Manuscript under review by AISTATS 2021

Kolesov Aleksander , Kungurtsev Vyacheslav

11.11.2020

Decentralized Langevin Dynamics over a Directed Graph

Abstract

The prevalence of technologies in the space of the Internet of Things and use of multi-processing computing platforms to aid in the computation required to perform learning and inference from large volumes of data has necessitated the extensive study of algorithms on *decentralized* platforms. In these settings, computing nodes send and receive data across graph-structured communication links, and using a combination of local computation and consensus-seeking communication, cooperately solve a problem of interest. Recently, Langevin dynamics as a tool for high dimensional sampling and posterior Bayesian inference has been studied in the context of a decentralized operation. However, this work has been limited to undirected graphs, wherein all communication is two-sided, i.e., if node A can send data to node B, then node B can also send data to node A. We extend the state of the art in considering Langevin dynamics on directed graphs.

1 Introduction

Recently, there has been a surge in the interest of using computing platforms situated on a network, modeled as a graph with vertices and edges. Originally termed *distributed*, this has more recently transformed as being denoted as *decentralized*, to contrast with data-parallel distributed computation methods in high performance computing. In this setting, a number of agents, defined as vertices $\mathcal{V}$ with communication links $\mathcal{E}$ in a fully connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ perform alternating sequences of steps of local computation and consensus-based communication to cooperatively solve some problem in optimization, learning, or inference. This line of research began with the seminal work on optimization in [10] although precursors in network control theory exist.

Preliminary work. Under review by AISTATS 2021. Do not distribute.

The rise of distributed algorithms can come from an underlying physical reality of a problem incorporating information that is distributed across a network of agents that they must cooperatively solve, or a situation arising from the contemporary age of "big data" in statistics and machine learning. In this case one computer is unable to store the entire dataset for learning and there is a practical necessity for separating it across a set of machines. As soon as we divide data to many computers, then the connection of such computers will constitute the distributed network. In case of the absence of a central master/server machine the network is described as decentralized.

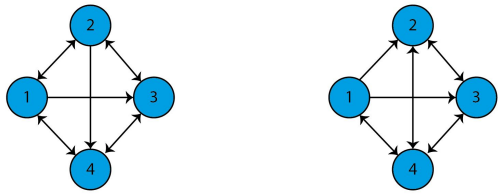
On undirected graphs, each communication link, modeled as an edge in the graph $e = (v_1, v_2) \in \mathcal{E}$ for $v_1, v_2 \in \mathcal{V}$ is such that communication is bi-directional, i.e., in this case $(v_2, v_1) \in \mathcal{E}$ as well. In a *directed* graph, this may not in general be the case, i.e., it could be that node v_1 can send data to note v_2, but not vice versa. In general, connections can be *time-varying*, meaning that nodes connect and drop out at various moments in time. This issue of communication crashes is one of practical importance as it is important that the optimization, inference or learning procedure be robust with respect to such occurrences.

As an illustration, let us consider the multi-agent system that contains 4 nodes depicted in Figure 1. Let the connection between the nodes at a certain time t_1 be as in Figure 1(a). Subsequently, for some reason the connection between from first to the second agent fails, then the graph loses the directed edge between them at a certain time t_2 that is depicted in figure 1(b). Hence, the network is time-varying directed graph.

Having realised the structure of a decentralized multi-agent system, one can move on the problem statement. The task is to sample from a probability distribution that is known up to a normalizing constant. For instance, such a problem arises in Bayesian Inference, when we want to calculate a posterior distribution $p(\theta|x)$ for a parameter as θ, knowing a prior knowledge $p(\theta)$ and a likelihood $p(x|\theta)$, where x is data. In accordance with the Bayes' theorem (1)

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{\int_\Theta p(x|\theta)p(\theta)d\theta}, \tag{1}$$

there is a constant in a denominator that constitutes the multiplication of the prior and the likelihood, in-



(a) at time t_1 (b) at time t_2

Figure 1: Time Varying Directed Network

tegrated across the parameters. This integral normalizing constant is typically difficult if not impossible to compute, and hence sampling methods often seek to avoid doing so.

Markov Chain Monte Carlo (MCMC) methods are able to obtain diverse samples from a target posterior distribution. The core of the techniques is to sample a proposal distribution which is then accepted or rejected. There are many variants of these proposal distributions. More recently, it was found that, for especially log concave potentials in general, discretizations of physically motivated stochastic differential equations are able to efficiently for distributions with a high dimension for the parameter θ. One of these proposals is based on Hamiltonian dynamics, which was observed at first in [6], while Langevin dynamics underlies another type of proposal, introduced for posterior Bayesian inference with the seminal work [17]. The overdamped Langevin diffusion equation, in particular, arrives at a stationary distribution characterized by a pdf of the form $e^{-U(x)}/Z$, where Z is the normalizing constant, when the diffusion drift term is $\nabla U(x)$. This form of potential is common for Gibbs type distributions and likelihood functions arising from exponential families.

Formally, we consider finding the stationary distribution of a pdf wherein there exists a potential given by,

$$U(x) = \sum_{i=1}^{m} U_i(x)$$

where node i knows only the strongly convex function U_i: $\mathbb{R}^d \to \mathbb{R}$.

We make the following assumption on $\{U_i(\cdot)\}$.

Assumption 1.1. *Each $U_i(x)$ is Lipschitz continuously differentiable with constant L, and strongly convex with constant μ. Furthermore $\sum_{i=1}^{m} U_i(x)$ is also Lipschitz continuously differentiable and strongly convex with, without loss of generality, the same constants.*

We can consider that the mode of the distribution, X^* is a non-empty set satisfying

$$X^* = \arg\min_{X \in \mathbb{R}^d} U(x)$$

.

2 Network

2.1 The general structure

As described above, the multi-agent system (network) constitutes directed and time-varying connections between nodes with changing in- and out-neighbours. The communication structure is modeled as a graph denoted by $\mathcal{G}(\mathcal{V}, \mathcal{E})$. That is defined as a set of vertex $\mathcal{V} = \{1,...,m\}$ and we will use $\mathcal{E}(t)$ to label a set of edges at a certain time t throughout the article.

Let us make some assumptions on the network. It was shown in the related work [8], that the property of B-strongly-connectedness is sufficient to derive different bounds on the speed of information propagation. So we require that the sequence $\{\mathcal{G}(t)\}$ is B-strongly-connected. In other words, there exists positive integer B such that the graph's edge set is strongly connected for any non-negative k. Formally:

$$\mathcal{E}_B(k) = \bigcup_{i=kB}^{(k+1)B-1} \mathcal{E}(i)$$

Since the observed multi-agent system is directed and time-varying, then one has to introduce in- and out-neighboors for each node i at the current time t.

$$\mathcal{N}_i^{in}(t) = \{j | (j,i) \in \mathcal{E}(t)\} \cup \{i\}$$

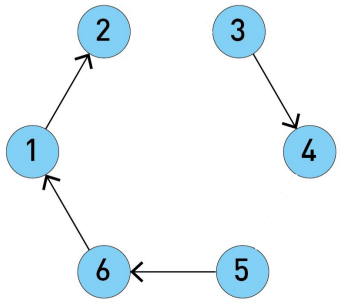
$$\mathcal{N}_i^{out}(t) = \{j | (i,j) \in \mathcal{E}(t)\} \cup \{i\}$$

The authors of the article [8] note that subgradient-push, which was developed in the article, requires only knowledge of out-degree for each node i. So, one should define the out-degree of node i at time t as:

$$d_i(t) = |\mathcal{N}_i^{out}(t)|$$

Now, let us introduce the mixing matrix $A(t)$. As an illustration, consider the graph that is depicted in Figure 2.

This graph is composed of 6 nodes and at this moment it is not fully connected (but is expected to complete its links over time). Taking the first agent, one can see, that it sends his signal to the second agent and



(a) The graph

Figure 2: Almost circular graph

itself. Let us then let a_{11} equal to $\frac{1}{2}$ and a_{21} the same. Continuing, the mixing matrix for the graph is written in (2). One can notice, that the sum of elements over any column is equal to 1, so the matrix $A(t)$ is a column-stochastic.

$$A(t) = \begin{bmatrix} \frac{1}{2} & 0 & 0 & 0 & 0 & \frac{1}{2} \\ \frac{1}{2} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \end{bmatrix} \tag{2}$$

2.2 Elements of spectral theory

Let us present some basic background on the spectral theory of mixing matrices. Introduce the following constants λ and δ associated to a graph sequence $\mathcal{G}(t)$. The value of δ corresponds to, once you continuously multiply the matrices $A(t)$, the smallest value of an entry of that product matrix. There is always a lower bound and in accordance assumption V.1 in [14], the standard condition is,

$$\delta \geq \frac{1}{m^{mB}}$$

One can interpret this as a lower bound on the weight any node assigns to past information from another node.

As for λ, for the case of fixed graphs, the variable λ will represent the connectivity of the graph, and it is usually the second eigenvalue of the graph adjacency matrix. Usually, the better connected the graph is, the bigger λ is, for poorly connected graphs λ is almost zero. However, when the graph changes with time, there is no direct interpretation via the eigenvalue, because every graph in the sequence will have a different eigenvalue.Thus one can interpret λ in the time-varying case as a lower bound for the connectivity of the time-varying graph.

$$\lambda \leq \left(1 - \frac{1}{m^{mB}}\right)^{\frac{1}{mB}}$$

3 Previous Work

Stochastic Gradient Langevin Dynamics was introduced for posterior Bayesian inference with the seminal work [17].

The paper [4] presented the first application of applying Langevin dynamics, and thus finding a stationary distribution, in the context of a decentralized setting of data distributed across a network. The work [3] extended the framework to consider the momentum based Hamiltonian Langevin dynamics. Finally [12] considered the problem relaxing the conditions for convexity of the potential (log-concavity of the distribution function) to a log-Sobolev inequality, thus permitting posterior inference for a wider class of decentralized problems.

In regards to decentralized optimization, besides the seminal work [8], there is the paper considering stochastic gradients in [9]. For a nice recent survey on decentralized optimization in machine learning see [7].

3.1 Probability Distances

In order to present our theoretical results, we must introduce some notation regarding the computation of the distance between two probability distributions. A transference plan $\zeta(\mu, \nu)$ of two probability measures μ and ν on $\mathcal{B}(\mathbb{R}^d)$ is itself a probability measure on $(\mathbb{R}^d \times \mathbb{R}^d, \mathcal{B}(\mathbb{R}^d \times \mathbb{R}^d))$ that satisfies: for all measurable $A \subseteq \mathbb{R}^d$, it holds that $\zeta(A \times \mathbb{R}^d) = \mu(A)$ and $\zeta(\mathbb{R}^d \times A) = \nu(A)$. We denote by $\Pi(\mu, \nu)$ the set of transference plans of μ and ν.Two $\mathbb{R}^d$-valued random variables (X, Y) is a *coupling* of μ and ν if there exists a $\zeta \in \Pi(\mu, \nu)$ such that (X, Y) are distributed according to ζ. The Wasserstein distance of order two is,

$$W_2(\mu, \nu) = \left(\inf_{\zeta \in \Pi(\mu,\nu)} \int_{\mathbb{R}^d \times \mathbb{R}^d} \|x - y\|^2 d\zeta(x, y)\right)^{1/2}.$$

For all μ, ν there exists a $\zeta^* \in \Pi(\mu, \nu)$ realizing the inf, i.e., for any coupling (X, Y) distributed according to ζ^* we have $W_2(\mu, \nu) = \mathbb{E}[\|X - Y\|^2]^{1/2}$, defined as the *optimal transference plan* and *optimal coupling* associated with W_2. The space $\mathcal{P}_2(\mathbb{R}^d)$ is the set of finite second moment probability measures and together with W_2 is a complete separable metric space.

We denote $\mu \ll \nu$ to mean that μ is absolutely continuous w.r.t. ν. Recall the Kullback-Leibler (KL)

divergence of μ from ν is defined by,

$$KL(\mu|\nu) = \begin{cases} \int_{\mathbb{R}^d} \frac{d\mu}{d\nu}(x) \log\left(\frac{d\mu}{d\nu}(x)\right), & \text{if } \mu \ll \nu, \\ \infty & \text{otherwise.} \end{cases}$$

4 Algorithm

Now, we propose an algorithm to solve the problem of sampling from the target distribution in a decentralized setting with directed graphs. As already mentioned, the perturbed push-sum protocol, which was published in the work [8] is at the heart of our developed algorithm. The aspect of the procedure seeking consensus is actually exactly the same. One should take the mixing matrix, which is set by the current graph's structure, and multiply this matrix $A(t)$ by the vector of coordinates X, where $X_{(i)}$ is a stack of components of coordinate on i'th node. As soon as we compute the consensus step, then we can compute the balancing vector $Y_{(i)}$ for each agent. The (stochastic) gradient is evaluated at another vector $Z_{(i)}$, and the update, together with the Gaussian noise, is applied to $X_{(i)}$.

The set of quantities available and computed by each agent is thus,

$$\left\{W_{(i)}(t), Y_{(i)}(t), Z_{(i)}(t), X_{(i)}(t), d^{out}_{(i)}(t)\right\}$$

We now present the specific Algorithm, as inspired by the merging of the perturbed push sum approach given in [8] and the unadjusted Langevin algorithm, as described for instance in [2].

As a quick formality, we introduce the notion of the operation of a Hadamard division. As known, there is the operation that is responsible for element-wise multiplication between two matrices(vectors) and the main requirement for them is the same dimension of these matrices(vectors). This multiplication is termed as the Hadamard product. At the same time, one can define the element-wise division for two matrices(vectors). However, there is additional requirement to these vectors with the exception of the same dimension: one has to demand non-zero elements of the vector that appears in the denominator. Formally, whereas the Hadamard product is defined as $(x_1, ..., x_n) \otimes (y_1, ..., y_n)$, one can do the same for Hadamard division. Then, it will be as follow: $(x_1, ..., x_n) \oslash (y_1, ..., y_n)$ and $\forall i : y_i \neq 0$

The Algorithm, from the perspective of agent i, is given in the set of equations (3)

$$\begin{cases} W_{(i)}(t+1) = \sum\limits_{j \in \mathcal{N}_i^{in}(t)} \frac{X_{(j)}(t)}{d_j(t)} \\ Y_{(i)}(t+1) = \sum\limits_{j \in \mathcal{N}_i^{in}(t)} \frac{Y_{(j)}(t)}{d_j(t)} \\ Z_{(i)}(t+1) = \frac{W_{(i)}(t+1)}{Y_{(i)}(t+1)} = |W(t+1) \oslash Y(t+1)|_{(i)} \\ X_{(i)}(t+1) = \sum\limits_{j \in \mathcal{N}_i^{in}(t)} \frac{X_{(j)}(t)}{d_j(t)} \\ \qquad -\alpha(t+1)\nabla U_{(i)}(Z_{(i)}(t+1))) \\ \qquad +\sqrt{2\alpha(t+1)}B(t+1)_{(i)} \end{cases} \tag{3}$$

where we denote $B(t+1) = \sqrt{\alpha(t+1)}\xi(t+1) + R(t+1)$, with $\xi(t+1)$ is a zero bias bounded variance stochastic gradient error and $R(t+1)$ is an isotropic Gaussian random variable. Let us set $\sqrt{\alpha(t+1)} \leq 1/\sigma^2 := 1/\mathbb{E}\|\xi\|$ so that the standard deviation associated with $B(t+1)$ is always less than 2.

Let us now consider the iterations on the full stack of vectors $\{X, Y, Z\}$ using the mixing matrix A,

$$\begin{cases} W(t+1) = A(t)X(t) \\ Y(t+1) = A(t)Y(t) \\ Z(t+1) = A(t)X(t) \oslash A(t)Y(t) \\ X(t+1) = A(t)X(t) - \alpha(t+1)\nabla U(Z(t+1)) \\ +\sqrt{2\alpha(t+1)}B_{t+1} \end{cases} \tag{4}$$

In the theory (and of course the numerical implementation), we shall consider the discretization, however, out of mathematical interest, we can note that, considering the form of the Euler-Maruyama (EM) discretization, writing $A(t)X(t) = A(t)X(t) + X(t) - X(t) = (A(t) - I)X(t) + X(t))$, we can notice that this corresponds to the EM discretization of the dynamics given by, (where we now overload t to be continuous)

$$\begin{aligned} dY_t &= (A_t - I)Y_t \\ dX_t &= (A_t - I)X_t - \alpha_t \nabla U(Z_t) + \sqrt{2\alpha_t}dB_t \end{aligned}$$

. where dB_t is a Brownian motion term and clearly the discretization is with a step-size of one.

5 Theoretical Results

We present the sequence of Lemmas and our final convergence result in measure for the Algorithm defined in Section 4. The proofs of the statements are left to the Supplementary Material.

First we will require a bound in expectation on the gradient vectors, which we derive by deriving a bound on the expectation of the norm of the vectors on which they are evaluated, $Z_{(i)}(t)$.

Lemma 5.1. *It holds that there exists some compact set $\mathcal{V}$ such that for all i*

$$\mathbb{E}\|X_{(i)}(t+1)\| \leq \begin{cases} \|Z_{(i)}(t+1)\| & \text{if } Z_{(i)}(t+1) \in \mathcal{V} \\ R & \text{otherwise} \end{cases}$$

with R depending on $\mathcal{V}$ and problem constants.

Lemma 5.2. *It holds that there exists a C such that,*

$$\mathbb{E}\|\nabla U_i(Z_{(i)}(t+1))\| \leq C$$

The next statement is the same as Corollary 1 in [9],

Lemma 5.3. *It holds that,*

$$\left\| Z_{(i)}(t+1) - \frac{\sum_{i=1}^{m} X_{(i)}(t)}{m} \right\| \leq \frac{8}{\delta}\left(\lambda^t \sum_{i=1}^{m} \|X_{(i)}(0)\|_1 + \sum_{s=1}^{t} \lambda^{t-s} \sum_{i=1}^{m} \|e_i(s)\|\right)$$

where,

$$e_i(s) = \alpha(s+1)\nabla U_i(Z_{(i)}(s+1))) + \sqrt{2\alpha(s+1)}B_{(i)}(s+1)$$

This together with Lemma 5.2 allows us to prove the following bound on the running sum of the consensus error.

Lemma 5.4.

$$\mathbb{E}\left[\sum_{t=1}^{\tau} \left\| Z_{(i)}(t+1) - \frac{\sum_{i=1}^{m} X_{(i)}(t)}{m} \right\|\right] \leq \frac{8}{\delta}\frac{\lambda}{1-\lambda}\sum_{i=1}^{m}\|X_{(i)}(0)\|_1 + \frac{8}{\delta}\frac{Dm}{1-\lambda}(1+\sqrt{\tau})$$

This implies,

Corollary 5.4.1. *Given any $\gamma > 0$, there exists $C_\gamma > 0$ such that,*

$$E\|\bar{X}(t)) - Z_{(i)}(t+1)\| \leq \frac{C_\gamma}{t^{1/2-\gamma}}$$

Now define,

$$\bar{X}(t) = \frac{\sum_{i=1}^{m} X_{(i)}(t)}{m}$$

we have, by the column-stochasticity of $A(t)$,

$$\bar{X}(t+1) = \bar{X}(t) - \frac{\alpha(t+1)}{m}\sum_{i=1}^{m}\nabla U(\bar{X}(t)) + \frac{\alpha(t+1)}{m}\sum_{i=1}^{m}\left(\nabla U(\bar{X}(t)) - U(Z_{(i)}(t+1))\right) + \bar{B}(t+1) \tag{5}$$

Let $\bar{\nu}_t$ be the distribution associated with $\bar{X}(t)$

We are now ready to use the arguments in [1] regarding perturbed Langevin methods. In particular, we can apply Proposition 2 to state that,

Lemma 5.5.

$$W_2(\bar{\nu}_{t+1}, \pi) \leq \rho_{t+1} W_2(\bar{\nu}_t, \pi) + 1.65L(\alpha_{t+1}^3 d)^{1/2} + \alpha(t+1)\sqrt{d}L\sum_{i=1}^{m}\mathbb{E}\|\bar{X}(t) - Z_{(i)}(t+1)\|$$

where $\rho_{t+1} = \max(1-\mu\alpha(t+1), L\alpha(t+1)-1)$.

Now recall

Lemma 5.6. *[13, Lemma 2.4] Let $u_k \geq 0$ and,*

$$u_{k+1} \leq \left(1 - \frac{c}{k}\right)u_k + \frac{d}{k^{p+1}}$$

with $d > 0$, $p > 0$ and $c > 0$ and $c > p$. Then,

$$u_k \leq d(c-p)^{-1}k^{-p} + o(k^{-p})$$

Apply the previous two Lemmas together with Corollary 5.4.1 to conclude,

Theorem 5.7. *Let $\alpha(0) \leq \frac{1}{2\max(\mu, L)}$ and $\alpha(t) = \alpha(0)/(1+t)$. We have that,*

$$W_2(\nu_{k+1}, \pi) \leq \frac{(C_\gamma m + 1.65)Ld^{1/2}}{(\mu\alpha(0) - 1/2 + \gamma)(1+t)^{(1/2-\gamma)}} + \beta_k$$

where $\beta_k = o\left((1+t)^{-(1/2-\gamma)}\right)$

6 Numerical Experiments

6.1 Notes on Parameter Tuning

There is a classical practical question in regards to implementing SGLD algorithms for sampling–how do we tune the step-size, considering its dual influence on the convergence as well as the variance of Brownian term? There are some papers, whose main goals are to study the question. The first mentioned related work [11] solves the problem via finding the choice that minimizes the asymptotic KL-divergence. However, it applies only in the case of independent proposal distributions, when new samples are generated regardless of the history. Thus, applying such approach to classic proposals as Random Walk Metropolis (RWM) or Metropolis-Hastings adjusted Langevin algorithm (MALA), it leads to a collapsed delta function solution. Recently, another article [16] was published extending [11]. Optimizing a special speed measure ((2) in [16]), one can obtain the algorithm which can tune the optimal step-size as well as covariance matrix for the best convergence. Also, it is worth noticing, that entropy is at the heart of Titsias's approach. However, entropy is not the only way to solve such problems. For instance, [15] and [5] use other methods to get optimal parameters, although the second paper applies their method for HMC. However, we prefer to pick out the Titsias's approach for obtaining optimal step-sizes in our algorithm.

6.2 Bayesian Linear Regression

In this subsection, we study the algorithm's ability to converge to a desired distribution arising from Bayesian Linear Regression. We consider a multi-agent network composed of 4 nodes. Classical linear regression is described by the following expression:

$$y = Xw + \delta$$

where y constitutes a target variable and the noise is denoted by δ, whereas X and w are a set of features and weights of the linear model, respectively. We can generate data as follows:

$$\delta \sim \mathcal{N}(0, \sigma^2), \quad X_j \sim \mathcal{N}(0, I_1) \quad \forall j$$

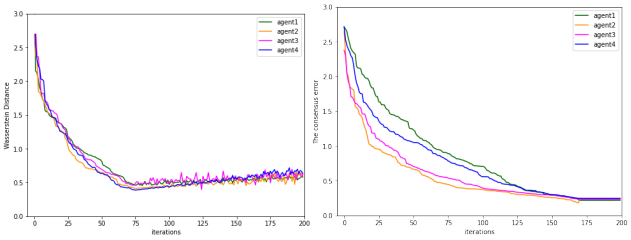
We generate 800 samples and separate 200 samples across each of four machines. Thus, each agent is going to process its 200 samples, not having access to samples of the other agents. Our main goal is to get the posterior distribution of weights. However, to facilitate the problem we are going to use "poor" Bayesian inference. In other words, one would like to find out the mode of the posterior distribution. Since we use Bayesian linear regression, we should introduce a prior distribution for weights. Implying features' equal a priori significance, we choose a zero-mean standard normal distribution with an identity 2 dimensional covariance matrix. Recall Bayes's theorem as follows:

$$p(w|X, y) = \frac{p(y|X, w)p(w)}{\int_w p(y|X, w)p(w)dw}$$

where $p(w)$ constitutes the prior distribution for weights, while $p(y|X, w)$ is a likelihood of data. In the experiment, each node will take its own mini-batch, whose size is equal to 1, will calculate consensus step and point where we take the gradient. Having done these computations, each node samples a new object from the posterior distribution of weights. Since our posterior distribution is a multivariate normal distribution and a distribution that makes a current sample on a node is the same, one can calculate the second Wasserstein distance between this distributions analytically as follows:

$$\mathcal{W}_2(\mathcal{N}(m_1, \Sigma_1), \mathcal{N}(m_2, \Sigma_2)) = \|m_1 - m_2\| + \|\Sigma_1^{\frac{1}{2}} - \Sigma_2^{\frac{1}{2}}\|$$

Thus, we repeat this process for 200 iterations (1 batch per one iteration, i.e., one epoch), sometimes changing the set of edges. In figure 3(a), one can look at the final plot that deals with the convergence between distributions in terms of the second Wasserstein distance. Meanwhile, one can see the consensus error for each one from 4 agents in figure 3(b).



(a) The 2nd Wasserstein distance (b) The consensus error

Figure 3: The Second Wasserstein Distance and the consensus error in case of Bayesian linear regression

6.3 Sampling from Gaussian mixture

In this subsection, we illustrate an experiment that deals with sampling from a multi-modal probability distribution. Recall the definition of a Gaussian mixture:

$$\sum_{k=1}^{n} \phi_k \mathcal{N}(m_k, \Sigma_k)$$

where a the mixture constants ϕ_k satisfy $\sum_{k=1}^{n} \phi_k = 1$. Let us consider a mixture of two one-dimensional Gaussian distributions under the assumption that their standard deviations are known, but not their means. However, there are some prior distributions for the means of both Gaussians. Then, we get for each sample x_i, that:

$$x_i \sim \frac{1}{2}\mathcal{N}(\theta_1, \sigma_x^2) + \frac{1}{2}\mathcal{N}(\theta_1 + \theta_2, \sigma_x^2) \tag{6}$$

where σ_x^2 equals to 2. As for prior distributions, they will be as below:

$$\theta_1 \sim \mathcal{N}(0, \sigma_1^2), \theta_2 \sim \mathcal{N}(0, \sigma_2^2)$$

where σ_1^2 and σ_2^2 equal to 10 and 1 respectively. Now, one can fix $\theta_1 = 0$ and $\theta_2 = 1$. Thus, having fixed values of means, we get the certain mixture of two Gaussian distributions. In this experiment, we consider a decentralized system that is composed of 4 nodes. Then, one can generate 800 samples from the distribution of x_i defined above (6) with these given θ_1 and θ_2 and randomly and separate the samples evenly across the network.

Then, the main goal of the experiment is to propose a posterior distribution for θ_1 and θ_2 from the set of samples, specifically,:

$$p(\vec{\theta}|X, \sigma_x, \sigma_1, \sigma_2) \sim [\prod_{i=1}^{n} p(X_i|\vec{\theta}, \sigma_x)]p(\vec{\theta}|\sigma_1, \sigma_2)$$

. Using MCMC methods and taking the gradients of joint log-likelihood, one can obtain samples from a

desired distribution. We apply the Algorithm given in (3), where we take the gradient of a joint log-likelihood, for each node, for its set of samples, along with the consensus step at each iteration.. The step-size is diminishing and is taken as $\alpha(t) = \frac{a}{(b+t)^\gamma}$, where γ equals to 0.65, when a and b are set such that $\alpha(t)$ is changed from 0.01 to 0.0001. In Figure 4: (a) - (d), one can see how each agent is able to sample from Gaussian mixture posterior distribution above.

Moreover, one would like to look at an error between each node and average over all nodes on each iteration. Such difference between "average" node and an agent of decentralized network one can call as "the consensus error". In other words, "Tte consensus error" shows up at all the following expression on each iteration for each node in the multi-agent network:

$$\|X_i(t) - \bar{X}(t)\|^2$$

Thus, one should take values, which are generated by the algorithm from i'th node, and calculate the squared difference on each t iteration. One can look at the consensus error for 4 agents in case of sampling from Gaussian mixture in figure 4(e).

6.4 Bayesian Logistic Regression

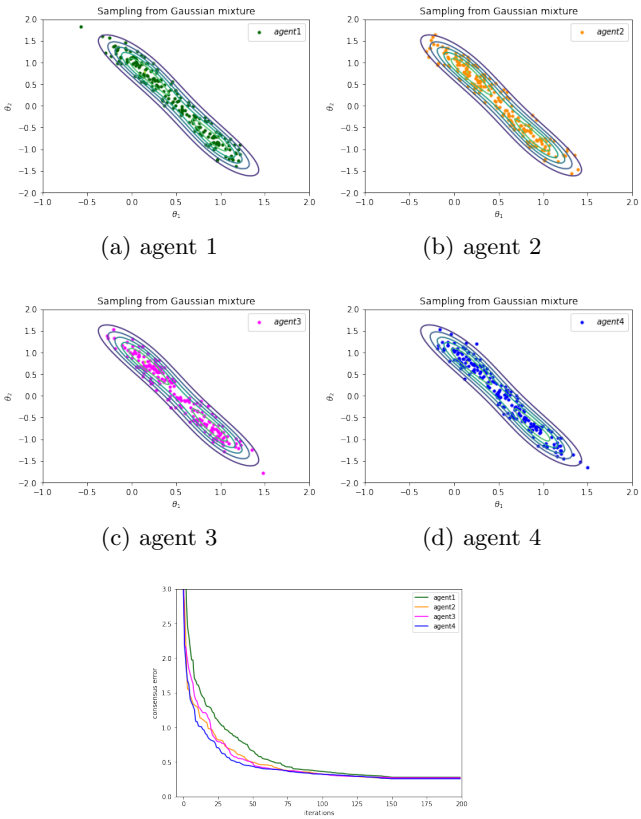
In the final subsection, we apply our algorithm, which is based on stochastic gradient Langevin algorithm, to a case of Bayesian logistic regression for real data. Specifically we consider the problem of binary classification, with a target variable y $\in \{-1, 1\}$. The a9a data-test is a huge-scale and classic data-set for binary classification. This data-set is available at the UCI machine learning repository. It is composed of 32561 samples and 123 features. Each feature has 2 unique values 1 or 0, whereas the target variable has values 1 and -1. In this experiment our decentralized setting is composed of 4 agents. Then, we take 80% of data as train data-set and separate between nodes equally and randomly.

We have the following expression for the posterior for N observations:

$$\pi(w) = p(w|x, y) = \frac{1}{C}\prod_{i=1}^{N} p(y_i|x_i, w)p(w)$$

where C is a normalizing constant. Then we take (3) with $\nabla U_i(Z(t+1)) = \nabla\left(\prod_{i\in S_i} - \log p(y_i|x_i, Z)p(Z)\right)$ where $S_i \subseteq [N]$ the subset of data given to agent i. One can write the expression for the gradient of potential function as follows:

$$\nabla U(Z(t+1)) = \sum_{i=1}^{m} \frac{y_i x_i}{1 + e^{-y_i w^T x_i}} - sign(w)$$

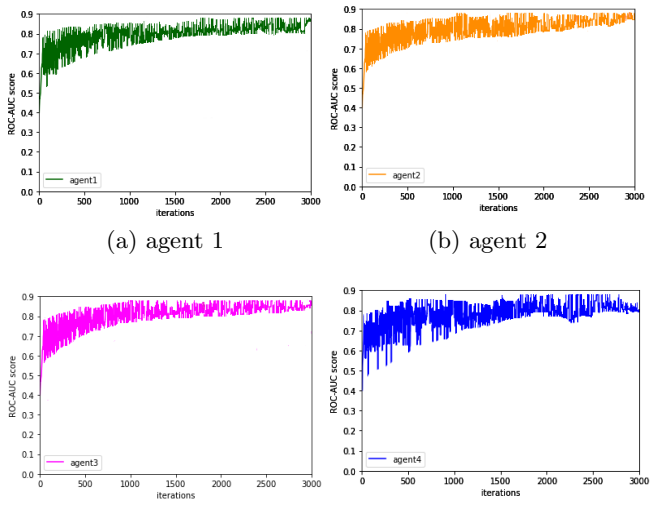


(a) agent 1 (b) agent 2

(c) agent 3 (d) agent 4

(e) The consensus error

Figure 4: (a) -(d) :The sampling from Gaussian mixture for 4 agents
(e) - The consensus error for 4 agents



(a) agent 1 (b) agent 2

(c) agent 3 (d) agent 4

Figure 5: ROC-AUC curve in Bayesian logistic regression

Having computed a consensus solution and weights, where we take the gradient of the potential function for each node, one can recalculate a new value for w and compute the ROC-AUC score on test data-set, whose the size is equal to 20% of the whole data-set, for each node. The step-size constitutes diminishing step-size and is equal to $\frac{\alpha(0)}{(\gamma+t)^{\phi}}$, where $\alpha(0) = 0.008$, $\gamma = 12$, $\phi = .45$. In Figure 5, we plot the ROC-AUC curve for each node in this decentralized setting. We see the convergence during one epoch (one iteration through the data). The average of nodes reaches an ROC-AUC score of 84.76% in 1632 iterations.

7 Conclusion

In this paper we considered the problem of Langevin dynamics for sampling from a distribution with a potential function consisting of data distributed across agents. We consider a decentralized framework wherein the agents communicate by the structure of a directed graph. We proved asymptotic consensus as well as convergence in Wasserstein distance to the stationary distribution of the procedure, and demonstrated the efficacy of the procedure on standard test problems in sampling.

Intended future work can be studying the properties of the method as a means of finding globally optimal points for nonconvex noisy optimization problems, as well as quantization, study of large scale speedup, and other aspects of sampling in a federated setting.

References

[1] Arnak S Dalalyan and Avetik Karagulyan. User-friendly guarantees for the langevin monte carlo with inaccurate gradient. *Stochastic Processes and their Applications*, 129(12):5278–5311, 2019.

[2] Alain Durmus and Eric Mouline. High-dimensional bayesian inference via the unadjusted langevin algorithm. *arXiv preprint arXiv: 1605.01559*, 2016.

[3] Mert Gürbüzbalaban, Xuefeng Gao, Yuanhan Hu, and Lingjiong Zhu. Decentralized stochastic gradient langevin dynamics and hamiltonian monte carlo. *arXiv preprint arXiv:2007.00590*, 2020.

[4] Vyacheslav Kungurtsev. Stochastic gradient langevin dynamics on a distributed network. *arXiv preprint arXiv:2001.00665*, 2020.

[5] Daniel Levy, Matthew D. Hoffman, and Jascha Sohl-Dickstein. Generalizing hamiltonian monte carlo with neural networks. *arXiv preprint arXiv: 1711.09268*, 2017.

[6] Radford M.Neal. Mcmc using hamiltonian dynamics. *arXiv preprint arXiv: 1206.1901*, 2012.

[7] Angelia Nedic. Distributed gradient methods for convex machine learning problems in networks. *IEEE Signal Processing Magazine 10.1109/MSP.2020.2975210*, 2018.

[8] Angelia Nedic and Alex Olshevsky. Distributed optimization over time-varying directed graphs. *arXiv preprint arXiv: 1303.2289*, 2014.

[9] Angelia Nedić and Alex Olshevsky. Stochastic gradient-push for strongly convex functions on time-varying directed graphs. *IEEE Transactions on Automatic Control*, 61(12):3936–3947, 2016.

[10] Angelia Nedic and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

[11] Kirill Neklyudov, Evgenii Egorov, Pavel Shvechnikov, and Dmitry Vetrov. Metropolis-hastings view on variational inference and adversarial training. *arXiv preprint arXiv:1810.07151*, 2018.

[12] Anjaly Parayil, He Bai, Jemin George, and Prudhvi Gurram. A decentralized approach to bayesian learning. *arXiv preprint arXiv:2007.06799*, 2020.

[13] Boris T Polyak. *Introduction to optimization*. Number 04; QA402. 5, P6. 1987.

[14] Alexandr Rogozin, Cesar A. Uribe, Alexandr Gasnikov, Alexandr Malkovsky, and Angelia Nedic. Optimal distributed optimization on slowly time-varying graphs. *arXiv preprint arXiv: 1805.06045*, 2018.

[15] Jiaming Song, Shengjia Zhao, and Stefano Ermon. A-nice-mc: Adversarial training for mcmc. *arXiv preprint arXiv: 1706.07561*, 2017.

[16] Michalis K. Titsias and Petros Dellaportes. Mcmc using hamiltonian dynamics. *arXiv preprint arXiv: 1911.01373*, 2019.

[17] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688, 2011.