Peer-to-peer parallel learning

Parallel learning over peer nodes without a central coordinating resource.

Performed over a network of sensors, mobile devices or geographically separated data centers.

Network is robust to failure due to lack of a central coordinating resource.

Challenges in peer-to-peer learning

- Non-iid data: The peer replicas may process non independent and identically distributed datasets. For example, physical sensors or data centers may collect biased samples.
- Random link failures: There may be intermittent link failures that may affect convergence for large scale learning

Goal: Provide a distributed consensus algorithm that provably converges in presence of non-iid data and link failures

Results

We integrate RWDDA with MALT and compare with model averaging with SVM[3] with the RCV1 dataset, both implemented over MALT[1].

Computation challenges

- Optimizing tricks

We only compute the sparse gradient and separately correct the regularizer.

Communicating z after each iteration strains the network.

We send z after nodes locally process a batch (500-5000) and adjust the learning rate to account for this batched communication.

Re-calculating z is expensive.

We maintain a running sum average over the dual, and only compute this sum only during reduce (i.e. when z parameters arrive).

Insights into RWDDA

Update steps in RWDDA

\[ z(t+1) \leftarrow z(t) + \frac{\sum_{i \in N(i)} z_i(t)}{|N(i)|+1} + \frac{g_i(t)}{|N(i)|+1} \]

Gradient scaling in RWDDA

- **Row-stochastic consensus protocol**

\[ P_0 = \lim_{t \to \infty} P^t = \begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix} \]

- **The stationary dist.**

\[ P \pi = 0 \]

Calculating \( P \) locally in RWDDA

- **Exact \( \pi \) is hard to obtain locally**

- **But**
  - random walk over undirected graph
  - \( \pi_i \propto \deg(i) + 1 \)
  - sufficient for this scaling purpose

\[ \pi_1 = \frac{\pi_2}{3} = \frac{\pi_3}{3} = \frac{\pi_4}{3} = \frac{\pi_5}{3} \]

Our Solution: RWDDA

Random-Walk Distributed Dual Averaging

- Row stochastic consensus protocol
- Robustify DDA(4) against node/edge failures
- Each node receives dual updates only from neighbors
- Borrows basic theories in random-walk over undirected graph

**Algorithm description**

- **f-th iteration**
  - **Broadcast**: Every node broadcasts to its neighbors
  - **Self-update**: Every node in parallel updates

**Design requirements**

- Effcient communication: RWDDA communication requirement is only one-sided (no hand-shake required). This is useful for asynchronous learning and using RDMA protocols that provide high speed one-sided semantics.
- Robust: The algorithm is robust to change in network topology or non-iid data.
- Provably convergent: We establish 1/sq root (t) convergence for the convex case. For the convergence analysis, please see our paper[2].
- Extensible: RWDDA is extensible to stochastic (sub) gradient and regularizers. Furthermore, asynchronous implementation is straightforward.

**Computation challenges**

- Optimizing tricks

We only compute the sparse gradient and separately correct the regularizer.

Communicating z after each iteration strains the network.

We send z after nodes locally process a batch (500-5000) and adjust the learning rate to account for this batched communication.

Re-calculating z is expensive.

We maintain a running sum average over the dual, and only compute this sum only during reduce (i.e. when z parameters arrive).

Goal: Provide a distributed consensus algorithm that provably converges in presence of non-iid data and link failures

References and Related Work


Algorithm insights:

- For the f-th iteration, there are two steps: broadcasting step and self-updating step. In the broadcasting step, every node broadcasts its dual variable \( z_i(t) \) to all its neighbors. In the subsequent self-updating step, every node is parallel following the same.

- Each node calculates and its gradient \( g_i(t) \). And then updates \( z_i(t+1) \) by simply negative scaling the dual variable.

- To implement RWDDA, each node only needs to know its neighborhood information, which makes the algorithm efficient.

- RWDDA is a convex function privately known by the agent \( i \).

- RWDDA method robustify the distributed dual averaging (DDA) method \[2\], based on row stochastic consensus protocol.

- Each node receives dual updates only from neighbors.

- The dual variables are equal weight of the received information and adds its most recent local subgradient scaled by \( |N(i)|+1 \).

- RWDDA's solution is a convex function privately known by the agent. The edge set \( E \) and a dual variable \( z_i(t) \) for each agent \( i \).

- With these notations, we are able to express (3.2) in a terser way. Imagine \( P \) is a row stochastic matrix. We maintain a running dual average \( \tilde{z}(t) \) as \( \tilde{z}(t) = \sum_{i \in N(i)} z_i(t) \).

- Furthermore, asynchronous execution is supported by simply negative scaling the dual variable.

- RWDDA borrows basic theories in random-walk over undirected graph, to get \( \tilde{z}(t) \) by just using the local information.

- We only send \( z \) after nodes locally process a batch (500-5000) and adjust the learning rate to account for this batched communication.

- We maintain a running sum average over the dual, and only compute this sum only during reduce (i.e. when \( z \) parameters arrive).

- We integrate RWDDA with MALT and compare with model averaging with SVM[3] with the RCV1 dataset, both implemented over MALT[1].