추천 시스템에서의 효율적인 행렬 분해 모델을 위한 정밀도 변환 기법

유재서, 고윤용, 배홍균, 강석원, 유용승, 박영준, 김상욱†
한양대학교 컴퓨터소프트웨어학과
{yjs08090, koyunyong, hongkyun, kswon0202, dydtmd1991, yongjunpark, wook}@hanyang.ac.kr

Precision Switching for Efficient Matrix Factorization in Recommender Systems

Jae-Seo Yu, Yun-Yong Ko, Hong-Kyun Bae, Seokwon Kang, Yongseung Yu, Yongjun Park, Sang-Wook Kim
Dept. of Computer Science, Hanyang University

요 약

최근 딥러닝 분야에서 모델 학습을 가속화하기 위해, 실수 표현 시 사용하는 비트 수를 줄이는 양자화 연구가 활발히 진행되고 있다. 본 논문은 추천 시스템 모델 중 하나인 행렬 분해 모델(Matrix Factorization, MF)에 대한 양자화 수행 시, 발생할 수 있는 학습 정확도 손실을 방지하기 위한 정밀도 변환 방안을 제시한다. 우리는 실세계 데이터셋을 이용한 실험을 통해, 제안 방안이 적용된 MF 모델은 양자화 기법이 적용되지 않은 모델과 비슷한 추천 정확도를 보이며, 약 30% 개선된 속도로 학습됨을 확인할 수 있었다.

1. 서론

최근 딥러닝 모델이 복잡해지고, 가용할 수 있는 데이터 양이 기하급수적으로 많아지고 있다. 이에 따라 연산량 증가폭이 큰 컴퓨터비전 분야를 중심으로 딥러닝 모델 양자화를 통한 학습 가속화 연구가 활발히 이뤄지고 있다. 모델 내 실수 표현에 일반적으로 사용하던 32 비트 부동소수점(FP32) 뿐만 아니라 16 비트 부동소수점(FP16), 나아가 8 비트 고정소수점(INT8)까지 사용하여 가속화를 하고 있다[1].

온라인 상의 클릭, 구매 기록과 같은 학습 데이터 크기 증가로 추천시스템 모델 학습 가속화에 대한 필요성이 점차 커지고 있다. 행렬 분해 모델(Matrix factorization, MF)은 대표적인 협업필터링 기반 모델이다. MF 는 유저, 아이템을 평점 패턴으로부터 추론된 임베딩 벡터들로 표현하고, 유저와 가까운 위치에 임베딩 된 아이템을 추천하는 기법이다[2]. MF 를 위한 임베딩 학습 기법으로는 일반적으로 SGD(Stochastic Gradient Descent)가 사용된다.

MF 는 전체 학습 시간 중 메모리 접근에 소요되는 시간이 크기 때문에, 파라미터 표현 비트 수를 줄이는 것이 처리량(throughput) 향상에 큰 도움이 된다[3]. 하지만 단순히 파라미터를 FP16 으로 저장하는 방법은 파라미터 업데이트 시 부동소수점 언더플로우로 인한 정확도 손실을 발생시킬 수 있다. 그림 1 은 학습이 지속됨에 따라 점차 그래디언트의 크기가 감소하는 현상을 보여준다. 이는 제한된 표현 범위를 갖는 FP16 사용으로 인한 정확도 손실을 심화시킨다.

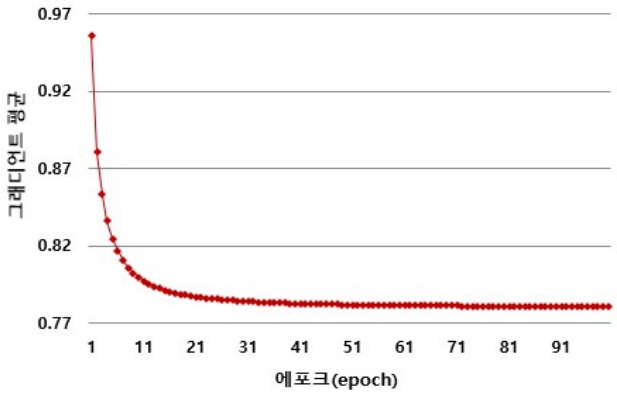

그림 1. MF epoch 별 그래디언트 평균 (MovieLens 10M)

본 논문은 학습 초반 FP16 로 파라미터를 저장하여 메모리 접근 처리량을 높이고, 학습 도중 연산과 파라미터 저장 정밀도를 FP32 로 변환하여 학습 정확도를 유지하며 학습을 가속화하는 방안을 제시한다.

† 교신저자

2. 제안 방안

기존 컴퓨터비전 분야 딥러닝 모델 양자화 기법을 MF 학습에 그대로 적용하면 가속화 효과가 크지 않을 수 있다. 기존 연구는 파라미터 업데이트의 정확성을 위해 FP32 로 파라미터를 표현하고, 행렬-행렬 곱셈 연산에 대해서는 FP16 과 같은 적은 비트 수를 사용하여 가속화를 수행한다[4]. 하지만 MF 는 memory-bound 모델로서 파라미터에 접근하는 시간이 학습 시간에 큰 비중을 차지한다.

제안 방안은 연산 뿐만 아니라 파라미터 정밀도를 줄임으로써 메모리 접근에 소요되는 시간을 줄인다. 먼저, MF 학습 초반에는 FP16 으로 파라미터 저장과 연산을 수행한다. 학습 도중 정확도가 떨어지는 epoch 지점에서 FP32 로 변환하여 저장 및 연산을 수행한다. 이 때, 변환 직전, FP32 파라미터 저장 공간을 메모리에 할당해야 한다. 이후, 기존 FP16 으로 저장된 파라미터를 새로 할당한 메모리 영역에 복사한 뒤 기존 메모리 영역을 해제하는 방식으로 수행된다.

3. 성능 평가

3.1 실험 환경

본 논문에서는 추천시스템에서 자주 사용되는 세 가지의 데이터 셋을 사용하여 실험을 수행하였다. 표 1 은 사용한 데이터 집합의 통계를 보여준다.

표 1. 데이터 집합의 통계

데이터	평점	유저	아이템	희소도
ML10M	10,000,054	71,567	10,681	98.7%
ML25M	25,000,095	162,541	59,047	99.7%
Netflix	100,480,507	480,189	17,770	98.8%

실험은 Ubuntu 18.04 운영체제에서 수행하였으며, CPU 는 Intel core i9-9900k, GPU 는 Nvidia Geforce RTX 2070 을 사용하였다. 연산장치 상세 정보는 표 2 에 나와있으며, GPU 병렬컴퓨팅 프레임워크인 CUDA(Compute Unified Device Architecture) 10.2 를 사용하였다.

표 2. 연산장치 정보

튜링(Turing)플랫폼	
중앙처리장치	인텔 옥타 코어 CPU (16 스레드), 16GB 메모리
그래픽처리장치	36 SMs, 8GB 디바이스 메모리, 448 GB/s 메모리 대역폭, SM 당: 16 쿠다(CUDA) 코어 , 2K 스레드

하이퍼파라미터의 경우 latent feature 수, learning rate, regularization, epoch 각각 128, 0.01, 0.015, 50 로 세 가지 데이터셋 모두 동일한 값을 사용하였다. 제안 방안의 정밀도 변환 지점을 찾기 위해 FP16 MF 를 사전에 학습하여 정확도가 떨어지는 epoch 을 체크한 결과 ML 10M, ML 25M, Netflix 데이터셋에 대하여 각각 31, 30, 24 의 지점을 찾을 수 있었다.

3.2 실험 결과

각 데이터셋에 대하여 세 버전(FP32 MF, FP16 MF, OURS)을 GPU 에서 실행했으며, 실험 결과는 표 3 과 같다. 표에는 각 버전의 학습 완료까지의 실행 시간(초)과 최종 RMSE 를 측정하여 기록하였다. 베이스라인은 FP32 MF 버전이며 괄호 안의 값은 베이스라인 대비 RMSE, 성능 개선 정도의 비율을 나타낸다.

학습 시간을 먼저 보면, 모든 데이터셋에서 FP16 MF 가 FP32 MF 에 비해 학습 시간이 약 53%~54% 적게 걸려 가장 빠른 속도를 보여준다. 하지만 FP16 MF 는 실수 표현 시 언더플로우로 인한 부정확한 모델 파라미터 업데이트로 최종 RMSE 정확도가 약 1.91 ~ 2.43% 감소된 것을 확인할 수 있다.

반면, 제안 방안의 경우 FP16 의 표현 범위를 벗어난 그래디언트가 많아지기 시작할 때 FP32 로 변환하여 FP32 MF 와의 정확도 차이가 최대 약 0.14% 밖에 나지 않는다. 비슷한 정확도를 유지하면서 학습에 걸리는 시간은 FP32 MF 에 비해 약 25~34% 감소했다.

표 3. 실행 시간 비교 (초)

데이터	MF 버전	학습 시간(%)	RMSE(%)
ML 10M	FP32 MF	1.26	0.7845
	FP16 MF	0.56(55%)	0.7998(-1.96%)
	OURS	0.82(34%)	0.7841(0.05%)
ML 25M	FP32 MF	3.52	0.7664
	FP16 MF	1.63(54%)	0.7810(-1.91%)
	OURS	2.38(32%)	0.7653(0.14%)
Netflix	FP32 MF	18.07	0.9176
	FP16 MF	8.57(53%)	0.9399(-2.43%)
	OURS	13.47(25%)	0.9175(0.01%)

4. 결론

본 논문은 MF 학습 중 그래디언트가 감소하는 현상을 근거로 정밀도 변환 기반 양자화 기법을 제안한다. 이는 학습 초반 파라미터의 정밀도를 줄여, memory-bound 인 MF 알고리즘을 효과적으로 가속화 하고, 이후 FP32 로 정밀도를 높여 학습 정확도를 유지한다. 추후 학습 중 정밀도 변환 지점을 자동으로 발견하여 정밀도 변환을 하는 기법을 고안하고자 한다.

사사

이 논문은 삼성전자 미래기술육성센터의 지원을 받아 수행된 연구 (과제번호 SRFC-IT1901-03)임.

참고문헌

[1] Rajagopal, A et al., Multi-Precision Policy Enforced Training (MuPPET) : A Precision-Switching Strategy for Quantised FixedPoint Training of CNNs. In ICML, pp. 7943–7952, 2020

[2] Yehuda Koren et al., Matrix factorization techniques for recommender systems. Computer, 2009.

[3] Wei Tan et al,. Matrix factorization on gpus with memory optimization and approximate computing. In ICPP, 2018.

[4] Paulius Micikevicius et al., 2018. Mixed Precision Training. In Proc. of ICLR.