Slow-learner Isn't My Problem: Exploiting Structure in high-D, high-V Latent Space Stochastic Learning

Abhimanu Kumar **Alex Beutel** **Qirong Ho** **Eric P. Xing**
School of Computer Science, Carnegie Mellon University
Pittsburgh PA 15213,USA,
{abhimank,abeutel, qho, epxing}@cs.cmu.edu

Abstract

We pesent here a scheme for exploiting distributable structure present in latent space model for high-dimensional (high-D) and high-volume (high-V) learning. Models like latent dirichlet allocation and mixed membership stochastic blockmodels have structures that can be exploted for big learning. We present a distributed learning strategy for such models. The scheme is distributed in data as well as parameter space and avoids waiting for slow learners to obtain full distributive leverage. The learning scheme, flexible to slow worker-processors, has provably correct strategy for convergence even with fewer synchronizations. It provides a tunable synchronization startegy that can be set based on the learning needs of the end user. We provide theoretical bounds on the parameter variance among workers with different synchronization strategies. Empirical results presented for latent space models such as latent dirichlet allocation and mixed membership model show that it scales very well with large data as well as parameter space. The comaparative evaluation with other parallel and distributed learning startegies shows better preformance of the model.

1 Introduction

In todays information age there is trillions of bytes of data being generated every moment. By one estimate we generate 5 exabytes of data on the internet every two days [1] and most of it is user generated content. Given this massive amount of data generated every moment large scale machine learning is not just of academic interest anymore but of practical significance too. Large scale learning or big learning has been an active area of research in recent times. But most of this research has been focused around big data and the aspect of distibuting the learnig model has take back seat. While dealing with big data is definitely a must in this massive content generation age the learning task might turn out to be non-trivial when big data meets complex models. Learning models with high dimension or number of parameters becomes non-trivial even with slight increase in data. Hence there is a need for learning scheme which can perform distributed data as well as parameter learning. This problem is highly prevalent in latent space model such as latent dirichlet allocation (LDA [2]), mixed membership stochastic blockmodels (MMSB [1]) as they inflate their parameter space by introducing large number of latent variables. Henceforth these models would be our object of focus in this paper. We develop a stochastic learning scheme for latent space model, specifically for LDA and MMSB, which is distributed in data as well as parameter space. We modify the originial objective to suit our optimization scheme.need to write this in a better way . The structure obtained can be further exploited to minimize the hazardous effects of slow-processors in the distributed system.

expand some more giving an over view of our learning scheme

[1] http://techcrunch.com/2010/08/04/schmidt-data/

2 Related Work

PSGD (only data parallel); Aggarawal and Duci's Distributed Delayed (Again Data partition only; and it's hard to code); Any Other? How do we discuss the original Haas's paper though we do considerable more in terms of theory and multi-iteration

3 Slow-Learner Agnostic Distributed Learning Framework

Our approach is to exploit independent cliques of structure present in large scale machine learning models. Probabilistic graphical models introduce massive amount of latent variables to induce the modelers belief regarding the generative process of the data. Though this enables these models with a unique ability to provide an interpretation and a generative story, it also makes the model harder to learn since the parameter as well as variable space increases massively. These models when run on large data have their learning problem becomes twice difficult.

While it may appear that latent space models' biggest boon of incorporating hidden variables is their biggest bane for large scale learning, it turns out that is not the case. On close examination it appears that these latent variables have another adavantage: they have a structure. They are generally found in cliques of correlated variable sets. This independence structure can be exploited effectively for a distributed learning framework. Moreover in models such as LDA and MMSB with a modified objective need to put it in a better way one can achieve ditributivity in data as well as parameters. We explain this using a basic LDA model. In simple terms, the aim of an LDA model given a *document by vocabulary* matrix (Y) is to split it into two matrices: a *documents by topics* (π) and a *topics by vocabulary* matrix (β). Equation 1 presents this in an optimization framework with simplex and non-negativity constraints. The is an ℓ_p norm which is typecally ℓ_2.

$$\text{argmin}_{\pi,\beta} L = ||Y - \pi\beta||_p^p = \sum_{i,j}(Y_{i,j} - \sum_k \pi_{i,k}\beta_{j,k})_p^p \tag{1}$$
$$s.t. \sum_k \pi_{i,k} = 1, \sum_k \beta_{j,k} = 1, \pi_{i,k} \geq 0, \beta_{j,k} \geq 0 \ \ \forall\, i, j$$

For MMSB a similar objective can be formulated. Given a *user by user* interaction strength is it the right term matrix Y it aims to find two matrices π and B, which are *user by topic* and *topic by topic* matrix respectively, such that $\pi^T B \pi$ reconstructs the original matrix interaction matrix Y. Equation 2 presents this in an optimization framework with the usual non-negativity and simplex constraints.

$$\text{argmin}_{\pi,B} L = ||Y - \pi^T B\pi||_p^p = \sum_{i,j}(Y_{i,j} - \sum_{g,h} \pi_{i,g} B_{g,h} \pi_{j,h})_p^p \tag{2}$$
$$s.t. \sum_k \pi_{i,k} = 1, \pi_{i,k} \geq 0, B_{i,k} \geq 0 \ \ \forall\, i, j$$

Do we present the symmetric matrix factorization objective too; since it's is nice scheduling scheme as well as ℓ_p and ℓ_2 combined loss

For the LDA objective, figure 3 shows the way parameters are divided in the distributive scheme. We perform parameter updates of different blocks parallely. For SGD we perform updates to our parameters π, β, which we will collectively refer to as $\mathbf{\Theta}$ matrix whereas θ are the individual components of the matrix. This definition of $\mathbf{\Theta}$ and θ will come in handy for parameter updates based on the gradient at individual data points. E.g. the update for the LDA objective L are (note: projections are involved here):

$$\theta^{(t+1)} = \theta^{(t)} - \eta_t \nabla \mathcal{L}_{Y_{i,j}}(\theta^{(t)}) \tag{3}$$

For these update rules, we list below the differentials for each component σ of θ where norm use is ℓ_2, and $(\nabla L_{Y_{i,j}}(\theta))_\sigma = \frac{\partial L_{Y_{i,j}}}{\partial \sigma}$:

$$(\nabla L_{Y_{i,j}}(\theta))_\sigma = \begin{cases} -2(Y_{i,j} - \sum_r \pi_{i,r}\beta_{j,r})\beta_{j,\ell} & \text{if } \sigma = \pi_{i,\ell} \\ 0 & \text{if } \sigma = \pi_{i',\ell},\ i \neq i' \end{cases} \tag{4}$$

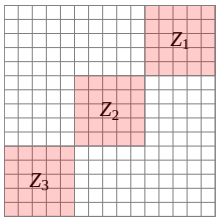

Figure 1: Dividing the *document by vocabulary* matrix fo LDA into blocks such that no two of them share any row or a column.

similarly for $\sigma = \beta_{j,l}$. From this we observe that SGD update for $\pi_{i,l}$ at a particular entry $Y_{i,j}$ depends only on previous $\pi_{i,r}, \beta_{j,r}$ where $r \in 1, \ldots, R$ and R is the number of topics we chose. The updates for each component are similar for the MMSB case. modify here for MMSB

Input : Y, β, π, sub-epoch size d
$\pi \leftarrow \pi_0$, $\beta \leftarrow \beta_0$
Block Y, π, β into corresponding w blocks
while *not converged* **do**
  Pick step size η
  Pick w blocks($Z_1^{(s)}, ..., Z_w^{(s)}$) to form stratum $Z^{(s)}$
  for $b = 0, \ldots, w-1$ ***in parallel*** **do**
    Run SGD on the training points $Z_b^{(s)}$
    /** Do unitl every block is ready to synchronize **/
    /** So potentially each block $Z_b^{(s)}$ runs through each data-point $n_{Z_b^{(s)}}$ times **/
  end
end

Algorithm 1: Algorithm for LDA updates. The w blocks correspond to w worker processors

Given this understanding of our optimization objective and SGD update rules, we would like to segment our data in such a way that certain blocks Z_b can be run in parallel, where we define $Z_b \subseteq Y$. Figure 3 sjows the way we segment our simple matrix Y to enable parallelization. In order to run SGD on our blocks in parallel, we divide them such that no two blocks share common rows or columns. To be more precise, we say that a point $y \in Z_b$ is the coordinates in the data, such as $y = (y_i, y_j y) \in Y$. Two blocks Z_b and $Z_{b'}$ are non-overlapping if for all $x \in Z_b$ and $y' \in Z_{b'}$, $y_i \neq y_i'$ and $y_j \neq y_j'$. In order to cover all regions of Y, we need multiple strata. We require w strata In our algorithm, we run the strata sequentially, but for each stratum we run SGD on the blocks in parallel. Different blocks in a starta can make different number of passes trhough the data. This is where the algorithm is robust against slow processors as the worker keep running instead of waiting until every body is ready to synchronize. We consider running SGD on one stratum to be a subepoch in our algorithm, and running it on all strata an epoch. (Note, the order in which you run the strata does not matter, as long as they are each run once per epoch.) Algorithm 1 explain the steps more formally. We next offer a proof that this multi-iteration per block distributed parameter learning converges appropriately.

Our Learning framework presented here. We present it in a way that emphasizes on latent space models
Salient points:
1) different processors can make different number of iterations over the data without synchronization.
2) The workers donot wait for slower workers and instead keep working and synchronize when everybody is ready. We should add that one can choose to not synchronize often based on how much they want variance among different processors and how slow is their slowest processor.
3) We present theoretical guaranties for convergence; **for simple as well as projected loss**
4) We provide bounds on variance among different workers at epoch and sub-epoch levels.
5) We demonstrate theoretically that keep doing work in place of waiting gets faster convergence. **TODO**
6) We show that our results are for generic loss fucntion and works even with projections/constraints($\ell 1$, Non-negativity, Simplex etc.)
7) **Our assumptions: The structure; Martingale Diference Error ε

4 Theoretical Analysis

1) Convergence (with projection)
2) Epoch variance bound
3) Sub-epoch variance bound
4) Show that waiting is not a good idea and workers should keep working till every one is ready

Here we analyse the method presented in algorithm 1 theoretically. We will prove that the multi-iteration per block strategy converges. We provide a bound on the variance accross two blocks with in a sub-epoch running parallely. We show theoretically how the variance varies after each synchronization between two consecutive epochs. In the end we show why working instead of wating for the slowest processor is a better strategy.

4.1 Converegnce proof

Suppose we have $V^{(t)} = V^{(t)}(\theta^{(t+1)}, \theta^{(t)})$ is a fucntion that keeps track of the state of $\theta^{(t)}$ and $\theta^{(t+1)}$ and depends upon the data-point ${Y_{i,j}}^{(t)}$ picked in the SGD update. From equation 3 we have

$$\begin{aligned}\theta^{(t+1)} &= \theta^{(t)} - \eta_t \delta L^{(t)}(V^{(t)}, \theta^{(t)}) \\ &= \theta^{(t)} - \eta_t \nabla\mathcal{L}(\theta^{(t)}) + \eta_t \left[\nabla\mathcal{L}(\theta^{(t)}) - \delta L^{(t)}(V^{(t)}, \theta^{(t)})\right] \\ &= \theta^{(t)} - \eta_t \nabla\mathcal{L}(\theta^{(t)}) + \eta_t \varepsilon_t \end{aligned} \tag{5}$$

Here $\nabla\mathcal{L}(\theta^{(t)})$ is the exact gradient at iteration t. And we call error at iteration t, $\left[\nabla\mathcal{L}(\theta^{(t)}) - \delta L^{(t)}(V^{(t)}, \theta^{(t)})\right]$, as ε_t. We make the assumption that the error $\varepsilon_t = \left[\nabla\mathcal{L}(\theta^{(t)}) - \delta L^{(t)}(V^{(t)}, \theta^{(t)})\right]$ in each step is a martingale difference sequence. i.e we assume that

$$\begin{aligned} E\left[\nabla\mathcal{L}(\theta^{(t)}) - \delta L^{(t)}(V^{(t)}, \theta^{(t)}) | \delta L^{(i)}(V^{(i)}, \theta^{(i)}), \theta^{(i)}, i < t, \theta^{(t)}\right] &= 0 \\ E\left[\varepsilon_t | \varepsilon_i, i < t\right] &= 0 \end{aligned} \tag{6}$$

Move the following justification of martingale sequence some where more relevant We have to note here that assuming error terms are a martingale difference sequence is a weaker assumption than assuming error terms are independent of each other. Making the martingale difference assumption means that $\delta L^{(t)}(V^{(t)}, \theta^{(t)})$ conditioned on $\theta^{(0)}$ and $\delta L^{(i)}(V^{(i)}, \theta^{(i)}), \theta^{(i)}$ given $\theta^{(t)}$ is independent of everything else. This is achieved since in the blocking strategy explained earlier has no overlap between two parameters that are updated parallely.

We have w worker processors (algorithm 1) and assume that every worker i touches each data point n_i times in its block before syncing. Hence

$$\begin{aligned} \theta^{(t+(\sum_1^w n_i)m)} = \theta^{(t)} + \sum_{i=t}^{t+m(\sum_1^w n_i)} -\eta_i \nabla\mathcal{L}(\theta^{(i)}) + \sum_{i=t}^{t+m(\sum_1^w n_i)} \eta_i \varepsilon_i \\ \implies \theta^{(t+mN_w)} = \theta^{(t)} + \sum_{i=t}^{t+mN_w} -\eta_i \nabla\mathcal{L}(\theta^{(i)}) + \sum_{i=t}^{t+mN_w} \eta_i \varepsilon_i \\ \text{assuming } \sum_1^w n_i = N_w \\ \implies \theta^{(t+mN_w)} = \theta^{(t)} + \sum_{i=t}^{t+mN_w} -\eta_i \nabla\mathcal{L}(\theta^{(i)}) + M_{mN_w} \end{aligned} \tag{7}$$

where $M_{mN_w} = \sum_{i=t}^{t+mN_w} \eta_i \varepsilon_i$ is a martingale sequence since it is a sum of martingale difference sequence. Besides mN_w captures the m whole sub-epochs of work done as a whole by all the

workers combined. From Doobs martingale inequality ([3], ch. 1, Thm 3.8)

$$P\left(\sup_{t+mN_w \geq r \geq t} |M_r| \geq c\right) \leq \frac{E\left[\left(\sum_{i=t}^{t+mN_w} \eta_i \varepsilon_i\right)^2\right]}{c^2} \tag{8}$$

where $M_r = \sum_{i=t}^{r} \eta_i \varepsilon_i$. Lets look at the RHS of equation 8 above:

$$E\left[\left(\sum_{i=t}^{t+mN_w} \eta_i \varepsilon_i\right)^2\right] = E\left[\sum_{i=1}^{mN_w} (\eta_i \varepsilon_i)^2\right]$$

(equation 6 $\implies E[\varepsilon_i \varepsilon_j] = 0$ if $i \neq j$)

$$= \sum_{i=1}^{mN_w} \eta_i^2 E[\varepsilon_i^2] \leq \sum_{i=1}^{mN_w} \eta_i^2 D \to 0$$

where $E[\varepsilon_i^2] < D \; \forall i$ and assuming $\sum \eta_i^2 < \infty$

$$\lim_{t \to \infty} \implies P\left(\sup_{i \geq t} |M_i| \geq c\right) = 0 \;\; as \;\; t \to \infty \tag{9}$$

From equation 9 we have

$$\theta^{(t+mN_w)} = \theta^{(t)} + \sum_{i=t}^{t+mN_w} -\eta_i \nabla \mathcal{L}(\theta^{(i)}) \tag{10}$$

asymptotically. Hence the algorithm converges to the same set of limit poits as the ecxact gradient descent asymptotically.

4.2 Variance in a Sub-epoch

Here $\theta^{(t)}$ is the value of parameter theta at iteration t and $V^{(t)}(\theta^{(t+!)}, \theta^{(t)})$ is a state function that is dependent upon the past $\theta^{(t)}$, future $\theta^{(t+1)}$ and the data points $x^{(t)}$ picked at iteration. η_t is the step size at iteration t and $L^{(t)}$ is the loss at point $x^{(t)}$ in iter t. We assume for simplicity that the parameter being updated in block-i is univariate. This analysis can be easily extended to a multivariate parameter updation case in each block of a sub-epoch.

$$\theta^{(t+1)} = \theta^{(t)} - \delta\theta^{(t)}(V^{(t)}, \theta^{(t)}) \tag{11}$$

where $\delta\theta^{(t)}(V^{(t)}, \theta^{(t)}) = \eta^{(t)} \delta L^{(t)}(V^{(t)}, \theta^{(t)}) \Rightarrow \theta^{(t+1)} = \theta^t - \eta_t \delta L^t(v^t, \theta^t)$

Summing equation 11 over n_i, the number of points in block i of a sub-epoch

$$\theta^{t+n_i} = \theta^t - \sum_{i=1}^{n_i} \eta_{t+i} \delta L^{t+i}(v^{t+i}, \theta^{t+i}) \tag{12}$$

$$\begin{aligned} p(\theta^{(t+1)}|\theta^t) d\theta^{(t+1)} &= p(u(\theta^{(t+ni)}, \theta^t)) du \;\; p(\theta^{(t+ni)}) d\theta^{(t+ni)} \\ &= \int_{\theta^t} p(\theta^{(t+ni)}|\theta^t) p(\theta^t) d\theta^t d\theta^{(t+ni)} = \int_{\theta^t} p(V(\theta^{(t+ni)}, \theta^t)) dV d\theta^t \end{aligned} \tag{13}$$

$$\begin{aligned} \mathbb{E}^{\theta^{(t+n_i)}}[u(\theta^{(t+n_i)})] &= \int_{\theta^{(t+n_i)}} u(\theta^{(t+n_i)}) p(\theta^{(t+n_i)}) d\theta^{(t+n_i)} \\ &= \int_{\theta^{t+i}} u(\theta^{(t+n_i)}) P(\theta^{(t+n_i)}) d\theta^{(t+n_i)} \\ &= \int_V \int_{\theta^t} u(\theta^{(t+n_i)}) P(V(\theta^{(t+n_i)}, \theta)) dV P(\theta^t) d\theta^t \\ &= \mathbb{E}^{\theta^t}[\mathbb{E}^V[u(\theta^{(t+n_i)})]] \end{aligned} \tag{14}$$

$$\mathbb{E}^V[u(\theta^{(t+ni)})] = \mathbb{E}^V[u(\theta^t + (\underbrace{-\sum_{i=1}^{n_i}\eta_{t+i}\delta L^{t+i}(v^{t+i},\theta^{t+i}))}_{\nabla}))]$$

$$= \mathbb{E}^V[u(\theta^t) - \frac{du(\theta^t)}{d\theta^t}\nabla + \frac{1}{2}\frac{du^2(\theta^t)}{d(\theta^t)^2}\nabla^2 + \mathcal{O}(\eta_t^3)]$$

$$= u(\theta^t) - \eta_t\frac{du(\theta^t)}{d\theta^t}\mathbb{E}^V[\sum_{i=1}^{n_i}\delta L^{t+i}(v^{t+i},\theta^{t+i})] + \eta_t^2\frac{1}{2}\frac{du^2(\theta^t)}{d(\theta^t)^2}\mathbb{E}^V[(\sum_{i=1}^{n_i}\delta L^{t+i}(v^{t+i},\theta^{t+i}))^2]$$

$$+ \mathcal{O}(\eta_t^3) \qquad \left(\text{since } \eta_t = \eta_{t+i} \text{ within a block}\right)$$

$$= u(\theta^t) - \eta_t\frac{du(\theta^t)}{d\theta^t}\sum_{i=1}^{n_i}\frac{d\mathbb{E}^V[L^{t+i}(v^{t+i},\theta^{t+i})]}{d\theta^{t+i}} + \eta_t^2\frac{1}{2}\frac{du^2(\theta^t)}{d(\theta^t)^2}\mathbb{E}^V[(\sum_{i=1}^{n_i}\frac{dL^{t+i}(v^{t+i},\theta^{t+i})}{d\theta^{t+i}})^2]$$

$$+ \mathcal{O}(\eta_t^3)$$

$$= u(\theta^t) - \eta_t\frac{du(\theta^t)}{d\theta^t}(n_i\frac{d\mathbb{E}^{v^t}[L^t(v^t,\theta^t)]}{d\theta^t}) \qquad \left(\text{by expectation property}\right.$$

$\mathbb{E}^V[L^{t+i}(v^{t+i},\theta^{t+i})] = \mathbb{E}^{v^{t+i}}[L^{t+i}(v^{t+i},\theta^{t+i})] = \mathbb{E}^{v^t}[L^t(v^t,\theta^t)]$,
& assuming the condition that we make small and equal gradient step in each iteration
$(t+i)$ in the same direction i.e. $\left.\frac{d\mathbb{E}^{v^{t+i}}[L^{t+i}(v^{t+i},\theta^{t+i})]}{d\theta^{t+i}} = \frac{d\mathbb{E}^{v^{t+j}}[L^{t+j}(v^{t+j},\theta^{t+j})]}{d\theta^{t+j}}\right)$

$$+ \eta_t^2\frac{1}{2}\frac{du^2(\theta^t)}{d(\theta^t)^2}\left[\mathbb{E}^V[\sum_{i=1}^{n_i}(\frac{dL^{t+i}(v^{t+i},\theta^{t+i})}{d\theta^{t+i}})^2] + \mathbb{E}^V[\sum_{i\neq j}\frac{dL^{t+i}(v^{t+i},\theta^{t+i})}{d\theta^{t+i}}\frac{dL^{t+j}(v^{t+j},\theta^{t+i})}{d\theta^{t+j}}]\right]$$

$$+ \mathcal{O}(\eta_t^3)$$

$$= u(\theta^t) - \eta_t\frac{du(\theta^t)}{d\theta^t}(n_i\frac{d\mathbb{E}^{v^t}[L^t(v^t,\theta^t)]}{d\theta^t}) + \eta_t^2\frac{1}{2}\frac{du^2(\theta^t)}{d(\theta^t)^2}\left[\sum_{i=1}^{n_i}\mathbb{E}^{v^t}[(\frac{dL^{t+i}(v^{t+i},\theta^{t+i})}{d\theta^{t+i}})^2]\right.$$

$$\left. + n_i(n_i-1)(\frac{d\mathbb{E}^{v^t}[L^t(v^t,\theta^t)]}{d\theta^t})^2\right] + \mathcal{O}(\eta_t^3) \tag{15}$$

Last statement in equation 15 holds because two different data points picked at iteration $(t+i)$ and $(t+j)$ are independent of each other, by expectation property

$$\mathbb{E}^V[\frac{dL^{t+i}(v^{t+i},\theta^{t+i})}{d\theta^{t+i}}\frac{dL^{t+j}(v^{t+j},\theta^{t+i})}{d\theta^{t+j}}] = \mathbb{E}^{v^{t+i}}[\frac{dL^{t+i}(v^{t+i},\theta^{t+i})]}{d\theta^{t+i}}]\mathbb{E}^{v^{t+j}}[\frac{dL^{t+j}(v^{t+j},\theta^{t+i})}{d\theta^{t+j}}] \tag{16}$$

From equation 15 and 14

$$\mathbb{E}^{\theta^{(t+n_i)}}[u(\theta^{(t+n_i)})] = \mathbb{E}^{\theta^{(t)}}\left[u(\theta^t) - \eta_t\frac{du(\theta^t)}{d\theta^t}(n_i\frac{d\mathbb{E}^{v^t}[L^t(v^t,\theta^t)]}{d\theta^t})\right.$$
$$+ \eta_t^2\frac{1}{2}\frac{du^2(\theta^t)}{d(\theta^t)^2}[\sum_{i=1}^{n_i}\mathbb{E}^{v^t}[(\frac{dL^{t+i}(v^{t+i},\theta^{t+i})}{d\theta^{t+i}})^2]$$
$$\left. + n_i(n_i-1)(\frac{d\mathbb{E}^{v^t}[L^t(v^t,\theta^t)]}{d\theta^t})^2]\right] + \mathcal{O}(\eta_t^3) \tag{17}$$

From equation above the variance of θ^{t+n_i} is

$$
\begin{aligned}
&Var(\theta^{t+n_i}) = \mathbb{E}^{\theta^{(t+n_i)}}[(\theta^{(t+n_i)})^2] - \left(\mathbb{E}^{\theta^{(t+n_i)}}[\theta^{(t+n_i)}]\right)^2 \\
&= \mathbb{E}^{\theta^t}[(\theta^t)^2] - \eta_t n_i \mathbb{E}^{\theta^t}[2\theta^t(C_0(\theta^t - \theta_*) + \mathcal{O}(|\theta^t - \theta_*|^2))] \\
&+ \eta_t^2 \frac{1}{2}\mathbb{E}^{\theta^t}[2\{n_i^2(C_1 + C_2\epsilon_t + \mathcal{O}(\epsilon_t^2)) + n_i(n_i-1)(C_0(\theta^t - \theta_*) + \mathcal{O}(|\theta^t - \theta_*|^2))^2\}] \\
&- \left(\mathbb{E}^{\theta^t}[(\theta^t)^2] - \eta_t n_i \mathbb{E}^{\theta^t}[2\theta^t(C_0(\theta^t - \theta_*) + \mathcal{O}(|\theta^t - \theta_*|^2))]\right)^2 \\
&= \mathbb{E}^{\theta^t}[(\theta^t)^2] - 2C_0\eta_t n_i \mathbb{E}^{\theta^t}[(\theta^t)^2] + 2C_0\eta_t n_i \theta_* \mathbb{E}^{\theta^t}[\theta^t] - \mathcal{O}(\eta_t \epsilon_t^2) \\
&+ \eta_t^2 n_i^2 C_1 + n_i^2 C_2 \mathcal{O}(\eta_t^2 \epsilon_t) + \mathcal{O}(\eta_t^2 \epsilon_t^2) \\
&- \left(\mathbb{E}^{\theta^t}[\theta^t]\right)^2 + 2n_i\eta_t \mathbb{E}^{\theta^t}[\theta^t](\mathbb{E}^{\theta^t}[C_0\theta^t] - C_0\theta_* + \mathcal{O}(|\theta_t - \theta_*|^2)) - \mathcal{O}(\eta_t^2\epsilon_t^2) + \mathcal{O}(\eta_t^3) \\
&= Var(\theta^t) - 2\eta_t n_i C_0(Var(\theta^t)) + \eta_t^2 n_i^2 C_1 + \mathcal{O}(\eta_t^2\epsilon_t) + \mathcal{O}(\eta_t\epsilon_t^2) + \mathcal{O}(\eta_t^3) \qquad (18)
\end{aligned}
$$

Make clear that we are simplifying here and making the assumption that θ is uni-variate

4.3 Variance between two consecutive sub-epochs

From our algorithm defined

4.4 Why waiting is bad

5 Experiments

This is the part that we need to decide very early

5.1 Scalability experiments in data as well as parameter

Experiments to show how it scales with more processors (Do we have enough processors?) a) Scaling with parameters (number of topics), and b) Scaling with Data (Documents)

5.2 Speed comparison and less sycnhing experiments

Comparison with PSGD on the speed and accuracy of convergence (Do we compare on the original objective or the new objective)

5.3 Qualitative results on latent space models

We do both for MMSB and LDA both

Qirong and Alex think that we should include symmetric matrix factorization as well as it shows the scheduling strategy when thinsg are not cleanly seperated.

6 Results Analaysis and Discussion

7 Conclusion

References

[1] Edoardo M. Airoldi, David M. Blei, Stephen E. Fienberg, and Eric P. Xing. Mixed membership stochastic blockmodels. *J. Mach. Learn. Res.*, 9:1981–2014, June 2008.

[2] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[3] Avner Friedman. Stochastic differential equations and applications. In Jaures Cecconi, editor, *Stochastic Differential Equations*, volume 77 of *C.I.M.E. Summer Schools*, pages 75–148. Springer Berlin Heidelberg, 1975.