISE, because ISE is not very well defined for moving data.

Figure 5-7 shows the predictive log likelihood of the next minibatch for one trial that runs Kalman Variational Bayes with different number of possible Gaussians. This is a very typical performance that we see, and whenever we choose number of possible Gaussians to be \( \geq 10 \) we obtain good performance and all the true clusters
are present in the final estimate. Therefore, we pick $K = 10$ for finite Gaussian Mixture Model.

Moving distributions at different speeds

![Figure 5-8: Log likelihood of last 10 minibatches averaged over 10 trials for different moving speeds.](image)

To test the ability of SVI and Kalman Variational Bayes to track different distributions, we run multiple trials on the dataset from equation 5.2 with different speeds.

Figure 5-8 shows the average log likelihood of last 10 minibatches (before models were trained on those minibatches) for different algorithms run on datasets with clusters moving at different speeds. Each point on the figure is averaged over 10 different trials with the same speed.

The cyan line in Figure 5-8 represents mean log likelihood of minibatches using the mean of true distributions for points in the minibatch being evaluated. So we would not expect to be able to beat that value, and if an algorithm is staying close to the cyan line it means it is successfully tracking the distribution.

Looking at Figure 5-8 we see that Kalman Variational Bayes tracks distributions even better than forgetful, which is the highest paced SVI algorithm, because it has higher predictive log likelihood for higher speeds. All algorithms show very similar
results for lower speeds, so our results from Section 5.1 still hold for distributions moving with very low speeds, i.e. $s < 1e^{-5}$ standard deviations per sample.

**Overall experimental findings**

Figures 5-8 and 5-5b demonstrate the key points of our experimental findings. We found that Streaming KVB has an advantage over Streaming SVI, because with the same set of hyperparameters KVB outperforms both

- Streaming SVI with learning rate fixed at 1 when the underlying distribution is changing fast
- Streaming SVI with optimal learning rate when the underlying distribution is stationary

The improvement in performance is achieved by using a Kalman filter instead of a Robbins-Monro smoother, and providing the inference algorithm with higher than expected number of desired clusters (Section 5.2).
Chapter 6

Conclusion

In this thesis I presented Streaming SVI and Streaming Kalman Variational Bayes, that are designed to be able to track non-stationary distributions in a streaming setting. Then I compared them with SVB, a streaming algorithm that is designed to be faster and more accurate than SVI for stationary distributions.

We learned from Section 5.1 that SVB is slower to converge than others, and it looses most of it advantage when it is not run in parallel.

Streaming Kalman Variational Bayes and Streaming SVI are closely matched when it comes to stationary distributions. Streaming Kalman Variational Bayes is more error prone, but that can be corrected using multiple seeds for cluster center initialization (Section 5.1.2) and using more potential clusters than the data needs (Section 5.2).

With the above changes Streaming Kalman Variational Bayes performs as well or even better than Streaming SVI for stationary data, and for fast moving distributions it outperforms Streaming SVI, beating even Streaming SVI with the fastest learning rate.
Bibliography


