멀티프로세싱을 이용한 연합학습 시뮬레이션 가속화

유상윤[O1] 박상현[1] 문수묵[1]

[1]서울대학교 전기·정보공학부

ysy970923@snu.ac.kr, lukepark@snu.ac.kr, smoon@snu.ac.kr

Fast Federated Learning Simulator based on Multiprocessing Technique

Sangyoon Yu[O1] Sanghyoen Park[1] Soo-Mook Moon[1]

[1]Department of Electrical and Computer Engineering, Seoul National University

요 약

딥러닝 학습을 위해서는 많은 양의 데이터가 필요하다. 기존에는 기업들이 사용자들의 정보를 수집/사용해 학습이 이루어졌으나 점차 사용자의 데이터 프라이버시가 문제 되고 있다. 개인 데이터의 프라이버시를 지키면서 딥러닝 모델을 학습하는 연합학습과 같은방법들에 대해서 관심이 많아지는 추세이다. 하지만 현재 연합학습 시뮬레이션을 위한 프레임워크들이 존재하지만 다양한 방법론을 적용, 실험하는데 있어 어려움이 있는 실정이다. 이에 본 연구에서는 멀티프로세싱에 기반해 성능을 가속화한 모듈형 구조의 시뮬레이터를 제안해 연합학습 실험을 보조하고자 한다.

1. 서 론

머신러닝은 딥러닝의 등장과, 하드웨어의 발전을 통해 비약적인 발전을 거듭하고 있다. 최근에는 개인정보보호와 탈중앙 분산컴퓨팅에 대한 관심 증대로 연합 학습(FL: Federated Learning)의 개념[1]과 중요성이 대두되고 있다. 기존 머신러닝의 경우 중앙서버가 각 사용자들의 데이터를 수집해 모델을 학습하는 데 반해, 연학학습은 로컬의 모델만을 받아서 학습하기 때문에, 기존의 인공지능 학습이 가지던 데이터 프라이버시 침해문제를 해결할 수 있는 방법으로 여겨진다.

하지만 실제 연합학습을 연구/개발하기 위해서 노드만큼의 엣지 디바이스들을 통해 실험을 진행해야 한다. 실제 연합학습의 사용되는 상황을 고려했을 때 많은 노드들이 각자 적은 비율의 데이터를 가진 시나리오를 시뮬레이션하는 것이 타당하다. 하지만 비용/효율의 문제 때문에 노드별 실제 디바이스를 사용한다면 이 같은 상황을 재현하기 어렵다.

따라서 결국 하나의 컴퓨터에서 연합학습 환경을 시뮬레이션해야 한다. 이를 위해 각 노드들의 학습과 이외 필요한 연산들을 순차적으로 수행하는 방식으로 실험할 수 있으나, 낮은 성능의 엣지 노드들의 학습을 높은 성능의 컴퓨터에서 그대로 재현하는 것은 자원을 효과적으로 활용하지 못하므로 느리다. 만약 속도를 위해서 배치 크기 등의 하이퍼 파라미터를 조정할 경우 시뮬레이션하고자 하는 실제 노드의 자원을 넘어서 실험결과에 차이가 나타날 수 있다.

이에 다양한 시나리오와 방법들을 개발자가 하나의 컴퓨터에서 빠르게 적용해 볼 수 있는 시뮬레이터가 요구된다.

1) 이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. 2020R1A2B5B02001845).

2. 배경지식

2.1 연합학습

연합학습은 로컬 노드들이 서버를 통해, 또는 서버를 통하지 않고 협력하여 데이터가 탈중앙화된 상황에서 글로벌 모델을 학습하는 기술이다[1]. 이때 로컬 노드들로는 사물 인터넷 기기, 스마트폰 등의 엣지 디바이스들이 해당된다.

연합학습은 데이터 프라이버시 향상과 커뮤니케이션 효율성 측면에서 장점을 가진다. 데이터를 공유하지 않으며 학습해 차등 개인정보 보호(Differential Privacy)를 통해 데이터 프라이버시를 향상할 수 있다[2]. 데이터 프라이버시가 향상되므로 연합학습을 통해 환자들의 임상 데이터와 같은 개인정보 보호가 굉장히 중요한 분야들에서 데이터 유출 없이 학습이 가능하다[3]. 또한 기존 방식으로 서버에서 학습하기 위해서는 로컬 디바이스 데이터를 모두 서버로 전송해야 하므로 네트워크 트래픽과 스토리지 비용이 증가하는 반면, 연합학습에서는 노드에서 학습 후 로컬 모델의 일부 업데이트 정보만을 전송하므로 커뮤니케이션 비용이 상당히 줄어들게 된다.

3. 기존 연합학습 프레임워크의 문제점

실제 연합학습을 구현하기 위해 노드만큼의 디바이스가 필요하다. 하지만 실험/연구환경에서 이는 굉장히 많은 노드들을 가정하는 최근 연합학습 경향에 비추어 보았을 때 비용적 측면과 실제 통신량을 고려할 경우 효율적이지 못함을 알 수 있다. 이 때문에 많은 연합학습 실험들은 하나의 컴퓨터에서 진행됨이 일반적이다.

이를 위한 PySyft와 같은 연합학습을 위한 프레임워크들이 최근에 등장하고 있다. 하지만 효율적인 실험을 위해서는 다음과 같은 문제들을 개선해야한다.

3.1 리소스 활용

인공지능 실험 시 많은 경우에 멀티코어 CPU(8코어 이상), 멀티 GPU(2개 이상) 환경에 기반한다. 하지만 연합학습 실험의 대부분에서, 노드는 연산력이 낮은 장비로 가정해야 한다. 따라서 연산력이 높은 하나의 컴퓨터 자원 여러 노드들로 적절히 분배해야 할 필요성이 있다. 적절히 자원이 분배될 경우, 실험환경의 자원이 좋을수록 이에 비례하여 실험 속도도 향상되어야 한다.

4. 시뮬레이터 설계

연합학습의 경우 실험에 다양한 방법들과 시나리오가 적용되는 경우가 많다[4]. 시뮬레이터 설계 시 다음과 같은 상황들을 재현할 수 있어야 한다.

- 노드 별로 역할이 다른 경우:
  - 정직한 노드와 악의적인 노드(비잔틴)
  - 서버 노드와 엣지 노드
- 학습 순서, 네트워크 상황에 따른 지연 가정
- 연합학습 방법의 차이
- 네트워크 구조
  - 중앙화 네트워크(클라이언트-서버)
  - 탈중앙화 네트워크(P2P)
- 하이퍼 파라미터의 수정
  - 노드 수
  - 노드별 학습 참여 횟수

4.1 언어 및 언어 특징

파이썬(Python) 기반 개발환경에서는 딥러닝과 데이터 사이언스를 위한 라이브러리인 scikit-learn, pytorch, tensorflow 등을 쉽게 이용할 수 있다. 하지만 파이썬에는 GIL(Global Interpreter Lock)이 존재하므로 멀티쓰레드(multi-thread)를 통해 효율을 높이는 것에 한계가 있다. 따라서 멀티프로세싱(multi-processing)을 활용하는 것이 효과적이다.

Algorithm 1 main process

```
1: for round = 1, 2, ... do
2:     trainees = get this round trainees
3:     for node in trainees do
4:         if round > 1 then
5:             msg = resultQ.get()
6:             now += 1
7:             delay[msg[node]].push(msg)
8:         end if
9:         if round < nRound then
10:            trainQ.put(data needed, ...)
11:            for node in all nodes do
12:                if delay[node] time is now then
13:                    update = delay[node].pop()
14:                    doUpdate(node, update)
15:                    testQ.put(round, testModel, ...)
16:                end if
17:            end for
18:        end if
19:    end for
20: end for
```

알고리즘 1 메인 프로세스

멀티프로세싱에서는 프로세스 간의 메모리 공간이 구분되어 있다. 이 때문에 데이터 전달을 위해서 공유 메모리를 사용할 수 있지만, 모든 데이터를 공유 메모리에 저장하는 것은 경쟁상태(race condition)와 같이 여러 불안정한 요소들이 많다. 따라서 공유 메모리를 사용하는 것이 아닌, 소켓을 통해 데이터를 전달하는 것이 좋다. 이는 동기화가 제공되는 멀티프로세싱 모듈의 큐(Queue)를 사용함이 적합하다.

4.2 각 프로세스 구성

노드 개수가 변동할 수 있으므로 노드별 프로세스를 생성하는 대신, 전체 상태를 관리하는 메인 프로세스와 학습 프로세스, 성능측정 프로세스를 구분해 사용한다.

4.2.1 메인 프로세스

학습 프로세스에게 노드를 전달해 학습한다. 이때 전달하는 데이터를 통해 네트워크 토폴로지(topology)를 재현할 수 있다. 이후 학습이 완료된 노드는 다시 메인 프로세스로 전달해 적절한 시점에 상태를 업데이트한다.

실제 원하는 수준의 지연(staleness) 정도를 반영할 수 있다. 네트워크 연결 상태나 학습 시간 등 실험자가 원하는 수준을 계산해 적용하게 된다. 이로부터 기존 연합학습 도구들과는 달리, 실제 상황을 시뮬레이션하는 효과를 낼 수 있다.

Algorithm 2 train worker

```
while true do
    msg = trainQ.get()
    if trainDoneCheck(msg) then
        break
    end if
    doTrain(msg['trainData'])
    resultQ.put(data(node, data needed...))
end while
```

알고리즘 2 학습 프로세스

Algorithm 3 test worker

```
while true do
    msg = testQ.get()
    if trainDoneCheck(msg) then
        break
    end if
    if testConditionCheck(msg) then
        doTest(msg['testData'])
    end if
end while
```

알고리즘 3 성능 측정 프로세스

알고리즘 1의 doUpdate 함수 부분을 통해 실험하고자 하는 추가적인 알고리즘을 수행할 수 있다.

4.2.2 학습 프로세스

전달받은 노드를 GPU에서 학습한 후, 결과를 다시 메인 프로세스에게 전달한다. 알고리즘 2의 doTrain 함수를 통해 원하는 방식으로 학습할 수 있다.

4.2.3 성능 측정 프로세스

성능측정 프로세스에서는 전달받은 모델의 성능을 측정해 학습 진행 상황을 주기적으로 측정 및 기록한다.

5. 성능 분석

본 연구에서 제안하는 연합학습 시뮬레이터의 성능 분석을 위해 숫자, 알파벳 대소문자로 이루어진 FEMNIST 데이터셋을 작가별로 구분해 참여자 각자를 자신의 손글씨를 보관하고 있는 노드로 가정한 non-iid 상황에서 연합학습을 진행하였다[5]. 모델로는 VGG-9을 사용해 성능을 분석했다[6]. 비교를 위해 학습 방법으로 FedAvg와 지식증류를 통해 학습하는 방법을 사용했다. 후자의 경우 링(ring) 네트워크에서 P2P 통신으로 인접 2개의 노드와 통신하는 시나리오를 가정했다.

FedAvg	기존 시뮬레이터	제안된 시뮬레이터
런타임	4226s	564s
GPU 사용량	6.69%	74.75%
P2P(Ring)	기존 시뮬레이터	제안된 시뮬레이터
런타임	5338s	657s
GPU 사용량	8.32%	83.77%

표 1 성능 비교

라운드(round)는 총 100 라운드를 진행하였고, 라운드별로 20노드씩 학습을 진행했다.

시뮬레이션은 AMD Ryzen Threadripper 1950X(CPU), Nvidia 2080 super GPU 2개를 사용해 진행되었다. GPU별 6개, 총 12개 학습 프로세스를 생성해서 실험하였다.

5.1 런타임 비교

학습 프로세스가 12개이지만 데이터를 주고받는 과정과 FedAvg 등 메인 프로세스에서 처리되는 일이 병렬적으로 처리되지 못하기 때문에 약 8배 차이가 나는 것을 확인할 수 있다.

5.2. GPU 평균 사용 메모리 비교

기존의 경우 하나의 학습에 해당하는 1GB 정도의 메모리만 사용한다. 반면 제안한 시뮬레이터에서는 12개의 학습이 병렬적으로 이루어지는 것을 확인할 수 있다.

6. 결 론

본 연구에서 제안하는 시뮬레이터를 통해, 연합학습의 실제 상황을 상정한 시뮬레이션을 효율적으로 진행할 수가 있다. 특히 실험환경의 GPU 개수가 증가할수록 학습 프로세스를 병렬적으로 더 많이 처리할 수 있다. 통상 딥러닝을 학습 및 실험하고자 하는 환경에서는 멀티 GPU와 같이 좋은 자원을 사용하는 경우가 많아 본 시뮬레이터를 더 효과적으로 활용할 수 있다.

참 고 문 헌

[1] Communication-Efficient Learning of Deep Networks from Decentralized Data, Brendan McMahan. 2017
[2] Privacy-Preserving Deep Learning. Reza Shokri. 2015
[3] Federated Machine Learning: Concept and Applications, Qiang Yang.
[4] Advances and Open Problems in Federated Learning, Peter Kairouz. 2019
[5] LEAF: A Benchmark for Federated Settings, Sebastian Caldas. 2018
[6] VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION, Karen Simonyan. 2014