EP3260: Fundamentals of Machine Learning over Networks
Computer Assignment 6, Due Date: April 29, 2020
Group 4: H. Hellström, M. Tavana, F. Vannella, and M. Zeng

Computer Assignment 6. *(Communication Efficiency)*
Split "MNIST" dataset to 10 random disjoint subsets, each for one worker, and consider SVM classifier in the form of $\min_{\boldsymbol{w}} \frac{1}{N} \sum_{i\in[N]} f_i(\boldsymbol{w})$ with $N = 10$. An alternative approach to improve communication-efficiency is to compress the information message to be exchanged (usually gradients, either in primal or dual forms). Consider two compression/quantization methods for a vector: (Q1) keep only K values of a vector and set the rest to zero and (Q2) represent every element with fewer bits (e.g., 4 bits instead of 32 bits).

(a) Repeat parts a-b from CA5 using Q1 and Q2. Can you integrate Q1/Q2 to your solution in part c of CA5? Discuss.

Solution. We consider multi-class SVM classifier which learns a vector model $\boldsymbol{w}_j$ for each class $j \in [M]$ and uses the loss function

$$L_i = \sum_{j\in[M]} \mathbb{I}\{j \neq y_i\} \left(\Delta + (\boldsymbol{w}_j - \boldsymbol{w}_{y_i})^T \boldsymbol{x}_i\right)_+, \quad (1)$$

annotate reference next time

for sample $\boldsymbol{x}_i$ with class label $y_i \in [M]$. We also add an extra bias dimension to $\boldsymbol{x}_i$ which is 1. The "MINST" dataset is partitioned into 10 disjoint subsets. The optimization problem is

$$\min_{\boldsymbol{w}} \frac{1}{10} \sum_{i\in[10]} f_i(\boldsymbol{w})$$

For Q1, we keep 25% values of a vector and set the rest to zero. For Q2, to make the results comparable to the case with Q1 compression, we used 16float $\boldsymbol{w}_j$ (default is 64float)

- **(part–a DGD)** We implemented the following algorithms over master-worker network with 10 workers and compared the convergence rates of them versus signaling which is denoted by T:
  - Decentralized gradient descent (DGD) without compression
  - Our proposed less communication and versions of DGD
  - DGD with Q1 and Q2 compressed signaling

  In Fig. 1, the loss function in the master node versus the number of $\boldsymbol{w}_j$ gradient exchanges (20 per iteration) is illustrated.

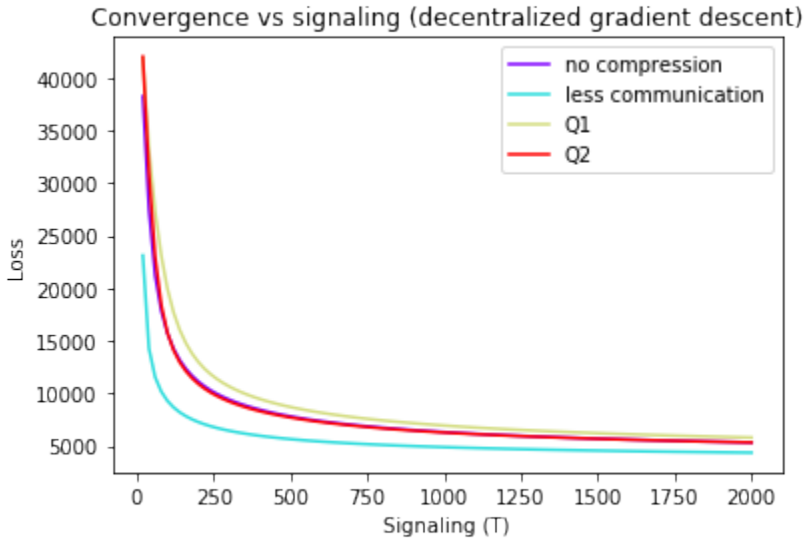

Figure 1: Convergence versus signaling for decentralized gradient decent (DGD).

Here, we can see that the convergence rate for (Q1) is slower than (Q2). This effect is due to the removing of 75% of the information in (Q1) compression. While in (Q2), the algorithm has more information about pixel values and the convergence is almost the same as DGD without compression.

- **(part–b)**
  - **DSM:** We implemented the following algorithms over 2-star network architecture with 10 agents and compared the convergence rates of then versus signaling which is denoted with T:
    - Decentralized subgradient method (DSM) without compression
    - Our proposed less communication and versions of DSM
    - DSM with Q1 and Q2 compressed signalings

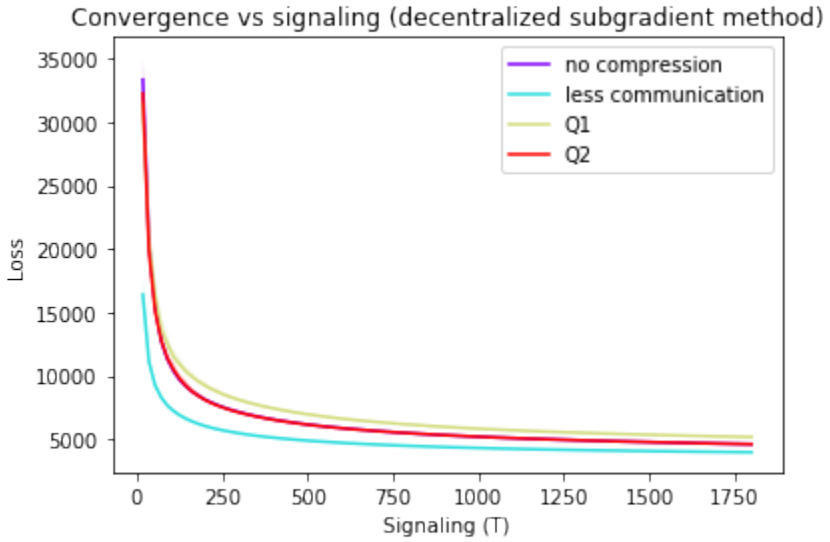

Figure 2: Convergence versus signaling for decentralized subgradient method (DSM).

  - **ADMM:** We implemented the following algorithms over 2-star network architecture with 10 agents and compared the convergence rates of then versus signaling which is denoted with T:
    - ADMM over network without compression
    - Our proposed less communication and versions of ADMM
    - ADMM with Q1 and Q2 compressed signalings

    In order to derive the update equations of ADMM over Network for multiclass SVM loss we use the auxiliary variables $u_{i,j} := (\boldsymbol{w}_j - \boldsymbol{w}_{y_i})^T \boldsymbol{x}_i$ and rewrite the problem with additional constrains as below

$$\begin{aligned} \text{min} \quad & \sum_a \sum_i \sum_{j \in [M]} (\Delta \mathbb{I}\{j \neq y_i\} + u_{i,j})_+ && && \text{(2a)} \\ \text{subject to} \quad & \boldsymbol{w}_{ja} = \boldsymbol{z}_{ja,l}, && \forall l \in \mathcal{N}_a, j \in [M] && \text{(2b)} \\ & u_{i,j} = (\boldsymbol{w}_j - \boldsymbol{w}_{y_i})^T \boldsymbol{x}_i, && \forall i, j && \text{(2c)} \end{aligned}$$

    After writing the dual problem using the augmented Lagrangian and simplifying the equations we get the following update rule for node a which is in fact very similar to DSM's rule

$$\boldsymbol{w}_{ja}^{k+1} = \frac{1}{2} \left(\boldsymbol{w}_{ja}^k + \frac{1}{|\mathcal{N}_a|} \sum_{l \in \mathcal{N}_a} \boldsymbol{w}_{jl}^k \right). \tag{3}$$

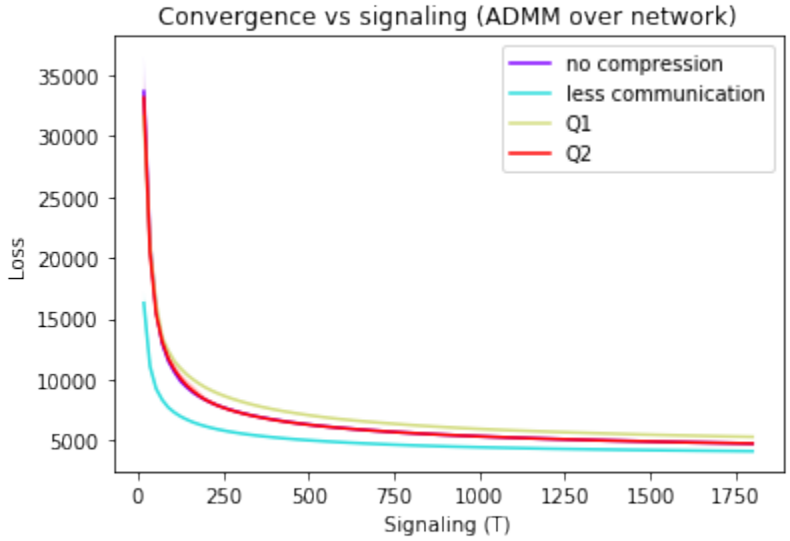

Figure 3: Convergence versus signaling for ADMM over network.

In Fig. 2 and Fig. 3, we see the average agent loss versus the number of $\boldsymbol{w}_j$ gradient exchanges (18 per iteration) for DSM and ADMM, respectively. It can be seen that the loss for DSM and ADMM are almost identical with small differences in final classification error and gradient norm after 10 iterations.

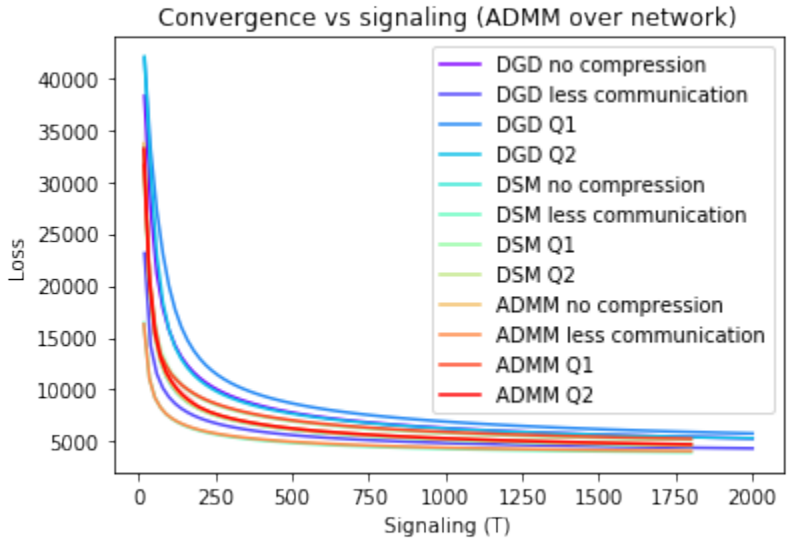

Figure 4: Convergence versus signaling for all methods

In Fig. 4, we have plotted the loss versus signaling T for all aforementioned methods.

□

(b) How do you make SVRG and SAG communication efficient for large-scale ML?

Solution.

- (SVRG) The algorithm is illustrated below.

Algorithm 1 SVRG (Johnson & Zhang, 2013)

Input: Epoch Length T, number of epochs K

for $k = 1$ to K **do**
  Compute all gradients and store $\widetilde{\nabla} f := \frac{1}{N} \sum_{i \in [N]} \nabla f_i \left(\widetilde{\boldsymbol{w}}_k \right)$
  Initialize $\boldsymbol{w}_{k,0} \leftarrow \widetilde{\boldsymbol{w}}_k$
  for $t = 1$ to T **do**
    Sample ζ_k uniformly from $[N]$
    $\boldsymbol{w}_{k,t} = \boldsymbol{w}_{k,t-1} - \alpha_k \left(\nabla_{\zeta_k} \left(\boldsymbol{w}_{k,t-1} \right) - \nabla_{\zeta_k} \left(\widetilde{\boldsymbol{w}}_k \right) + \widetilde{\nabla} f \right)$
  end for
  Update $\widetilde{\boldsymbol{w}}_{k+1} \leftarrow \boldsymbol{w}_{k,T}$
end for
return $\widetilde{\boldsymbol{w}}_{K+1}$

nice

The inner loop of SVRG can be implemented inside each worker/agent so that the agents only transmit the last $\boldsymbol{w}_{jT}^k$ and receive and consent over $\widetilde{\nabla} f$.

- (SAG) The algorithm is shown below.

Algorithm 2 SAG (Schmidt & Le Roux & Bach, 2012, 2017)

for $k = 1, 2, \ldots$ **do**
  Sample ζ_k uniformly from $[N]$ and observe $\nabla f_{\zeta_k} \left(\boldsymbol{w}_k \right)$
  Update for all $i \in [N]$, $\hat{g}_i \left(\boldsymbol{w}_k \right) = \begin{cases} \nabla f_i \left(\boldsymbol{w}_k \right), & \text{if } i = \zeta_k \\ \hat{g}_i \left(\boldsymbol{w}_{k-1} \right), & \text{otherwise} \end{cases}$
  Update $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \frac{\alpha_k}{N} \sum_{i \in [N]} \hat{g}_i \left(\boldsymbol{w}_k \right)$
end for

One can modify SAG in a way that the agents run SAG internally for a few iterations before sharing their information.

integration of your methods not given

□