맵리듀스 환경에서 프로덕트 양자화

조재오 박하명†

국민대학교 소프트웨어학부

cho5095@kookmin.ac.kr, hmpark@kookmin.ac.kr

Product Quantization on MapReduce

Jaeo Cho Ha-Myung Park

College of Computer Science, Kookmin University

요 약

프로덕트 양자화(Product Quantization)는 다차원 공간에 존재하는 벡터를 하위 차원으로 나누어 양자화하는 기법이다. 프로덕트 양자화는 근사 최근접 이웃 문제(Approximate K-Nearest Neighbor Problem)와 데이터 압축 등 여러 도메인에 활용되고 있다. 이 논문에서는 대용량 데이터에서 프로덕트 양자화를 수행할 수 있도록, 프로덕트 양자화 알고리즘을 분산 컴퓨팅 모델인 맵리듀스(MapReduce)에 적용한 분산 프로덕트 양자화 알고리즘 MRPQ(MapReduce Product Quantization)를 제안한다. MRPQ는 양자화를 위한 코드북(Codebook)을 생성하고, 코드북을 활용하여 양자화 과정을 수행한다. 실험 결과, MRPQ는 10억 개 이상의 벡터에 대해 프로덕트 양자화를 성공적으로 수행한다.

1. 서 론

최근접 이웃 문제(K-Nearest Neighbor Problem; KNN)는 다차원 공간에 존재하는 여러 점 중에서 한 점(질의 점)과 가장 가까운 K 개의 점을 찾는 문제이다. 근사 최근접 이웃 문제(Approximate K-Nearest Neighbor Problem; ANN)는 어느 정도의 오차를 허용하는 최근접 이웃 문제이며, 찾아낸 K 개의 이웃이 실제 K 개의 최근접 이웃과 되도록 많이 겹치는 것을 목표로 한다. 근사 최근접 이웃 문제는 기계 학습과 컴퓨터 비전 등의 분야에서 많이 활용되며, 데이터의 양과 차원이 기하급수적으로 증가하면서 높은 성능으로 빠르게 최근접 이웃을 찾는 문제의 중요성이 부각되었고 최근 관련 연구가 활발히 진행되었다 [1].

벡터 양자화(Vector Quantization)는 다차원 공간의 점(벡터; Vector)을 유한한 정수인 인덱스(Index)로 표현하는 방법으로, K-평균 알고리즘(K-means Algorithm) 등의 클러스터링 알고리즘으로 코드북(Codebook)을 생성하여 벡터들을 인덱스로 양자화한다. 인덱스를 활용하여 벡터 간 거리 계산을 빠르게 수행할 수 있으므로, 근사 최근접 이웃 문제에 널리 활용된다. 하지만 벡터 양자화는 높은 차원에서 정확도는 줄고 연산량은 늘어나는 현상인 '차원의 저주(Curse of Dimensionality)'에 빠지게 될 수 있다 [2] [3]. 프로덕트 양자화(Product Quantization) [3]는 벡터 양자화의 일반화된 방식으로, 벡터의 차원을 크기가 같은 여러 개의 낮은 차원으로 나누어 각각에 대해 양자화를 진행한다. 프로덕트 양자화는 높은 성능과 활용도를 보이며, 그에 따라 프로덕트 양자화를 빠르게 수행하기 위한 다양한 연구가 진행되었다 [4] [5] [6] [7].

본 연구에서는 방대한 데이터에 대해서 프로덕트 양자화를 수행하기 위한 분산 알고리즘 MRPQ(MapReduce Product Quantization)를 제안한다. MRPQ는 대용량 벡터의 처리를 목표로 프로덕트 양자화를 맵리듀스에 적용한 모델이다. 실험 결과, MRPQ는 10억 개 이상 벡터에 대해 프로덕트 양자화도 성공적으로 수행한다.

2. 관련 연구

2.1 맵리듀스(MapReduce)

맵리듀스(MapReduce) [3]는 키(Key)와 값(Value) 쌍으로 대용량 데이터를 분산 환경 기반으로 처리하는 프로그래밍 모델이다. 맵리듀스는 크게 데이터를 분할하는 맵(Map) 단계와 분할된 데이터를 같은 키로 묶어 처리하는 리듀스(Reduce) 단계로 나뉜다. 맵리듀스는 분산 환경을 기반으로 연산장치의 수에 따라 확장이 용이하다. 이러한 장점으로 빅데이터 처리에서 많이 활용된다. 이 중에서도 하둡(Hadoop)이 대표적인 프레임워크로 자리 잡고 있다.

2.2 프로덕트 양자화(Product Quantization)

프로덕트 양자화(Product Quantization) [4]는 벡터 양자화(Vector Quantization)를 일반화한 방식으로 D차원 벡터 $v \in \mathbb{R}^D$를 m개의 하위 차원으로 나누어 코드북 C로 양자화하는 방식이다. 코드북 C는 m개의 하위 코드북 C_u ($u \in \{1,2,\cdots,m\}$)의 합성곱(Cartesian Product)으로 구성된다. C_u는 k개 중심점의 집합으로 크기가 n인 D차원의 학습 벡터 집합 V를 m개의 하위 차원으로 나누어 각각에 클러스터링

†: 교신저자

본 연구는 2022년 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학사업의 연구결과로 수행되었음(2022-0-00964)

알고리즘을 수행한 결과이다. 벡터 v를 양자화 하는 것은 각 u번째 하위 벡터 v_u를 거리 계산법에 기반하여 생성한 코드북 $C_u = \{c_{u,idx} \mid idx \in \{1,2,\cdots,k\}\}$ 중 가장 가까운 중심점 $c_{u,idx}$ 의 idx로 양자화 $q_u(v_u) \in \{1,2,\cdots,k\}$하는 것을 의미한다.

프로덕트 양자화 연구는 근사 최근접 이웃 문제, 압축과 코드북 생성 등에서 진행되고 있다. 먼저 근사 최근접 이웃 문제 성능을 높이기 위해 회전(rotate)과 차원 재배열(reordering of dimensions)을 이용한 OPQ [5], LOPQ [6]가 있으며 병렬처리에 특화된 GPU를 활용하여 계산 속도를 높이는 FAISS [7]가 있다. DeltaPQ [9]는 그래프와 프로덕트 양자화를 이용한 벡터 압축 기법을 제안한다. OnlinePQ [8]는 데이터가 스트림(Stream)에서 입력될 때의 프로덕트 양자화의 코드북을 생성한다.

3. 모델 제안

MRPQ에서 프로덕트 양자화는 (1) 코드북 생성과 (2) 양자화 2가지의 과정으로 진행된다. 본 장에서는, 분산 환경 코드북 생성 기본 모델인 MRPQ$_{BASE}$를 먼저 제안하고, 하위 차원 수렴 불균형 문제를 해결한 MRPQ$_{DROP}$와 대용량 데이터 메모리 초과 문제를 해결한 MRPQ$_{HASH}$를 순서대로 제안한다. MRPQ로 생성한 코드북으로 각 벡터를 m개의 하위 벡터로 나누어 양자화 $q_u(v_u)$하며, 무차별 대입(brute force) 방식으로 진행한다.

3.1 MRPQ$_{BASE}$: MapReduce Product Quantization BASE

맵(Map) 단계
입력: 벡터 집합 $v \in V$
Step1. 이전 사이클 코드북 C^{t-1}를 읽어온다.
Step2. 벡터 v를 m개의 하위 벡터로 나눈다.
Step3. 각 하위 벡터 v_u를 하위 코드북 C_u^{t-1}에 대응
출력: $< (u,\ q_u(v_u)); v_u >$

리듀스(Reduce) 단계
입력: $< (u,\ idx); V_{u,idx} >$
Step1. $c_{u,idx}^t \leftarrow \text{avg}(V_{u,idx})$
Step2. 수렴 여부를 판단한다.
출력: $<(u,\ idx);\ c_{u,idx}^t >$

알고리즘 1 MRPQ$_{BASE}$ 의사코드

알고리즘 1은 MRPQ$_{BASE}$의 의사 코드이다. 코드북 생성 과정의 각 하위 벡터 집합에 대해서 K-평균 알고리즘을 사용하고 있다. 맵 단계에서는 이전 코드북 C^{t-1}를 읽어온다. 기존 벡터 v는 m개의 하위 벡터 $v_u(u \in \{1,2,\cdots,m\})$로 나눈다. 각 하위 벡터는 C^{t-1}를 기준으로 양자화해준다. 그 다음 키로는 u와 양자화 결과를, 값으로는 해당 하위 벡터를 넘겨준다. 리듀스 단계에서는 같은 키로 묶인 벡터의 평균으로 새로운 $c_{u,idx}^t \in C_u^t$를 생성한다. 그 다음으로 이전 코드북 C^{t-1}와 비교하여 수렴 여부를 판단한다. 단, 초기 코드북 C^0는 벡터 중에서 무작위로 k개를 뽑아 정의한다.

이 과정을 하나의 사이클(cycle)로 정의하고 코드북 C가 완전히 수렴할 때까지 반복한다. 수렴의 조건은 이전 중심점 C^{t-1}에 연결된 벡터와 새로운 중심점 C^t에 연결된 벡터의 집합이 같을 때이다.

3.2 MRPQ$_{DROP}$: MapReduce Product Quantization DROP

3.1장에서 제안한 MRPQ$_{BASE}$의 경우 특정 코드북 C_u가 수렴하더라도 불필요하게 해당 코드북 생성을 위해 다시 연산을 수행한다. MRPQ$_{DROP}$은 MRPQ$_{BASE}$와 달리 수렴한 C_u에 대해서는 맵 단계 연산을 진행하지 않음으로써 불필요한 연산을 줄인다.

3.3 MRPQ$_{HASH}$: MapReduce Product Quantization HASH

MRPQ$_{BASE}$, MRPQ$_{DROP}$의 경우 수렴 조건을 확인하기 위해 중심점에 연결된 벡터의 인덱스를 모두 저장하고 비교한다. 중심점에 대해 벡터가 골고루 분포되어 있다고 가정할 때, 각 연산장치의 메모리 사용량은 $O\left(\frac{nm}{M}\right)$ (M : 연산장치 수)으로 벡터의 개수 n에 비례해 증가한다. 하지만, 해시 코드를 사용할 경우 비트(bit)의 수를 b라고 할 때, 메모리 사용량은 연산장치 당 $O\left(\frac{mkb}{M}\right)$ 이다. 이 방법은 메모리 사용량이 고정적이며 인덱스 비교 시 처리 속도가 더 빨라진다.

4. 실험 결과

4.1 데이터셋(Datasets) 및 실험환경

데이터셋	차원(D)	데이터개수(n)	출처
SIFT1M	128	1,000,000	[10]
SIFT1B	128	1,000,000,000	

표 1 데이터셋

데이터셋은 SIFT1M과 SIFT1B를 사용한다. SIFT1M은 128차원이며 100만 개 벡터로 구성된다. SIFT1B는 같은 차원에 10억 개의 벡터로 구성된다. 본 실험은 5대의 연산장치로 구성된 클러스터 서버에서 진행한다. 각 연산장치는 Intel(R) Xeon(R) E-2224 CPU @ 3.40GHz 4코어의 CPU로 구성되어 있다.

4.2 코드 길이(code length)에 따른 평균 제곱 오차(MSE)

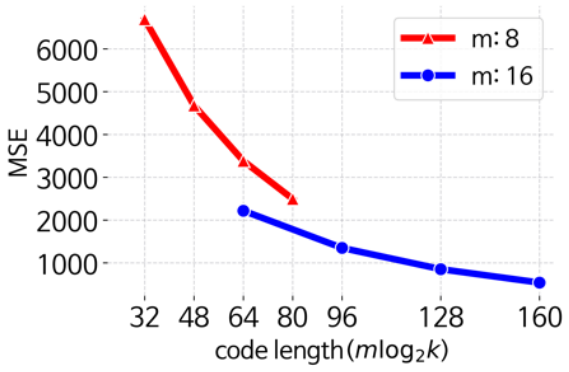

그림 1 코드길이에 따른 평균 제곱 오차(MSE)

그림 1은 코드 길이(code length; $m\log_2 k$)와 평균 제곱 오차(Mean Squared Error)의 관계를 보여준다. m과 k가

클수록 평균 제곱 오차의 값이 감소한다. 이는 벡터를 프로덕트 양자화할 때 왜곡(distortion)이 감소함을 의미한다.

4.3 코드북 생성 시간

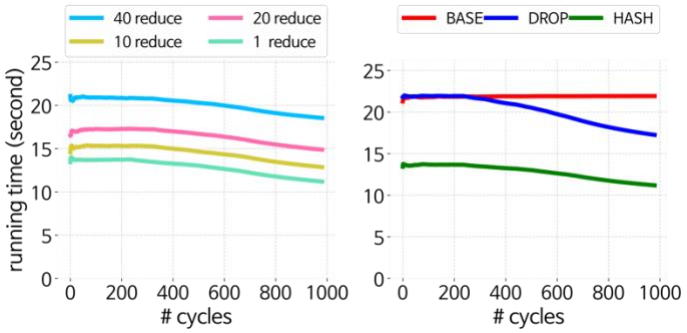

그림 2 평균 시간 비교

m을 8, k를 256으로 설정하여 SIFT1M 데이터를 처리했을 때, 그림 2의 좌측 그림은 서로 다른 리듀스 태스크(reduce task)에 대한 평균 시간(특정 사이클까지의 연산 시간에 해당 사이클의 수를 나눈 값)이다. 리듀스 태스크가 많을수록 평균시간이 증가함을 알 수 있다. 이는 코드북을 태스크 수에 따라 생성된 파일 개수가 많아져 파일 입출력이 증가하기 때문이다. 그림 2의 우측 그림은 $MRPQ_{BASE}$, $MRPQ_{DROP}$, $MRPQ_{HASH}$ 간의 평균 시간을 비교한다. $MRPQ_{BASE}$, $MRPQ_{DROP}$은 초반에 둘 차이가 없지만 수렴한 하위 코드북의 개수가 발생하고 불필요한 연산이 줄어들면서 시간이 감소하는 것을 확인할 수 있다. 또한, $MRPQ_{HASH}$의 경우 전체적으로 시간이 감소함을 확인할 수 있다.

4.4 SIFT1B 처리 능력

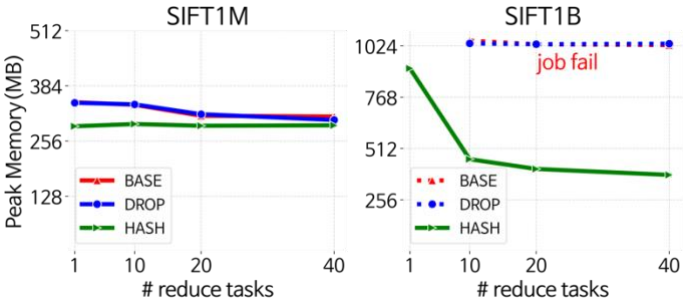

그림 3 한 사이클에서 리듀스 태스크의 최대 메모리 사용량

그림 3은 m이 8이고 k가 256일 때, SIFT1M과 SIFT1B 데이터에 대해 리듀스 태스크의 최대 메모리 사용량을 보여준다. SIFT1M에서는 데이터가 작아 $MRPQ_{BASE}$, $MRPQ_{DROP}$과 $MRPQ_{HASH}$의 메모리 사용량 차이가 크지 않다. 하지만 SIFT1B에서는 2배 이상의 차이를 보이며 $MRPQ_{BASE}$, $MRPQ_{DROP}$ 일부에서는 메모리 초과로 연산이 불가했다. 리듀스 태스크가 1일 때는 모든 태스크가 실패했다. 리듀스 태스크가 1일 때를 제외하면 일부는 성공했지만 다른 태스크는 메모리 초과로 작업이 실패했다.

5. 결론

이 연구에서는 프로덕트 양자화를 맵리듀스에 적용하여 대용량 데이터에 대해서 K-평균 알고리즘 기반으로 코드북을 생성하고 맵리듀스에 적용한 모델 MRPQ를 제안했다. MRPQ는 실험결과에서 10억 개 이상의 벡터로 구성된 데이터가 주어졌을 때 코드북 생성과 프로덕트 양자화를 성공한다.

참 고 문 헌

[1] M. Aümuller, E. Bernhardsson and A. Faithfull, ANN-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms, pp. 34--49, 2017.

[2] C. Böhm, S. Berchtold and D. A. Keim, “Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases,” *ACM Computing Surveys (CSUR),* vol. 33, pp. 322--373, 2001.

[3] J. Dean and S. Ghemawat, “MapReduce: simplified data processing on large clusters,” *Communications of the ACM,* vol. 51, pp. 107--113, 2008.

[4] H. Jégou, M. Douze and C. Schmid, “Product quantization for nearest neighbor search,” *IEEE transactions on pattern analysis and machine intelligence,* vol. 33, pp. 117--128, 2010.

[5] T. Ge, K. He, Q. Ke and J. Sun, “Optimized product quantization,” *IEEE transactions on pattern analysis and machine intelligence,* vol. 36, pp. 744--755, 2013.

[6] Y. Kalantidis and Y. Avrithis, “Locally optimized product quantization for approximate nearest neighbor search,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 2321--2328, 2014.

[7] J. Johnson, M. Douze and H. Jégou, “Billion-scale similarity search with GPUs,” *IEEE Transactions on Big Data,* vol. 7, pp. 535--547, 2019.

[8] D. Xu, I. W. Tsang and Y. Zhang, “Online product quantization,” *IEEE Transactions on Knowledge and Data Engineering,* vol. 30, pp. 2185--2198, 2018.

[9] R. Wang and D. Deng, “DeltaPQ: lossless product quantization code compression for high dimensional similarity search,” *Proceedings of the VLDB Endowment,* vol. 13, pp. 3603--3616, 2020.

[10] [Online]. Available: http://corpus-texmex.irisa.fr/.