1. Introduction

In the world of data explosion, designing a model that can capture continuously upcoming data is a basic need. By considering that new data depend on the past data in a probabilistic way, the Bayesian approach is highly suitable to model data streams where the data come sequentially and infinitely. Many researches have been successful in using this idea [1–5]. Through streaming learning methods, a model can be learned in a forward way without revisiting the old data, and hence more efficiently in terms of time and memory space. However, many challenges exist, including extreme sparsity and noisy data.

Sparsity in which the observed data are (extremely) sparse in nature is prevalent in practice, such as user ratings in recommender systems [6,7], and posts/comments from social networks [8–10]. Sparse data contain little information, and thus pose a severe challenge for modeling even in cases of extreme large number of samples [11]. Further, noisy data (e.g., coming from social networks or online forums) with contradicting or irrelevant information present inherent difficulties for modeling [12–16]. Fig. 1 shows two examples where the learnt models seem to encounter severe overfitting as learning from noisy and sparse data. For those challenges, designing an entirely new model is not always possible and possibly takes a high cost. Therefore providing human knowledge is often a good choice to improve a Bayesian model.

Consider the task of learning from a data stream, where the data come sequentially and infinitely. Existing learning methods have difficulties to effectively exploit prior knowledge. Streaming variational Bayes (SVB) [1] can use the external knowledge as initial prior at the first step of the learning process. However, in other steps, the prior is replaced by the posterior learned from the previous learning step. This strategy leads to losing information from the past, as shown in Section 3.2, and limits the effect of external knowledge. Population variational Bayes (PVB) [2] employs a population distribution to capture streaming data. Masegosa et al. [3] develop SVB further to balance the old and new knowledge, all learned from data, in a Bayesian way. Faraji et al. [5] study the same problem as [3], while Theis and Hoffman [19] develop a variant of stochastic variational inference (SVI) [20] for data streams. Due to the ignorance of external knowledge, those studies are limited when we want to exploit valuable knowledge in order to deal with the three challenges above. Note that good prior knowledge are now available in various forms including ontologies, open knowledge bases (e.g., Wikipedia), word embeddings, unsupervised pre-trained models [21,22], Zipf's law, etc.
In this paper, we make the following contributions:

- We show that SVB will quickly forget the prior knowledge through the streaming learning process.
- We then propose a simple framework for boosting prior knowledge in streaming Bayesian learning (BPS), a variant of SVB. Our framework emphasizes the role of the prior by encoding it in every step, therefore the valuable prior can be efficiently exploited in the entire learning process.
- We conduct experiments in four scenarios: streaming learning with synthetic data, streaming learning for unsupervised sentiment analysis [17], streaming learning for latent Dirichlet allocation (LDA) [18], and streaming classification [23, 24]. Comparing with the framework that does not keep prior, BPS gives a significant improvement in performance. We further find that the improvement of BPS over SVB is often with a large margin for short text. This suggests that an appropriate exploitation of prior human knowledge in streaming environments will provide significant benefits, especially for short text and sparse data.

1.1. Further related work

Many researches have gained significant evidences in improving Bayesian models by incorporating prior knowledge. Diamond and Kaul [25] showed the effect of using prior from megatrials in Bayesian analysis for clinical data. Alfaro and Holder [26] used prior in phylogenetics to learn a Bayesian model of biological data.

For text data, by capturing the Zipf's law [27, 28], the performance of topic models can be improved [29] in comparison with LDA [18]. In text classification, Lin et al. [30] used an asymmetric prior which gave high weight for seed words of each class to gain better performance. In sentiment analysis, some studies [17, 31] exploited a word list that consists of a predefined set of positive and negative words in learning Bayesian models to determine the sentiment of each document. However, most of those existing researches focus on static conditions, leaving an unexplored problem of how to exploit human knowledge in streaming environments where the data come sequentially and infinitely. Such an unexplored problem is the object of interest in this work.

AA Another term related to prior is power prior [32, 33]. Power prior is used to balance the old knowledge learned from past data and the new knowledge learned from the current minibatch of data. While the initial prior is gained even when there is no data received, it can be provided from external knowledge that is outside of the observed data. Nonetheless, existing studies [5, 3, 19] did not emphasize in the role of the external prior but concern more on balancing the old and new knowledge which are all learned from data.

Other related topic is stochastic methods for training a Bayesian model from a dataset of big size. Examples include stochastic variational inference (SVI) [20] and memorized online variational inference (MOVI) [34]. The problem setting in our work is learning from a data stream, where the data come sequentially and infinitely. SVI and MOVI seem to be unsuitable for this problem setting. SVI not only requires to know the data size in advance, but also assumes the ability to have multiple learning passes over the whole data. MOVI divides dataset into the fixed number of minibatches and iterates through each minibatch multiple times. Both SVI and MOVI can work well on a big (but fixed) training data. However, in the streaming environment, since the dataset size is infinite and the underlying properties of the data may change over time, iteratively going through all minibatches is intractable. Some works [19, 2] already pointed out those limitations of SVI (and MOVI).

1.2. Roadmap

Section 2 briefly presents some preliminaries with basic notions in Table 1. Section 3 reviews the SVB framework and the problem of vanishing priors. In Section 4, we propose the BPS framework. Section 5 presents case studies in applying SVB and BPS in streaming learning. Finally, Section 6 concludes our work.

2. Background

2.1. Variational inference

Variational inference is an approximate technique to solve Bayesian inference problem [35]. Formally, datum $x$ is assumed to be generated from a Bayesian model with the prior $\eta$, the model parameter $\beta$ and the hidden variable $z$. The task is to find the value of $\beta$ and $z$ to maximize the log likelihood function of the observed data $x$:

$$\log p(x|\eta)$$

In the variational inference, instead of searching for the whole space of $\beta$ and $z$, we seek a variational distribution $q(\beta, z|\epsilon, \gamma)$, which approximates $p(\beta, z)$. We have:

$$\log p(x|\eta) = \log \int_x \int_\beta p(x, \beta, z|\eta) \, d\beta dz$$

$$= E_q[\log p(x, \beta, z|\eta)] - E_q[\log q(\beta, z|\epsilon, \gamma)]$$

$$+ E_q[\log \frac{q(\beta, z|\epsilon, \gamma)}{p(\beta, z|\eta, x)}]$$

(1)

(2)
Assuming that both $\beta$ and $z$ are continuous, otherwise integration is replaced by summation.

Note that the element in (2) is the Kullback Leibler (KL) divergence [36] between the approximate variational distribution $q(\beta, z|\epsilon, \gamma)$ and the true distribution $p(\beta, z|\eta, x)$ which has $KL(q||p) \geq 0$. Now denoting the element in (1) by $L(\epsilon, \gamma; \eta, x)$, we get:

$$
\log p(x|\eta) = L(\epsilon, \gamma; \eta, x) + KL(q||p)
$$

so that:

$$
\log p(x|\eta) \geq L(\epsilon, \gamma; \eta, x)
$$

$L(\epsilon, \gamma; \eta, x)$ is a lower bound of the likelihood function. Maximizing the lower bound $L$ is equal to minimizing the KL distance between the variational distribution $q$ and the posterior $p$. The form of $q$ is selected in order to make $L$ tractable or easier to optimize than the original likelihood function. The solution $q$ is an approximation for the posterior $p$; hence:

$$
p(\beta, z|\eta, x) \approx q(\beta, z|\epsilon, \gamma)
$$

### 2.2. Additive property of the prior of exponential family

In probability and statistics, exponential family is a popular class of distributions that subsumes many common distributions including Bernoulli, Poisson, normal, gamma, multinomial and Dirichlet distributions (Table 2). Their density function is of the form:

$$
f_x(x|\eta) = h(x) \exp(\eta \cdot T(x) - A(\eta))
$$

where $\eta$ is the natural parameter, $T(x)$ is the sufficient statistics, $h(x)$ is a known function, and $A(\eta)$ is a normalizing element.

Consider an observed datum $x$ generated from an exponential distribution with prior which is encoded in the natural parameter $\eta$, and we know one more information that $x$ is also contributed independently by another prior $\tilde{\eta}$ in the same family of distribution, see Fig. 2. We have:

$$
p(x|\eta, \tilde{\eta}) \propto p(x|\eta)p(x|\tilde{\eta})
$$

$$
\propto h(x)h(x)\exp((\eta + \tilde{\eta}) \cdot T(x))
$$

which has $f_{x}(x|\eta + \tilde{\eta})$ as its density function. The new prior for $x$ has the natural parameter that is the sum of the two natural parameters: $\eta$ and $\tilde{\eta}$, this is the additive property of exponential family.

### 2.3. Latent Dirichlet allocation

We briefly describe latent Dirichlet allocation (LDA) [18], a well-known model, which will be referred in several parts later.

Formally, LDA (Fig. 3) is a generative model for modeling text data. It assumes that a corpus $C$ containing $M$ documents is composed from $K$ topics of $V$ words, $\beta=(\beta_1, \ldots, \beta_K)$, each of which is drawn from a $V$-dimensional Dirichlet distribution: $\beta_k \sim \text{Dirichlet} (\eta)$. Each document $d$ is a mixture of those topics and is presumed to be created by the following generative process:

1. Draw topic mixture $\theta |x \sim \text{Dirichlet} (\chi)$
2. For the $i^{th}$ word in document $d$:
   - Draw topic index $z_i | \theta \sim \text{Multinomial} (\theta)$
   - Draw word $w_i | z_i, \beta_k \sim \text{Multinomial} (\beta_{z_i})$

The learning problem of LDA is to estimate the posterior $p(\theta, z, \beta|C, x, \eta)$ of the latent variables. Nonetheless, this problem is intractable. Applying variational inference, we approximate $p(\theta, z, \beta|C, x, \eta) \approx q(\theta, z, \beta|\gamma, \phi, \lambda) = q(\theta|\gamma)q(z|\phi)q(\beta|\lambda)$ where $\gamma, \phi, \lambda$ are the parameters of the following variational distributions:

$$
q(\theta) = \text{Dirichlet} (\theta|\gamma)
$$

$$
q(\beta) = \text{Multinomial} (\beta|\lambda)
$$

$$
q(z) = \text{Multinomial} (z|\phi)
$$

and $\sum_{\phi, \lambda} = 1$. According to [18], we have the lower bound of the log likelihood of $C$:

$$
L = \sum_d \left( E_q [\log p(w_d, \theta_d, z, \beta)] - E_q [\log q(\gamma, \phi, \lambda)] \right)
$$

$$
= \sum_d \left( E_q [\log p(w_d|\theta_d, \beta)] + E_q [\log p(\theta_d|\phi)] 
- E_q [\log q(z|\phi)] + E_q [\log p(\phi|\lambda)] - E_q [\log q(\beta|\lambda)] \right)
$$

Taking partial derivative of $L$ with respect to $\phi$ and $\gamma$ and set them to zero, we have the following inference equations for local variables:

### Table 1

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>a data point</td>
</tr>
<tr>
<td>$C_i$</td>
<td>a minibatch – a collection of data points</td>
</tr>
<tr>
<td>$\beta, \Phi$</td>
<td>model’s parameters</td>
</tr>
<tr>
<td>$\eta$</td>
<td>prior</td>
</tr>
<tr>
<td>$</td>
<td>\eta</td>
</tr>
<tr>
<td>$\epsilon, \gamma$</td>
<td>variational parameter symbols</td>
</tr>
<tr>
<td>$\zeta_i$</td>
<td>natural parameter of an exponential family</td>
</tr>
<tr>
<td>$i, d$</td>
<td>learned information of a minibatch $i$</td>
</tr>
<tr>
<td>$s$</td>
<td>independent and identically distributed</td>
</tr>
<tr>
<td>$r$</td>
<td>scale ratio</td>
</tr>
<tr>
<td>$b$</td>
<td>boosting rate</td>
</tr>
</tbody>
</table>

### Table 2

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Parameters</th>
<th>Natural parameter $\eta$</th>
<th>Sufficient statistic $T(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bernoulli</td>
<td>$p$</td>
<td>$\ln \frac{p}{1-p}$</td>
<td>$x$</td>
</tr>
<tr>
<td>Poisson</td>
<td>$\lambda$</td>
<td>$\ln \lambda$</td>
<td>$x$</td>
</tr>
<tr>
<td>Normal</td>
<td>$\mu, \sigma^2$</td>
<td>$\frac{\mu}{\sigma^2}$</td>
<td>$x^2$</td>
</tr>
<tr>
<td>Gamma</td>
<td>$\alpha, \beta$</td>
<td>$\frac{\alpha - 1}{\beta}$</td>
<td>$</td>
</tr>
<tr>
<td>Multinomial</td>
<td>$p_1, \ldots, p_k$</td>
<td>$\ln p_i$</td>
<td>$x_i$</td>
</tr>
<tr>
<td>Dirichlet</td>
<td>$\alpha_1, \ldots, \alpha_k$</td>
<td>$\ln \pi_1$</td>
<td>$\ln x_1$</td>
</tr>
</tbody>
</table>
3. Revisiting streaming variational Bayes

In this section, we briefly review the streaming variational Bayes framework (SVB) [1] for learning a Bayesian model from a data stream, then discuss the problem of vanishing prior in this framework.

3.1. Streaming variational Bayes

Streaming data is considered as a sequence of minibatches \( \{C_i\}_{i=1}^b \) where \( b \) can be infinite. Each instance is assumed to be generated from a Bayesian model with parameter \( \Phi \). The learning problem is to find \( \langle \Phi \rangle \) that maximizes the posterior:

\[
p(\Phi|C_1, C_2, \ldots, C_b)
\]

There is a general framework for computing the posterior in Eq. 9. The idea is to use the prior from the past data and information from the current data. Fig. 4 provides a graphical representation of their idea.

Given the prior \( \eta \), presuming that \( b-1 \) minibatches have been processed, the posterior after \( b \) minibatches can be calculated using Bayes rule:

\[
p(\Phi|C_1, C_2, \ldots, C_b, \eta)
\]

\[
\propto p(C_b|\Phi, \eta)p(\Phi|C_1, C_2, \ldots, C_{b-1}, \eta)
\]

\[
\propto \frac{p(\Phi|C_b, \eta)\cdot p(\Phi|C_1, C_2, \ldots, C_{b-1}, \eta)}{p(\Phi|\eta)}
\]

Which means that the posterior after \( b \) minibatches is proportional to the product of the likelihood of current data \( p(C_b|\Phi, \eta) \) and its prior \( p(\Phi|C_1, C_2, \ldots, C_{b-1}, \eta) \). Note that the prior element is also the posterior of \( b-1 \) minibatches, so the posterior can be computed sequentially. Unfortunately, the posterior in Eq. 11 is often intractable to precisely compute. This problem leads to the need for an approximation method. Using the variational inference, we approximate:

\[
p(\Phi|C) \approx q(\Phi|\xi) \propto \exp(\xi \cdot T(\Phi))
\]

and:

\[
p(\Phi|C_1, C_2, \ldots, C_{b-1}, \eta) \approx q(\Phi|\xi_{b-1})
\]

\[
p(\Phi|C_b, \eta) \approx q(\Phi|\xi_b)
\]

Assume that the initial prior \( p(\Phi|\eta) = q(\Phi|\xi_0) \). Here the prior \( \eta \) is transformed to a natural parameter of an exponential family: \( \xi_0 \rightarrow \eta \), so \( \xi_0 \) is also called as the initial prior knowledge. Now combining these approximates with Eq. 12, we have:

\[
q(\Phi|\xi_b) \approx \frac{q(\Phi|\xi_b)}{q(\Phi|\xi_0)}q(\Phi|\xi_{b-1})
\]

Taking the log function for both sides and use (13), we obtain

\[
\xi_b = (\xi_b - \xi_0) + \xi_{b-1}
\]

Denote:

\[
\tilde{\xi}_b = \xi_b - \xi_0 = \xi_b
\]

We have:

\[
\xi_0 = \tilde{\xi}_0 = \xi_0 + \xi_{b-1} + \ldots + \xi_1 + \xi_0
\]

Eq. 19 is the streaming update function of the SVB framework [1]. It shows that the current model's parameter is equal to the sum of the past parameter \( \xi_{b-1} \) with the learned information \( \tilde{\xi}_b \) from current data, and hence SVB provides us a principled way to model streaming data without revisiting past data.

3.2. The problem of vanishing prior

A drawback in the SVB framework is the vanishing prior problem. In other words, the information from the prior is probably vanished through learning process. This subsection will show this
phenomenon for LDA model in particular, and for any model learned by SVB.

Replacing Eq. 18 with \( \tilde{z}_0 = \eta \) and \( \tilde{\gamma} = \lambda \), we have the following for the \( b \)th minibatch:

\[
\tilde{z}_{b\ell} = \tilde{z}_{b\ell} - \tilde{z}_{a\ell} = \sum_{d \in s} \phi_{d\ell} \eta_{d\ell}.
\]

As a result, the prior \( \tilde{z}_0 \) in SVB is used only one time in the initial stage as in Eq. 20. It is clear that the learned parameters \( \tilde{z}_b \) represent the learned information from minibatch data. \( \tilde{z}_1 + \ldots + \tilde{z}_b \) represents the total information learned from the data, and \( \tilde{z}_b \) represents the information from the prior.

We will analyze the ratio between the norms of information learned from the data and the information from the prior.

**Lemma 3.1 (LDA).** The SVB update of the LDA model has the following property:

\[
||\tilde{z}_b - \tilde{z}_0|| = ||\tilde{z}_1 + \ldots + \tilde{z}_b|| \rightarrow +\infty \text{ as } b \rightarrow +\infty,
\]

suggesting that as \( b \) increases, the information from the prior \( \tilde{z}_0 \) quickly becomes vanishing in comparison with the learned information \( \tilde{z}_1 + \ldots + \tilde{z}_b \) from data.

**Proof.** It is clear that \( ||\tilde{z}_0|| \) is a constant. We now prove \( ||\tilde{z}_1 + \ldots + \tilde{z}_b|| \rightarrow +\infty \text{ as } b \rightarrow +\infty \). Because \( \sum_d \phi_{d\ell} = 1 \) and \( \tilde{z}_{b\ell} \neq 0 \), from Eq. 21 we have the following for any \( b \geq 1 \):

\[
||\tilde{z}_b|| = \sum_{d \in s} \sum_{\ell} \tilde{z}_{b\ell} = \sum_{d \in s} \sum_{\ell} \phi_{d\ell} \eta_{d\ell} = \sum_{d \in s} \eta_{d\ell} = 1.
\]

So

\[
||\tilde{z}_1 + \ldots + \tilde{z}_b|| \geq b,
\]

which means \( ||\tilde{z}_1 + \ldots + \tilde{z}_b|| \rightarrow +\infty \text{ as } b \rightarrow +\infty \). \( \square \)

The result in Lemma 3.1 can be interpreted as following: In SVB with the initial prior only exists in the first stage, through the stream learning process, the new knowledge is added into the model’s parameter. When data is big enough, almost all information in the model’s parameter comes from the data and the role of the initial prior will vanish quickly. This problem can be easily seen in LDA. Next, we discuss the vanishing prior phenomenon of SVB in broader contexts. For more complex models, such a phenomenon is not easily observed. We have to employ the well-known law of large numbers under some assumptions. Formally, we have the following.

**Lemma 3.2 (Other models).** Assuming that the learned information \( \{\tilde{z}_b\}_{b=1}^b \) are i.i.d samples from a probability distribution with mean \( \tilde{z} \neq 0 \), we have the following property for SVB with probability 1:

\[
||\tilde{z}_b - \tilde{z}_0|| = ||\tilde{z}_1 + \ldots + \tilde{z}_b|| \rightarrow +\infty \text{ as } b \rightarrow +\infty,
\]

suggesting that as \( b \) increases, the prior \( \tilde{z}_0 \) quickly becomes vanishing in comparison with the learned information \( \tilde{z}_1 + \ldots + \tilde{z}_b \) from data.

**Proof.** Using the law of large numbers, we have:

\[
\Pr \left( \lim_{{b \rightarrow +\infty}} \frac{\tilde{z}_1 + \ldots + \tilde{z}_b}{b} = \frac{\tilde{z}}{b} \right) = 1,
\]

suggesting that

\[
\Pr \left( \lim_{{b \rightarrow +\infty}} \frac{||\tilde{z}_1 + \ldots + \tilde{z}_b||}{||\tilde{z}_0||} = 1 \right) = 1,
\]

and because \( ||\tilde{z}_0|| \) is a constant, we have:

\[
\Pr \left( \lim_{{b \rightarrow +\infty}} \frac{||\tilde{z}_1 + \ldots + \tilde{z}_b||}{||\tilde{z}_0||} = +\infty \right) = 1.
\]

\( \square \)

This lemma shows that, in general using SVB in streaming learning will lead to the problem of losing the prior information. Although the assumption of Lemma 3.2 is not always met, the result provides a significant message for practice of streaming learning, especially when we have valuable prior knowledge about the domain/task of interest.

Considering the circumstances that we have valuable prior human knowledge that describes properties of data, losing the information of these priors in the learning process is highly wasteful. Bayesian methods often assume the data to be generated from a probabilistic process. If no knowledge is present, those methods can learn information efficiently only from data. However, note that most models are mis-specified, and as a consequence, might not model the data exactly. From this reason, combining prior human knowledge into a Bayesian model is an essential method to improve the quality of Bayesian models.

To overcome the problem of vanishing prior in SVB, we propose a framework for Bayesian learning to maintain the valuable information from the initial prior.

**4. Boosting prior knowledge in streaming Bayesian learning**

In this section, we first present the BPS framework for boosting the valuable prior knowledge in SVB. We then propose to choose the linear boosting function and analyze some advantages of such a choice. Finally, we discuss how to balance between boosted prior and the knowledge learned from each minibatch of data.

**4.1. The BPS framework**

The idea of BPS is to directly add one more prior information \( \eta \) to each minibatch. Fig. 5 depicts this idea clearly. The new prior \( \eta_b \) of the minibatch \( b \) is a boosting function of the initial prior \( \eta \) so that it reflects how the prior knowledge impacts to each minibatch of the streaming learning process.

\[
\eta_b = f_b(\eta)
\]
An additional prior in the form of an exponential family \( \tilde{p}(\eta|\tilde{\eta}_k) \propto \exp(\eta \cdot T(\Phi)) \) is used in each minibatch. To emphasize the role of the new prior \( \tilde{\eta}_k \), we will optimize the following probability:

\[
p(\Phi|\tilde{\eta}_k)p(C_k|\Phi, \eta)p(\Phi|C_1, C_2, \ldots, C_{b-1}, \eta) \propto \frac{p(\Phi|\tilde{\eta}_k)p(\Phi,C_k, \eta)}{p(\Phi|\eta)}
\]

With \( \hat{\tilde{\eta}}_b \leftarrow \tilde{\eta}_k \), using the additive property of prior for exponential family and the local minimal, we have an approximation:

\[
p(\Phi|\tilde{\eta}_k)p(\Phi,C_k, \eta) \propto p(\Phi|C_0, \tilde{\eta}_k + \eta)
\]

Replacing Eqs. (32)–(30), we have the new update function:

\[
\begin{align*}
\hat{\tilde{\eta}}_b & \leftarrow (\hat{\tilde{\eta}}_b - \tilde{\eta}_0) + \hat{\tilde{\eta}}_{b-1} \\
\ &= \hat{\tilde{\eta}}_b + (\tilde{\eta}_b + \hat{\tilde{\eta}}_{b-1})
\end{align*}
\]

The difference between the streaming update function in Eq. 28 of BPS and the one of SVB in Eq. 19 is that there is an additional element of the boosting prior \( \hat{\tilde{\eta}}_b \). This property means that BPS boosts the knowledge from the initial prior through streaming learning process. If we set \( \tilde{\eta}_k = 0 \), the BPS will become SVB.

### 4.2. Linear boosting function

In BPS, we introduce a boosting function in Eq. 28 that describes how the original prior impacts on the learned model. Now we propose a simple form of the boosting function in term of linear relationship with a boosting rate \( r_b \in \mathbb{R} \):

\[
\tilde{\eta}_b = \hat{\tilde{\eta}}_b + r_b
\]

Assuming \( \eta \) is the natural parameter in the exponential form of prior knowledge \( p(\Phi|\eta) \propto \exp(\eta \cdot T(\Phi)) \), we will show that with the linear function in Eq. 35, the new prior \( p(\Phi|\tilde{\eta}_b) \) will keep some main features of the original prior distribution with a level of importance.

**Definition:** The feature points \( \Phi \) of a probability distribution \( p(\Phi|\eta) \) are defined by its local extrema.

The feature points \( \Phi \) can be used to describe the shape of the prior distribution, containing stationary points and boundary points. The local maximal points are positions that have high density, and the local minimal points have low density.

Taking derivative of the density functions of \( p(\Phi|\eta) \) and \( p(\Phi|\tilde{\eta}_b) \) over \( \Phi \), we have:

\[
g_{\eta}^\prime = \frac{\partial(\exp(\eta \cdot T(\Phi)))}{\partial(\Phi)} = \exp(\eta \cdot T(\Phi)) \frac{\partial(\eta \cdot T(\Phi))}{\partial(\Phi)}
\]

and

\[
g_{\tilde{\eta}_b}^\prime = \frac{\partial(\exp(\tilde{\eta}_b \cdot T(\Phi)))}{\partial(\Phi)} = \exp(\eta \cdot T(\Phi)) \frac{\partial(\eta \cdot T(\Phi))}{\partial(\Phi)}
\]

Obviously, \( g_{\eta}^\prime = 0 \) and \( g_{\tilde{\eta}_b}^\prime = 0 \) have the same set of solutions which means both \( p(\Phi|\eta) \) and \( p(\Phi|\tilde{\eta}_b) \) have the same feature points \( \Phi \).

In another word, we interpret the value of \( g_{\eta} \) near a feature point as the changing of the concentrated density around the feature point. The big value of \( g_{\eta} \) means that there is a high concentrated density level or a high important level.

Assuming \( \eta \cdot T(\Phi) > 0 \), it is clear that if \( r < 1 \), then \( |g_{\tilde{\eta}_b}^\prime| < |g_{\eta}^\prime| \), we decrease the important level of feature points. Otherwise, if \( r > 1 \), then \( |g_{\tilde{\eta}_b}^\prime| > |g_{\eta}^\prime| \), we increase the important level of feature points. In another words, by changing the value of boosting rate in the linear boosting function, we can change the important level of feature points of the prior distribution.

To illustrate this property, we consider an example with Dirichlet distribution on the 2-simplex \( x_1, x_2, x_3 \):

\[
p(x_1, x_2, x_3) \propto \exp \left( \sum_{i=1}^{3} \eta_i \ln(x_i) - \sum_{i=1}^{3} \ln \Gamma(\eta_i) - \ln \Gamma(\sum_{i=1}^{3} \eta_i) \right)
\]

In this example, the natural parameter \( \eta = [4.5.6] \), and the boosting rates are tested with: 0.5, 1 and 1.5 respectively. The results are shown in Fig. 6. Intuitively, the figure shows that the bigger boosting rate, the more concentrated level of the density.

#### 4.2.1. Relation to power prior

In BPS, we observe that \( \exp(\eta \cdot T(\Phi)) = [\exp(\eta \cdot T(\Phi))]^\prime \). This is equivalent to the use of the variational distribution of the following form

\[
q(\Phi|\eta) = [q(\Phi|\eta)]^\prime
\]

In fact, linear boosting means that we use a power scale of the initial prior. This concept is related with the power prior method in [32, 33], but BPS has some significant differences.

In power prior, presume the model’s parameter \( \Phi \) has the initial uninformative prior \( p(\Phi|\eta) \), a past data \( C_0 \) and a power factor \( \rho \). The power prior is defined by:

\[
p(\Phi|C_0, \eta) \propto \text{Likelihood}(\Phi|C_0)^\rho \cdot \text{Prior}(\Phi|\eta)
\]

\[
= p(C_0|\Phi)^\rho \cdot p(\Phi|\eta)
\]

Receiving new data \( C \) leads to the posterior:

\[
p(\Phi|C, C_0, \eta) \propto p(C|\Phi) \cdot p(\Phi|C_0, \eta)
\]

\[
= p(C|\Phi) \cdot p(C_0|\Phi)^\rho \cdot p(\Phi|\eta)
\]
Fig. 6. Dirichlet distributions on the 2-simplex with different boosting rates of natural parameter $\eta = [4.5.6]$.

contrast, the viewpoint of BPS is to inject human/external knowledge into a model at each time step.

4.3. Keeping prior in BPS

We will show that in BPS with linear boosting functions, information from prior is not vanished. In BPS, the information learned from the data is: $\tilde{z}_1 + \ldots + \tilde{z}_b$ and the total information from prior is $\zeta_0 + \eta_1 + \ldots + \eta_b$. By evaluating ratio of the norms of these two kinds of information, we have the following lemma:

**Lemma 4.1 (BPS).** Assume that the learned information $\{\tilde{z}_h\}_{h=1}^b$ are i.i.d samples from a probability distribution with mean $\tilde{z} \neq 0$, and BPS uses the linear boosting function $\eta_b = r_b \tilde{z}_0$, where coefficients $r_b$ are lower bounded by some constant $c > 0$. We have the following property, with probability 1,

$$\frac{\|\tilde{z}_1 + \ldots + \tilde{z}_b\|}{\|\zeta_0 + \eta_1 + \ldots + \eta_b\|} \leq \frac{\|\tilde{z}\|}{c\|\zeta_0\|} \quad \text{as} \quad b \to +\infty,$$

(40)

suggesting that as $b$ increases, the prior information $\zeta_0 + \eta_1 + \ldots + \eta_b$ is not vanished in comparison with the learned information $\tilde{z}_1 + \ldots + \tilde{z}_b$ from data.

**Proof.** Using the law of large numbers, we have:

$$\Pr\left(\lim_{b \to +\infty} \frac{\tilde{z}_1 + \ldots + \tilde{z}_b}{b} = \tilde{z}\right) = 1,$$

and because $\|\zeta_0\|$ is a constant:

$$\Pr\left(\lim_{b \to +\infty} \frac{\|\tilde{z}_1 + \ldots + \tilde{z}_b\|}{bc\|\zeta_0\|} = \frac{\|\tilde{z}\|}{c\|\zeta_0\|}\right) = 1.$$

Note that $\bar{\eta}_b = r_b \bar{\zeta}_0$ and $r_b \geq c$ for any $b$, we have:

$$\Pr\left(\lim_{b \to +\infty} \frac{\|\tilde{z}_1 + \ldots + \tilde{z}_b\|}{\|\zeta_0 + \bar{\eta}_1 + \ldots + \bar{\eta}_b\|} \leq \frac{\|\tilde{z}\|}{c\|\zeta_0\|}\right) = 1.$$

This lemma shows that, in BPS, the information from external prior is not overwhelmed by the information learned from data. Hence, we can remain the valuable knowledge.

4.4. Balancing between learned information and boosting prior

BPS aims to inject the information from the initial prior into the new data. In Eq. 34, the learned information from the new data $\tilde{z}_b$ is independent of $\zeta_0$ and $\bar{\zeta}_0$. Because $\tilde{z}_b$ depends on the amount of data, oversusing the boosting prior may lead to underfitting because it dominates the learned information. To tackle this problem, we propose an information scale ratio between the learned information ($\tilde{z}_b$) and the prior knowledge ($\zeta_0$):

$$\frac{\|\tilde{z}_b\|}{c\|\zeta_0\|} = s \frac{\|\tilde{z}\|}{\|\zeta_0\|},$$

(41)

where $s$ is a scale parameter. The purpose of this balancing is to let the information from the prior knowledge account for a certain ratio in the learned information of the new data.

Suppose we use the linear boosting function in Eq. 35. Note that $\zeta_0 = \eta$ and $\tilde{z}_0 = \tilde{\eta}^b$, combining with Eq. 41 leads to the boosting rate:

$$r_b = s \frac{\|\tilde{z}_b\|}{\|\zeta_0\|},$$

(42)

and the boosting prior:

$$\frac{\tilde{z}_b}{\zeta_0} = s \frac{\|\tilde{z}\|}{\|\zeta_0\|} \frac{\tilde{z}}{\zeta_0},$$

(43)

In practice, at first we compute the learned information $\tilde{z}_b$, then we follow Eq. 43 to get the boosting prior. Finally, we update the parameter using Eq. 34. In BPS, $s$ is a hyperparameter, we can choose a good value for $s$ by grid search in the range $[0, 1]$.

Note that with scale factor $s = 0$, BPS turns out to be SVB without using boosting prior.

4.5. Discussion about BPS

BPS is a generalization of SVB with the ability to appropriately use prior knowledge. While a self-contained model to cover the data with high performance is not available and we often spend a high cost on deriving a good model, BPS is a considerable solution. In cases of having a good prior knowledge, BPS is a straightforward way with low cost to enhance a Bayesian model and reduce some bad effects from extreme sparsity and noise. One can also interpret that BPS uses external knowledge to regularize a model when dealing with ill-posed problems, e.g., sparsity and noise. We observe that external/human knowledge is available in various forms and easily accessible, including pre-trained machine learning models [22,37,38], ontology, Wikipedia, Zipf’s law, Wordnet, etc. Furthermore, we believe that the idea of BPS can be easily
employed in other streaming frameworks [2,19,35]. Those observations suggest the solution from BPS is significant.

Nonetheless, BPS requires a good prior knowledge to improve a model, otherwise, it may worsen the model. This consequence is explicitly shown in a case study of Section 5. In some situations, obtaining prior knowledge with good quality may be hard, and therefore BPS has some limitations.

5. Case studies

In this section, at first we conduct a case study with synthetic streaming data and then we build 3 other case studies with 3 models and different type of prior knowledge to evaluate the performance of BPS framework in comparison with SVB. The streaming data is simulated by dividing a dataset into continuous collections. For selecting models' hyperparameters, we do a grid search around their recommended settings in its original papers. For BPS, we first do a grid search to find a good range of scale factor s then evaluate how BPS responses to the change of s. We also do experiments on the sensitivity of BPS over different values of the batchsize parameter. Each experiment is run five times with random initialization of variables. The final result is the mean value of all runs. For simplicity, we only plot standard deviation of experiments in several cases. In all cases, higher values are better.

5.1. Case study 1: streaming with synthetic data

In this case study, we examine the performance of BPS in comparison with SVB in a synthetic streaming data, and investigate the impacts of the quality of prior knowledge and noise.

Consider a model where the binary data are all generated from a Bernoulli distribution: \( x \sim \text{Bernoulli}(\theta) \). We fix \( \theta = 0.2 \) and generate a sequence of 300 data points.

Suppose that we use the following misspecified model to work with those data: \( \theta = \text{Beta}(\gamma^p, \eta^p); x \sim \text{Bernoulli}(\hat{\theta}) \). Note \( E[\theta] = \frac{\gamma^p}{\gamma^p + \eta^p} \), where \( \gamma^p \) and \( \eta^p \) are hyperparameters.

This model is denoted as \( \text{BB} \). The learning involves the following posterior: \( p(\theta|\gamma^p, \eta^p, C) \). Denote \( n_b \) and \( k_b \) be the number of data points and the number of 1’s in the data \( C_b \) of minibatch \( b \), respectively. Using variational inference with the variational distribution \( q(\theta) \sim \text{Beta}(\tilde{\gamma}, \tilde{\eta}) \), a lower bound of the log likelihood of the data is:

\[
L = E_q \log p(\theta, C|\gamma^p, \eta^p) - E_q \log q
\]

Similar to the learning method of LDA, by taking the partial derivative of \( L \) with respect to \( \tilde{\gamma} \) and \( \tilde{\eta} \) and setting them to zero, we have Algorithm 3 and Algorithm 4 for learning \( \text{BB} \) by SVB and BPS respectively.

5.1.1. Impact of prior knowledge quality

We evaluate the impact of prior knowledge quality into the performance of BPS by setting different levels of prior knowledge. The knowledge to be used in BPS is encoded in \( (\gamma^p, \eta^p) \). Note that \( \frac{\gamma^p}{\gamma^p + \eta^p} \) being closer to \( \theta \) = 0.2 means that the prior quality is better.

**Settings:** The batchsize is fixed to 20, and the scale factor of BPS is \( s = 0.3 \).

**Evaluation:** We compare the expectation of \( \theta \) in the learned model with its true value. The closer \( \hat{\theta} \) is to the true value, the better the model is.

**Prior in use:** We investigate different priors \( (\gamma^p, \eta^p) \) so that \( \hat{\theta} = \frac{\gamma^p}{\gamma^p + \eta^p} \in \{0.5, 0.4, 0.3, 0.2, 0.1\} \). The closer \( \hat{\theta} \) to the truth \( \theta = 0.2 \), the better the quality of prior knowledge.

**Result:** The result is illustrated in Fig. 7a for SVB and in Fig. 7b for BPS. It shows that SVB has less impact of prior knowledge while BPS has a strong dependence with the quality of prior knowledge. A good prior knowledge will significantly enhance the quality of model. In contrast, a poor quality of prior knowledge misleads the model, and even makes it poorer than SVB. Therefore, the quality of prior knowledge has a real impact to BPS, and BPS only keeps its good performance in case of having a good prior.

5.1.2. Learning from noisy data

Suppose that some random noises change the results of Bernoulli trials. In more details, the noisy generative process for each data point is:

- Sample data point \( x \sim \text{Bernoulli}(\theta) \).
- Change the value of \( x \) with a noise \( \text{Bernoulli}(\delta) \).\(^2\)

With the same settings as sub-Section 5.1.1, but only using the good prior with the expectation \( E[\theta] = 0.2 \) and the noisy probability \( \delta \in \{0.01, 0.05, 0.1, 0.2, 0.3\} \), we have the results in Fig. 8a for SVB and in Fig. 8b for BPS.

It is clear that in noisy data, BPS with a good prior knowledge will drive the model into its true value. Meanwhile, SVB is impacted by wrong information from noise leading to a weak performance. In other words, BPS has ability to help the model cope with noisy information when providing a good prior knowledge.

\[^2\] Note that the contaminated \( x \) turns out to be a trial from \( \text{Bernoulli}(\theta + \delta - 2\delta) \).
Impact of noise. 'True' is the correct value of model parameter \( \theta \) in the BB model that generates the training data. 


Algorithm 4 BPS-BB

Input: Prior \( \eta^f, \eta^d \), sequence of mini-batch \( C_1, C_2, \ldots \)

Output: a sequence \( \lambda_b^f, \lambda_b^d \)

Initialize: \( \lambda_b^f \leftarrow \eta^f, \lambda_b^d \leftarrow \eta^d \)

for each mini-batch \( C_b \) in \( C_1, C_2, \ldots \) do

\( \lambda_b^f \leftarrow k_b \)

\( \lambda_b^d \leftarrow -n_b - k_b \)

\( \lambda_b^f \leftarrow \lambda_b^f - 1 + \lambda_b^d \)

\( \lambda_b^d \leftarrow \lambda_b^d - 1 + \lambda_b^f \)

end for

5.2. Case study 2: streaming sentiment analysis

Sentiment analysis is an interesting area in text mining, with the aim to extract opinions from text data. One of the basic tasks in sentiment analysis is sentiment classification, in which opinions often need to be classified into positive, negative or neutral. However, a document can contains many aspects with different sentiments. Yo and Oh [17] proposed the Aspect and Sentiment Unification model (ASUM) for modeling the sentiments about aspects in an unsupervised manner. The graphical model of ASUM is presented in Fig. 9. ASUM assumes that a corpus is talking about \( E \) sentiments and \( T \) aspects, represented by many Dirichlet distributions \( \beta_e \sim \text{Dir}(\eta_e) \), where \( e \in \{1, \ldots, E\} \) is the index of a sentiment and \( z \in \{1, \ldots, T\} \) is the index of an aspect. In each document, containing \( M \) sentences, the distribution of sentiments is \( \pi_e \sim \text{Dir}(\gamma) \); and for each sentiment \( e \), the aspect distribution is defined by \( \theta_{e,z} \sim \text{Dir}(\alpha) \). We assume the \( m \)th sentence of length \( N \) is generated as follow: first choose the sentiment by a multinomial distribution \( \epsilon_m \sim \text{Mul}(\pi_e) \), then generate the corresponding aspect \( z_m \sim \text{Mul}(\theta_{e,z}) \), generate each word \( w \sim \text{Mul}(\beta_{e,w}) \).

Learning algorithm: Given data \( C \), the full posterior of interest is:

\[
p(\beta, \pi, \theta, e, z|C, \gamma, \alpha, \eta) \quad (44)
\]

The inference procedure infers the local variables \( \pi, \theta, e, z \). We use the method proposed by [39]. The Dirichlet parameters \( \gamma = 1 \) and \( \alpha = 1 \) are fixed so that the posterior in Eq. 44 is a convex function with respect to \( \pi \) and \( \theta \) and can be efficiently inferred by Frank-Wolfe algorithm [40], \( e \) and \( z \) can be directly approximated through \( p(\epsilon|\pi, \theta, w, \beta) \) and \( p(z|\pi, \theta, w, \beta) \). The details of this procedure is presented in Algorithm 12 of the Appendix.

To update the global variable \( \beta \), let \( q(\beta|\zeta) \) be the variational distribution of \( p(\beta|\zeta) \). Using variational inference, we derive the two streaming learning algorithms for ASUM, SVB in Algorithm 5 and BPS in Algorithm 6.

Prior knowledge in use: We use the same way as in [17,39]. We use a list of positive (e.g., “good”, “excellent”) and negative words (e.g., “bad”, “poor”) as prior knowledge, often called seed words. \( \beta_e \) contains information about the distribution of words over each sentiment. So for the seed words, the priors are set with higher value than other words. By this way, we form a human prior of sentiment over words: \( \eta_e \). Therefore, the prior encoded in BPS is

\[
\tilde{f}_b(\eta) = s \frac{||\lambda_b||}{||\eta||} \eta.
\]

We set \( \eta_{ij} = 0.01 \) for any sentiment seed word \( j \) and otherwise \( \eta_e = 0.0001 \) for any other word \( i \). \( \eta_e \) is used for both SVB at the first learning step and BPS.

Evaluation metric: The accuracy of sentiment classification is used to evaluate the quality.

Experimental setups: We use 4 datasets (Electronics, Yelp, Kitchen & Housewares, and Music)\(^3\) with some statistics shown in Table 3.

To investigate the effect of scale ratio \( s \), we fix batch-size to 1000 for the three first datasets, and to 2000 for Music as it’s size is large. We test \( s \in \{0.2, 0.4, 0.6, 1.0\} \). To study the effect of batch-size, we fix the scale ratio to \( s = 0.4 \) and change batch-size in \( \{500, 1500, 2000\} \) for the three small corpora, and in \( \{4000, 6000, 8000\} \) for Music. We have \( E = 2 \) to represent

\(^3\) The full list can be found in [39].

\(^4\) http://www.cs.fiu.edu/~mdredze/datasets/sentiment/
positive/negative sentiments, and fix $T = 50$ in all experiments for ASUM.\(^5\)

**Algorithm 5 SVB-ASUM**

- **Input:** Prior $\eta$, $\alpha$, $\gamma$, sequence of mini-batch $C_1, C_2, \ldots$
- **Output:** a sequence $\lambda_1^{(1)}, \lambda_2^{(1)}$ ...
- **Initialize:** $\lambda_0 \leftarrow \eta$
- **for** each mini-batch $C$ in $C_1, C_2, \ldots$ **do**
  - **Inference:**
    - for each document $d$ in $C$ **do**
      - Infer local variables $(\theta, \pi, e, z)$
    - **end for**
  - **Update:**
    - $\tilde{\lambda}_{beg} = \sum_{d \in C} \sum_{m} \sum_{n} \tilde{\beta}_{bmn} \tilde{z}_{dmn}$
    - $\tilde{\lambda}_{b} \leftarrow \tilde{\lambda}_{b-1} + \tilde{\lambda}_{b}$
    - $\beta_{b} \propto \tilde{\lambda}_{b}$
- **end for**

---

**Algorithm 6 BPS-ASUM**

- **Input:** Prior $\eta$, $\alpha$, $\gamma$, sequence of mini-batch $C_1, C_2, \ldots$
- **Output:** a sequence $\lambda_1^{(1)}, \lambda_2^{(1)}$ ...
- **Initialize:** $\lambda_0 \leftarrow \eta$
- **for** each mini-batch $C$ in $C_1, C_2, \ldots$ **do**
  - **Inference:**
    - for each document $d$ in $C$ **do**
      - Infer local variables $(\theta, \pi, e, z)$
    - **end for**
  - **Update:**
    - $\tilde{\lambda}_{beg} = \sum_{d \in C} \sum_{m} \sum_{n} \tilde{\beta}_{bmn} \tilde{z}_{dmn}$
    - $\tilde{\lambda}_{b} \leftarrow \tilde{\lambda}_{b-1} + \tilde{\lambda}_{b} + \tilde{f}_{b}(\eta)$
    - $\beta_{b} \propto \tilde{\lambda}_{b}$
- **end for**

---

**Algorithm 7 SVB-LDA**

- **Input:** Prior $\eta$, hyper-parameter $\alpha$, sequence of minibatches $C_1, C_2, \ldots$
- **Output:** $\lambda$
- **Initialize:** $\lambda_0 \leftarrow \eta$
- **for** each minibatch $C$ in $C_1, C_2, \ldots$ **do**
  - **Inference:**
    - for each document $d$ in $C$ **do**
      - Infer local variables $\phi, \gamma$
    - **end for**
  - **Update:**
    - $\tilde{\lambda}_{b} \leftarrow \sum_{d \in C} \phi_{d} \phi_{d}$
    - $\tilde{\lambda}_{b} \leftarrow \lambda_{b-1} + \tilde{\lambda}_{b}$
    - $\beta_{b} \propto \tilde{\lambda}_{b}$
- **end for**

---

**Algorithm 8 BPS-LDA**

- **Input:** Prior $\eta$, hyper-parameter $\alpha$, sequence of minibatches $C_1, C_2, \ldots$
- **Output:** $\lambda$
- **Initialize:** $\lambda_0 \leftarrow \eta$
- **for** each minibatch $C$ in $C_1, C_2, \ldots$ **do**
  - **Inference:**
    - for each document $d$ in $C$ **do**
      - Infer local variables $\phi, \gamma$
    - **end for**
  - **Update:**
    - $\tilde{\lambda}_{b} \leftarrow \sum_{d \in C} \phi_{d} \phi_{d}$
    - $\tilde{\lambda}_{b} \leftarrow \lambda_{b-1} + \tilde{\lambda}_{b} + \tilde{f}_{b}(\eta)$
    - $\beta_{b} \propto \tilde{\lambda}_{b}$
- **end for**

---

\(^5\) We found the same conclusion as investigating with different values of $T$. Hence for ease of discussion, we just report the results with $T = 50$. 

---

**Results:**

The effect of prior: Fig. 10 shows the results of BPS with different scale ratios, in comparison with SVB. Higher value of the scale ratio means we are using more prior knowledge.

It is clear that the results of BPS are better than those of SVB even with different values for scale ratio. In addition, there are rises in the accuracy of BPS when the scale ratio increases. It means that the prior is meaningful, so that emphasizing it will enhance the quality of the learning process. Moreover, Fig. 10b and Fig. 10d give illustrations that ignoring the external prior in SVB may lead to a considerable decrease in accuracy, while by maintaining the prior, BPS improves the accuracy of the model. Those subfigures also suggest that by boosting the good prior, BPS can perform better by a large margin than SVB.

Fig. 11 summarizes the results when changing the batch-size. We observe that BPS works well with different settings of batch-size. In all 4 datasets, the accuracy of BPS with different batch-sizes is higher than that of SVB. The performance of BPS in Yelp and Music is particularly good, and significantly better than that of SVB. The improvement with a large margin in Music suggests that the prior is really meaningful for sentiment analysis. Losing such a good prior knowledge causes to quickly reduce the discriminative ability of SVB.
5.3. Case study 3: streaming learning for LDA

In this case study, we evaluate BPS’s performance with the LDA model presented in Section 3.2.

**Streaming learning algorithm:** We present more detailed versions of the learning algorithms for LDA than in Section 3.2. The posterior of the latent variables given a corpus $C$ is: $p(\theta, \beta, z| C, x, \eta)$. We inherit the learning algorithm for LDA in Section 2.3 with the inference procedure for the local variables as in Algorithm 1. After applying BPS and SVB framework, we have two streaming learning algorithms for LDA in Algorithm 7 and 8. Note that Algorithm 7 is the same as the SSU algorithm in [1]. The only difference between the two algorithms is $f_s[\eta]$.

**Prior knowledge in use:** Relating to the distribution of word in natural language, Zipf’s law [27,28] gives us an interesting property that the frequencies of words in a specific language followed a power-law distribution of the form: $p(w) \propto r_w^{-\alpha}$, in which $p(w)$ is the proportion of word $w$ in the language, $r_w$ is the rank of the word in the descending sorted frequencies. This means that the most frequent word has rank $r = 1$. Parameter $l$ depends on the specific language. We use Zipf’s law as the prior knowledge in BPS as follows:

\[
\eta_w \propto r_w^{-l}.
\]

**Evaluation metric:** We use log predictive probability [1] to evaluate the predictive capacity of the learned models. A testing corpus $C_t$ is taken out from data. Each document of this test corpus is split into 2 parts: $d_{test}$ and $d_{dev}$ (with a ratio of 1 : 4 in this experiment). Suppose that the global parameters are given, the log predictive probability (LPP) is defined by:

\[
LPP = \sum_{d \in C_t} \log p(d_{test}|C_{dev}) = \sum_{d \in C_t} \sum_{w \in d_{test}} \log \sum_{k=1}^{K} E_k(\alpha_k) E_k(\beta_k).
\]

**Experimental setups:** We use 3 datasets for evaluation: New York Times from the UCI Machine Learning Repository, Yahoo and Twitter from [10]. Some statistics are presented in Table 4. Note that Yahoo and Twitter are two corpora of extremely short text, while New York Times is long text. The number of topics is set equally with $K = 100$, and the hyperparameter $x = 0.01$. The prior $\eta$ is taken from Eq. 45 with a heuristic parameter $l = 1.07$ [41] and the ranking of word’s frequencies is downloaded from top 100,000 most frequently-used English words text, only the words appeared in vocabulary of dataset are used.

For testing the effect of the scale ratio, we fix the batch-size $s = 5000$ for large corpora and 1000 for New York Times, and change $s \in \{0.02, 0.05, 0.08, 0.1, 0.2\}$. With batch-size testing, we fix $s = 0.02$ and change batch-size in $\{500, 2000, 4000\}$ for New York Times. With Yahoo and Tweet we fix the scale ratio to $s = 0.08$ and change the batch-size in $\{5000, 10000, 20000\}$. We choose greater $s$ and batch-size for short text corpora, because in those cases more information/knowledge should be needed.

The effect of prior: The results in this case study have a slightly different pattern with that in case study 2, and can be divided into two groups: long text (New York Times), and short text (Yahoo and Twitter). In general, a higher weight of the prior leads to worse performance of BPS in long text, however gives significant improvement for short text (Fig. 12).

In the experiments for testing boosting scale with the long text group, the smallest value of scale ratio for $BPS$ in the experiments $s = 0.02$ gives the highest quality of the model and better than in SVB. However, when increasing the boosting scale, the predictive capacity of LDA using BPS may be lower than using SVB. In contrast, the performance of BPS for the short text increases as increasing the boosting scale and gets the highest quality with $s = 0.2$. The better performance of BPS over SVB is very clear and often with a large margin. The same behavior of $BPS$ is reported in Fig. 13 when changing the batch-size and fixing the scale ratio with a suitable value (0.02 for short text, and 0.08 for long text).

We can explain such a behavior of BPS as follows. The short text often contains little information, and poses severe challenges for SVB. Further even a large number of short text could not overcome those challenges [11]. Therefore, exploiting external knowledge would be necessary in order to learn a good predictive model from short text. $BPS$ seems to exploit considerably well the knowledge from Zipf’s law, represented in (45), to surpass SVB with a large margin in Yahoo and Twitter. On the other hand, the long text itself contains much information and hence helps SVB perform well. Oversusing prior knowledge with a unsuitable scale factor will lead to overwhelm the information from data. As a result, the role of prior is not very significant in this case.

5.4. Case study 4: streaming text classification

To solve text classification problem, multi-view topic model [MviewLDA] [23] was introduced. The idea of MviewLDA is that each document of $D$ documents belongs to one of $J$ classes with probability contributed by a multinomial distribution $\chi \sim \text{Mult}(\pi)$. Each class $j$ contains $K$ local topics $\{\phi_k^j\}_{k=1}^{K}$, with the distribution over topics $\theta^j$ which are sampled from the Dirichlet distribution $\text{Dir}(\chi^j)$. Besides, they assume that there exists $R$ global topics $\{\phi_k^g\}_{k=1}^{R}$ that are shared by all classes. The binary variable $\delta$ decides when a word belongs to global or local topics. The graphical representation of MviewLDA is presented in Fig. 14.

The generative process of MviewLDA is as follows:

- **Generate topic distribution:**
  - For local topics: $\theta^j \sim \text{Dirichlet}(\eta^j)$
  - For global topics: $\theta^g \sim \text{Dirichlet}(\eta^g)$

- **Generate a document $d$ of length $N_d$:**
  - Draw a class: $\chi \sim \text{Mult}(\pi)$
  - Draw local topic proportion: $\delta\theta^j \sim \text{Dir}(\chi^j)$
  - Draw global topic proportion: $\delta\theta^g \sim \text{Dir}(\chi^g)$
  - Draw Bernoulli parameter $\epsilon_0 \sim \text{Beta}(\gamma)$
For each word \( w \) of document \( d \):
- Draw a binary indicator \( \delta \sim \text{Bernoulli}(\omega) \)
- If \( \delta = 1 \), word \( w \) belongs to local topic:
  - Draw a local topic \( z_i^d \sim \text{Mul}(\theta_i^d) \)
  - Draw word \( w \sim \text{Mul}(\beta_{jw}) \)
- If \( \delta = 0 \), word \( w \) belongs to global topic:
  - Draw a global topic: \( z_i^d \sim \text{Mul}(\phi^d) \)
  - Draw word \( w \sim \text{Mul}(\beta_{jw}) \)

Let each local topic being contributed by a Dirichlet prior \( \eta_i^d \). Note that local topics within each class contain feature words of its class. Therefore, if we set the weighted value of the feature words larger than the others and use it as a prior, we will provide more information into the model to differentiate classes. Such a technique can increase the quality of learning process.

**Algorithm 9 SVB-MviewLDA**

**Input:** Prior \( \eta \), sequence of minibatches \( C_1, C_2, \ldots \)

**Output:** \( \lambda \)

Initialize \( \lambda_0 \leftarrow \eta \)

for each minibatch \( C \) in \( C_1, C_2, \ldots \) do

**Inference:**

for each document \( d \) in \( C \) do

Infer local variables \((\zeta, \phi, \tau)\)

end for

**Update:**

\[
\begin{align*}
\lambda_i^d & \leftarrow \lambda_{i-1} + \lambda_i^d \\
\beta_i & \propto \lambda_i
\end{align*}
\]

end for

**Algorithm 10 BPS-MviewLDA**

**Input:** Prior \( \eta \), sequence of minibatches \( C_1, C_2, \ldots \)

**Output:** \( \lambda \)

Initialize \( \lambda_0 \leftarrow \eta \)

for each minibatch \( C \) in \( C_1, C_2, \ldots \) do

**Inference:**

for each document \( d \) in \( C \) do

Infer local variables \((\zeta, \phi, \tau)\)

end for

**Update:**

\[
\begin{align*}
\lambda_i^d & \leftarrow \lambda_{i-1} + \lambda_i^d \\
\beta_i & \propto \lambda_i
\end{align*}
\]

end for

**Learning algorithm:** Given corpus \( C = \{C_i\} \), in case of supervised learning with known class label \( y \), the posterior of interest is:

\[ p(\omega, \theta, \theta^d, \delta, z, \beta^d | y, C, \eta, \alpha, \psi, \gamma, \tau). \]  

The learning algorithm in [23] is Gibbs sampling and designed for batch learning. In this case study, we use variational inference (as in [42]) and modify to streaming classification learning. The variational distribution is:

\[
\begin{align*}
q(\omega, \theta^d, \theta^e, \delta, z_i, z_i^d, \theta^e, \beta^d | y, C) = q(\omega)q(\theta^d | \mu^d)q(\theta^e | \mu^e)q(\delta | \tau)q(z_i | \phi_i)q(z_i^d | \phi_i)q(z_i^d | \phi_i) \]
\]

Fig. 10. Streaming ASUM results with different scale ratios. The x-axis shows the amount of data received in the streaming learning process.
where $\phi_l^{(i)}$ and $\phi_g^{(i)}$ are multinomial parameters, $\nu$ is Beta parameter, $\tau$ is Bernoulli parameter, $\phi_l^{(i)}, \phi_g^{(i)}, \chi_l^{(i)}, \chi_g^{(i)}$ are Dirichlet parameters. We denote the indicator $I_{jd}^{(i)} = 1$ when document $d$ belongs to class $j$ and $I_{jd}^{(i)} = 0$ otherwise. Similarly, $I_{vd}^{(i)} = 1$ if the $i$th word of document $d$ is $v$. The detail of the inference procedure are shown in Algorithm 11 of the Appendix.

Because we aim to keep the information from prior $\gamma_l$ of the local topics, we only apply BPS to $\beta_l^{(i)}$. Adopting SVB and BPS with the variational inference leads to 2 streaming versions for updating $\lambda_l^{(i)}$ in Algorithm 9 and Algorithm 10.

**Table 4**

Datasets for evaluation of LDA. $l_d$ denotes the average number of words per document.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Vocabulary size</th>
<th>Training size</th>
<th>Testing size</th>
<th>$l_d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York Times</td>
<td>102,660</td>
<td>200,000</td>
<td>10,000</td>
<td>228.8</td>
</tr>
<tr>
<td>Yahoo</td>
<td>24,419</td>
<td>500,000</td>
<td>10,000</td>
<td>4.7</td>
</tr>
<tr>
<td>Twitter</td>
<td>89,474</td>
<td>1,500,000</td>
<td>10,000</td>
<td>9.8</td>
</tr>
</tbody>
</table>

**Fig. 11.** Streaming ASUM results with different batch-sizes.

**Fig. 12.** Streaming LDA results with different scale ratios.
Prior knowledge in use: Borrowing the idea from [30], we extract the feature words of each class and use them as the prior knowledge. At first, we calculate TF.IDF for the words in each class then select top 5000 words with highest TF.IDF values as seed words. The seed words of class \( j \) are then used to initialize prior \( g_l(\cdot) \) by assigning a value \( v = 0.5 \). The other values are set to a small value /C15 = 0.01.

Evaluation metric: The classification accuracy is used in this evaluation.

Experimental setups: We use 2 labeled datasets (Cade 12 and News20) with some statistics described in Table 5.

Table 5: Dataset for text classification.
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Num of classes</th>
<th>Vocabulary size</th>
<th>Training size</th>
<th>Testing size</th>
</tr>
</thead>
<tbody>
<tr>
<td>News20</td>
<td>20</td>
<td>62,061</td>
<td>16,000</td>
<td>3,900</td>
</tr>
<tr>
<td>Cade12</td>
<td>12</td>
<td>193,997</td>
<td>27,322</td>
<td>4000</td>
</tr>
</tbody>
</table>

For News20, we set the number \( K = 10 \) of topics in each local class and the number \( R = 8 \) of global topics. For Cade 12, we use \( K = 15 \) and \( R = 4 \) because of being larger in size than News20. The other Dirichlet prior parameters are set equal to 0.01 as case study 3 on LDA. We take SVM into account as another baseline to evaluate the quality of classification.

For testing scale ratio, we set batchsize as 1000 and change scale factor \( s \in \{0.2, 0.4, 0.6, 0.8, 1.0\} \). For testing batchsize, we fix scale factor \( s = 0.4 \) and change batchsize in \( \{1000, 1500, 2000\} \).

Results: The results are summarized in Fig. 15 and Fig. 16. The results of BPS are comparable with those of SVB. However, we can see that the accuracy of BPS tend to be higher than SVB in the latter minibatches, and significantly better in News20 dataset. Again, this pattern reflects that maintaining the
prior knowledge is valuable in streaming learning. SVB will lose prior information while BPS still keep it, therefore, BPS is able to produce a better result than SVB.

From these experiments, we can see that the appropriate values of the scale ratio depend on the quality of the prior knowledge provided into models. Doing a pre-search to find a good range value of scale ratio will help to improve the result of the learned models. The result of BPS is better in comparison with SVB in most cases with the same batchsize settings.

6. Conclusion

In this paper, we discussed the problem of incorporating external knowledge as the prior for streaming Bayesian learning. We showed that SVB easily forgets prior knowledge as more data come. This may be problematic when we want to exploit valuable knowledge to deal with the challenges of sparsity and noise. We then proposed BPS to boost the role of the prior knowledge. Within BPS, the valuable information from prior is maintained through learning process, and hence BPS easily overcomes the vanishing prior problem. BPS provides a simple but effective solution for practice, and its idea can be easily employed in other streaming methods.

Appendix A

This section presents the details of the inference of local variables in ASUM (Algorithm 12) and MviewLDA (Algorithm 11).

Algorithm 11 Inference of local variables in ASUM

**Input:** Prior $\eta, \pi = 1$, global variable $\beta$, document $d = \{W_1, \ldots, W_m\}$ with $M$ sentences

**Output:** Local variables: $\theta, \pi, c, z_d$

Randomly initialize $\theta_d, \pi_d$. 

repeat

Infer $\theta_d$ by using Frank-Wolfe algorithm [40] to maximize

$$f(\theta_d) = \sum_{m=1}^{M} \log \sum_{c=1}^{N} (\pi_c \theta_{dc}) \prod_{n=1}^{N} \beta_{nmw}$$

Infer $\pi_d$ by using Frank-Wolfe algorithm to maximize

$$f(\pi_d) = \sum_{m=1}^{M} \log \sum_{c=1}^{N} (\pi_c \theta_{dc}) \prod_{n=1}^{N} \beta_{nmw}$$

until convergence

Estimate $\hat{e}_d, \hat{z}_d$ for each sentence $m$:

$$\hat{e}_m = \hat{p}(e_m | e_i, \theta_d, W_m, \phi) \approx \sum_{c=1}^{N} \pi_c \theta_{dc} \prod_{n=1}^{N} \beta_{nmw}$$

$$\hat{z}_m = \hat{f}(z_m | c_i, \theta_d, W_m, \phi) \approx \sum_{c=1}^{N} \pi_c \theta_{dc} \prod_{n=1}^{N} \beta_{nmw}$$

Algorithm 12 Inference of local variables in MviewLDA

**Input:** Global variable $\eta^{(1)}, \eta^{(2)}$, document $d$ and label $\lambda_d$

**Output:** $v, \mu, \tau, \phi$

Randomly initialize $v, \mu, \tau, \phi$.

repeat

$$v_1 = \bar{y}_1 + \sum_{i=1}^{n} \tau_i, v_2 = \bar{y}_2 + \sum_{i=1}^{n} (1 - \tau_i)$$

$$\mu_i^{(1)} = \mu_i^{(2)} + \sum_{j=1}^{K} \sum_{k=1}^{N} \phi_{ijk}^{(1)} \phi_{ijk}^{(2)} + 1 - \mu_i^{(2)}, \mu_i^{(2)} = \frac{\bar{y}_1}{\sum_{i=1}^{n} (1 - \tau_i)}$$

$$\tau_i = \{1 + \exp(-\Psi(y_{1i}) + \Psi(y_{2i})) - \sum_{j=1}^{K} \sum_{k=1}^{N} \phi_{ijk}^{(1)} \phi_{ijk}^{(2)} - \Psi(\bar{y}_2)\}^{-1}$$

$$\phi_{ijk}^{(1)} \propto \exp(\mu_i^{(1)} - \Psi(\bar{y}_2) - \Psi(\bar{y}_1) + \sum_{k=1}^{N} \phi_{ijk}^{(2)} - \sum_{k=1}^{N} \phi_{ijk}^{(2)} \cdot \Psi(\bar{y}_2))$$

$$\phi_{ijk}^{(2)} \propto \exp((1 - \tau_i)(\mu_i^{(2)} - \Psi(\bar{y}_2) + \sum_{k=1}^{N} \phi_{ijk}^{(2)} - \sum_{k=1}^{N} \phi_{ijk}^{(2)} \cdot \Psi(\bar{y}_2)))$$

until convergence

CRediT authorship contribution statement

Duc Anh Nguyen: Conceptualization, Methodology, Software Validation, Formal analysis, Writing - original draft, Visualization, Investigation. Van Linh Ngo: Conceptualization, Methodology, Validation, Formal analysis, Writing - review & editing, Investigation.

Kim Anh Nguyen: Conceptualization, Methodology, Validation, Formal analysis, Writing - review & editing, Supervision. Canh Hao Nguyen: Formal analysis, Writing - review & editing, Supervision. Khoat Than: Conceptualization, Methodology, Software, Validation, Formal analysis, Writing - review & editing, Supervision, Project administration, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.
References


Canh Hao Nguyen received his B.S. degree in Computer Science from the University of New South Wales, Australia, M.S. and Ph.D. degrees from JAIST, Japan. He has been working in machine learning and bioinformatics. His current interests are machine learning for graph data, sparse modeling with applications in biological network analysis.

Khoat Than is currently an associate professor at Hanoi University of Science and Technology. He received Ph.D. degree from Japan Advanced Institute of Science and Technology in 2013. He joins the Program Committees of various leading international conferences, including ICML, NIPS, IJCAI, ICLR, PAKDD, ACML. His recent research interests include representation learning, deep generative models, topic modeling, continual learning.