# Distributed Federated Learning in two time scale

November 2020

## 1 Problem Formulation

Reinforcement learning (RL) algorithms are used to solve problems which can be modelled as a Markov Decision Process (MDP's) where the dynamics of underlying process are unknown. In this paper we aim on solving a policy evaluation problem across multiple agents in a federated learning setting. Federated learning is a multi agent machine learning technique which trains across multiple agents, in which each agent communicates local parameters to the centralized server instead of sending data samples. Policy evaluation is a basic problem in RL, where given a policy our goal is to compute value function associated with the policy. As the state space of MDP can be very large we use linear function approximation and assume that parameters are updated using Gradient TD Learning (GTD).

For each agent $i$, an MDP is defined by $(X_i, A_i, P_i, R_i, \gamma)$, where $X_i$ is the state space of agent $i$, $A_i$ the action space, $P_i : (X_i x A_i) \to X_i$, where $P_i$ is the probability transition function for agent $i$. $R_i : (X_i x A_i) \to \mathcal{R}$ is the reward function and $\gamma$ is the discount factor. Each agent follows a fixed stationary policy $\pi_i : X_i \to A_i$. We intend to learn a set of parameters to approximate value function across all the agents (with different MDPs) using GTD. But, when approximating value function using linear function approximation the GTD can be modeled as solving system of linear equations. We consider this problem of policy evaluation in distributed federated learning setting and model it as problem of finding solution $(x^*, y^*)$ of the following system of linear equations:

$$A_{11}x^* + A_{12}y^* = \frac{1}{N}\sum_{i=1}^{N} b_1^i \tag{1}$$

$$A_{21}x^* + A_{22}y^* = \frac{1}{N}\sum_{i=1}^{N} b_2^i \tag{2}$$

We assume that the matrices $A_{ij}$ and $b_1^i, b_2^i$ are unknown. Instead we have access only to noisy samples of them, which motivates us to consider the linear two-time scale stochastic approximation, iteratively updating the estimates $(x_k, y_k)$ of $(x^*, y^*)$ as:

$$x^i_{k,t+1} = x^i_{k,t} - \alpha_k\big(A_{11}(\xi^i_{k,t})x^i_{k,t} + A_{12}(\xi^i_k)y^i_{k,t} - b^i_1(\xi^i_{k,t})\big)$$
$$y^i_{k,t+1} = y^i_{k,t} - \beta_k\big(A_{21}(\xi^i_{k,t})x^i_{k,t} + A_{22}(\xi^i_k)y^i_{k,t} - b^i_2(\xi^i_{k,t})\big)$$

$$x_{k+1} = \frac{1}{N}\sum_{i=1}^{N} x^i_{k+1,T}$$

$$y_{k+1} = \frac{1}{N}\sum_{i=1}^{N} y^i_{k+1,T}$$

Here $A_{11}(\xi^i_{k,t}), A_{12}(\xi^i_{k,t}), A_{21}(\xi^i_{k,t}), A_{22}(\xi^i_{k,t}), b^i_1(\xi^i_{k,t}), b^i_2(\xi^i_{k,t})$ are noisy observations where $\{\xi^i_{k,t}\}$ is a Markov chain. Here the sequence $\{\xi^i_{k,t}\}$ are the samples of a markov process. This markov process can be thought of being governed by MDP discussed above. In this case $\beta_k << \alpha_k$, therefore ($x^i_{k+1}$ is updated at faster time scale than $y^i_{k+1}$).

Now tying everything back to our problem statement discussed above, We have mentioned before that N agents communicate with a centralized server to share their local parameters and received global averaged parameters. We have modeled this also in our linear equation above. After every $L$ local updates (such that $k < L$), of $(x^i_{k+1}, y^i_{k+1})$, they are sent to the server for global averaging. Then after averaging each server sends these estimates $(x_{k+1}, y_{k+1})$ back to each agent which then become local estimates for next time step.

At any time $k \geq 0$, each agent agent $i$ gets $x_k$ and $y_k$ from the server. Each agent $i$ will run $T$ local steps of GTD, i.e., agent $i$ initializes $x^i_{k,0} = x_k$ and $y^i_{k,0} = y_k$ and then for $t = 1, \ldots, T-1$

# 2 Experiments

We investigate the convergence of the algorithm presented above in a experimental setting. For the first part of experiments we use 20 different maze like environments. Each maze consists of a start, end and blocks through which agent in the maze cannot traverse. Each maze environment differs from others by the location of the blocks. Each individual maze consists of an agent and all of the agents can communicate their parameter vector with a centralized server. Each agent communicates its parameters with the centralized server for every 't' time steps. Each agent follows a fixed deterministic policy. Overall, in our experimental setting a multi-agent multi task policy evaluation problem, where all the agents are collectively solving for parameters which approximate their respective value functions.

To analyze the convergence of parameters we look at NEU(Norm of Expected td Updates) as an indicator for convergence. We show that NEU converges to zero as the time steps increase. We also experiment with time interval of communication between the agents i.e. for how many time steps does each agent communicate with the centralized server. We found that the more frequently each agent communicates with the centralized server the faster the NEU decays.