아파치 스파크에서 모바일 빅 데이터에 대한 다계층 인공신경망 기반 분산 딥러닝 구현 및 최적화1)

명노영† · 안범진† · 유헌창†
† 고려대학교 컴퓨터학과

Implementation and Optimization of Distributed Deep learning based on Multi Layer Neural Network for Mobile Big Data at Apache Spark

Rohyoung Myung† · Beomjin Ahn† · Heonchang Yu†
† Dept. of Computer Science, Korea University

요 약

빅 데이터의 시대가 도래하면서 이전보다 데이터로부터 유의미한 정보를 추출하는 것에 대한 연구가 활발하게 진행되고 있다. 딥러닝은 텍스트, 이미지, 동영상 등 다양한 데이터에 대한 학습을 가능하게 할 뿐만 아니라 높은 학습 정확도를 보임으로써 차세대 머신러닝 기술로 각광 받고 있다. 그러나 딥러닝은 일반적으로 학습해야하는 데이터가 많을 뿐만 아니라 학습에 요구되는 시간이 매우 길다. 또한 데이터의 전처리 수준과 학습 모델 튜닝에 의해 학습정확도가 크게 영향을 받기 때문에 활용이 어렵다. 딥러닝에서 학습에 요구되는 데이터의 양과 연산량이 많아지면서 분산 처리 프레임워크 기반 분산 학습을 통해 학습 정확도는 유지하면서 학습시간을 단축시키는 사례가 많아지고 있다. 본 연구에서는 범용 분산 처리 프레임워크인 아파치 스파크에서 데이터 병렬화 기반 분산 학습 모델을 활용하여 모바일 빅 데이터 분석을 위한 딥러닝을 구현한다. 딥러닝을 구현할 때 분산학습을 통해 학습 속도를 높이면서도 학습 정확도를 높이기 위한 모델 튜닝 방법을 연구한다. 또한 스파크의 분산 병렬처리 효율을 최대한 끌어올리기 위해 파티션 병렬 최적화 기법을 적용하여 딥러닝의 학습속도를 향상시킨다.

1. 서 론

모바일 기기, 센서 네트워크[1], 사물 인터넷[2]의 발달로 수집되는 데이터의 양이 폭발적으로 증가하고 있다. 다양한 머신러닝 기법들을 통해 대량의 데이터로부터 다양한 정보를 추출할 수 있게 되면서 데이터의 수집 및 관리의 중요성이 부각되고 있다. 더불어 데이터를 처리할 수 있는 분산처리 환경, GPU를 활용한 처리 환경이 출현하면서 해당 환경에서 빅 데이터에 최신 머신러닝 기술을 적용하는 사례가 증가하고 있다.

머신러닝 기술들은 데이터들을 각 기술에 알맞게 전처리하여 적용될 경우 다양한 분야에서 활용될 수 있다. 특히 최근 각광받는 딥러닝[3,4,5]는 다양한 입력데이터: 텍스트, 이미지, 스트리밍 데이터에 대해 기존과는 다른 특색 있는 결과물을 만들 수 있다. 딥러닝은 텍스트에 대한 단순한 회귀, 분류뿐만 아니라 이미지를 학습하여 이미지에 부합하는 캡션을 생성하거나 음성을 학습하여 음성 인식 정확도를 급격하게 높일 수 있다.

딥러닝을 통해 빅 데이터를 학습할 경우 처리해야 할 연산량이 많아지고 이는 학습시간의 급격한 증가로 이어진다. 특히 딥러닝의 경우 각각의 연산 계층에서 다수의 뉴런을 사용할 뿐만 아니라 다수의 연산 계층을 사용하여 데이터를 학습하기 때문에 연산량이 많다. 이를 해결하기 위해 범용적인 분산처리 프레임워크[6,7,8]에 분산 학습 모델[9]를 적용하여 학습시간을 단축시키거나 GPU 같은 특수목적형 연산장치를 사용해서 학습시간을 단축시킨다[10].

본 논문에서는 대표적인 범용 분산 처리 프레임워크인 아파치 스파크[6]에서 분산 학습 기반 딥러닝을 구현하여 모바일 빅 데이터를 분류한다. 이때 모바일 빅 데이터[11]의 특징을 고려하여 딥러닝 모델에 알맞게

1) "본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 SW중심대학지원사업의 연구결과로 수행되었음"(2015-0-00936)

전처리하고 반복 시행을 거쳐 학습정확도를 최대화 할 수 있도록 모델을 튜닝한다. 또한 스파크의 분산 학습 모델을 분산 처리 모델에 적용 할 때 처리 속도를 향상시키기 위해 파티션 병렬 최적화 기법을 적용한다. 마지막으로 모델 튜닝을 통한 학습 정확도의 변화와 병렬 최적화 기법을 적용 했을 때의 성능 향상치를 제시한다.

2. 이론적 배경

2.1 다계층 인공신경망

다계층 인공신경망[12]는 퍼셉트론으로 해결하지 못하는 학습 문제를 해결하기 위해 제안된 머신러닝 기법이다. 다계층 인공신경망은 입력계층, 단수 혹은 다수의 은닉계층 그리고 출력계층으로 구성된다. 입력계층은 입력 데이터의 벡터의 차원수와 동일한 뉴런으로 구성되고, 은닉계층은 다수의 뉴런, 출력계층은 사용자의 학습 목적성에 맞는 개수의 뉴런으로 구성된다.

학습과정은 반복적인 전방 전달과 역전파 과정으로 나뉜다. 전방 전달과정에서 데이터는 입력계층에서 선형변환, 은닉계층에서 활성화 함수(예: 시그모이드, ReLU 등)을 거쳐 최종적으로 출력계층에서 소프트맥스 함수를 통한 정규화 과정을 통해 최종적인 값을 출력한다. 역전파 과정에서는 전방 전달과 반대방향으로 연산이 진행되며 정의된 손실함수(예: Mean Squared Error, Cross-Entropy 등)의 값을 최소화시키는 방법(예: Stochastic Gradient Descent 등)을 통해 각 계층의 뉴런들의 가중치 값을 수정한다. 이러한 학습과정을 거쳐서 완성된 인공신경망은 이후에 입력되는 데이터에 대한 분류를 하는데 사용된다.

2.2 분산 학습 모델

분산 학습 모델[9]는 다수의 연산 장치에서 학습 모델을 분산 시켜서 처리하기 위한 모델로서 데이터 병렬화 기법과 모델 병렬화 기법으로 나눠진다. 데이터 병렬화 기법은 학습에 사용되는 데이터를 분할해서 각각의 연산 장치에서 병렬로 학습한 후 모델 파라미터를 병합하여 최종 학습 모델을 생성한다. 모델 병렬화 기법은 인공 신경망을 분할하여 전방 전달의 결과물과 역전파 과정 중 각 계층의 가중치 갱신값을 교환하는 방법이다. 모델 병렬화기법의 경우 모델의 크기가 메모리보다 클 경우 사용된다. 그러나 한 번의 전방 전달과 역전파마다 연산장치들에서 각 계층의 출력값과 가중치 갱신값을 동기화해야하기 때문에 네트워크 부하가 발생하는 단점을 갖는다. 이에 반해 데이터 병렬화 기법은 배치 크기에 따라 다수의 전방 전달과정과 역전파 과정에서 수집된 결과물을 한 번에 수집하여 모델에 적용가능하기 때문에 동기화로 인한 부하가 적게 발생한다.

2.3 아파치 스파크

아파치 스파크[6]은 범용성을 지닌 다목적용 분산처리 프레임워크로써 인메모리 컴퓨팅, lazy execution, 선택적인 데이터 캐싱을 통해 기존 분산 처리 프레임워크에 비해 크게 처리 성능을 향상시켰다. 또한 고유한 자료구조인 Resilient Distributed Datasets(RDD)를 사용한 Directed Acyclic Graph(DAG)기반 병렬처리를 통해 내결함성을 보장한다.

3. 모델 튜닝

본 연구에서 사용한 데이터는 모바일 빅 데이터인 [11]으로써 가속도 센서로부터 x, y, z축 가속도 값과 라벨(해당 상황에서의 사용자의 동작상태), timestamp 값을 수집한 데이터이다. 해당 데이터를 전처리 과정 없이 다계층 인공신경망에 학습시킬 경우 같은 가속도 값을 갖지만 다른 라벨을 갖는 경우가 빈번하기 때문에 학습정확도가 매우 낮아지는 문제가 발생한다. 이를 해결하기 위해 해당 데이터는 연속성을 갖는다는 점을 이용해서 다수의 연속적인 데이터를 스펙트로그램으로 변환하는 전처리를 수행했다.

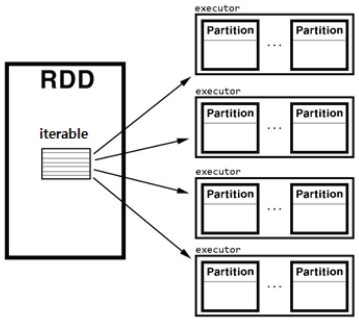
모델의 학습 정확도 향상을 위해 은닉계층의 개수[1,2], 은닉계층별 뉴런의 개수[500,1000], 학습률[0.1,0.5], 학습 반복 횟수[1,200]의 조합을 선형적으로 증가시킬 때 분류 정확도의 변화를 관찰하며 모델을 조율했다. 이때 가장 높은 학습정확도는 [13]을 기준으로 조율을 수행했다. 일반적으로 딥러닝은 은닉계층을 구성하는 뉴런의 개수가 많아질수록, 그리고 은닉계층이 많아질수록 분류정확도가 높아진다는 통념이 있으나 은닉계층이 많아질 때 활성함수로 시그모이드를 사용할 경우 'vanishing gradient' 현상[14]가 발생해서 오히려 분류 정확도가 낮아지는 결과를 초래했다. 은닉계층별 뉴런의 개수를 선형적으로 증가시켰을 때에도 일정 개수 이상이 되면 분류 정확도가 감소한다. [11]의 경우 학습률은 0.5에서 가장 빨리 높은 분류 정확도에 도달하며 바로 수렴하는 추세를 보였다. 마지막으로 반복횟수는 일정 횟수가 되면 한동안 수렴하지만 그 이후에는 다시 감소하는 경향을 보였다. 정확한 튜닝 결과는 실험 결과 부분에서 다룰 것이다.

4. 파티션 병렬 최적화

아파치 스파크[6]은 [그림 1]과 같이 RDD를 다수의 파티션으로 분할하여 연산 장치인 엑세큐터들에서 병렬로 처리한다. RDD는 반복처리가 가능한 자료구조를 포함하는 추상 자료구조다. 실제 RDD를 처리할 때는 아파치 스파크의 기존 스케줄링 정책(FIFO)에 따라 반복처리가 가능한 자료구조를 동일한 크기의 파티션[15]들로 분할하고 엑세큐터들에게 할당해서 처리한다.

이때 모든 엑세큐터들에게 할당된 작업량이 동일하

지 않을 경우 작업 완료 시간이 가장 늦게 작업을 완료하는 엑세큐터로 수렴하기 때문에 성능이 저하되는 경우가 발생한다.



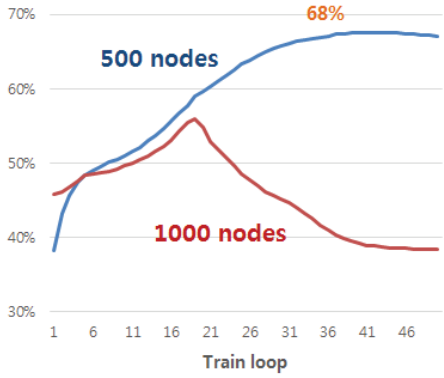
[그림 1] 아파치 스파크에서 RDD의 분산 처리 과정

본 연구에서는 이러한 성능저하를 방지하기 위해 파티션 병렬 최적화 과정을 수행한다. 파티션 병렬 최적화는 시스템의 연산처리 자원에게 동일한 작업량을 할당해서 엑세큐터들의 작업 완료시간을 최대한 동일하게 만드는 기법이다. 이를 위해 RDD를 파티션들로 분할할 때 두 가지 척도를 고려해서 분할한다. 첫 번째로 RDD를 분할할 때 엑세큐터의 병렬처리 최소단위인 코어의 개수들을 고려해서 분할해야 한다. 각각의 엑세큐터는 멀티코어 CPU를 사용할 경우 다수의 코어를 갖게 되고 할당받은 파티션들을 각각의 코어에서 병렬로 처리하기 때문에 엑세큐터들로 할당할 파티션 총 개수는 모든 엑세큐터들의 코어 개수의 배수가 돼야 한다. 두 번째로 모든 파티션들이 최대한 동일한 크기를 가질 수 있도록 RDD를 분할해야 한다. 이를 위해 RDD를 모든 엑세큐터들의 코어 개수로 분할할 때 파티션들의 크기를 비교해서 최대한 각 파티션들의 크기가 동일해 지도록 분할해야 한다. 그러나 파티션들의 개수가 너무 많아질 경우 파티션들의 처리에 대한 스케줄링 오버헤드가 커지기 때문에 반복적인 시행착오를 거쳐 최적의 파티션 개수를 찾아야 한다.

5. 실험 결과

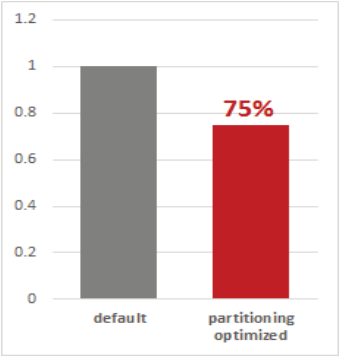
실험은 총 4대의 노드로 구성된 아파치 스파크 클러스터에서 수행됐다. 각각의 노드는 Ubuntu 14.04 LTS, Intel Core i7(8 cores, 3.40GHz), 16GB 메모리, 256GB SSD, 1Gbps 네트워크의 스펙을 갖는다. 아파치 스파크는 버전 2.0.2버전을 사용했고, 클러스터 매니저로는 기본 클러스터 매니저인 Stand Alone 매니저를 사용했다.

모델 튜닝은 3장에서 언급했던 것처럼 튜닝할 파라미터들의 조합을 선형적으로 증가시키면서 반복 수행한 결과 중에 가장 높은 분류 정확도를 가졌던 두 경우를 [그림 2]에서 표현했다. 가로축은 학습의 반복횟수를 의미하고 세로축은 분류 정확도를 의미하며 학습률은 0.5를 사용했다.



[그림 2] 모델 튜닝을 수행했을 때 학습의 반복 횟수에 따른 분류 정확도 변화

은닉층에 포함되는 뉴런의 개수를 500개로 했을 때 [14]에서 언급됐던 가장 높은 분류 정확도인 68%를 달성할 수 있었고, 달성 후에 학습의 반복 횟수가 증가하더라도 해당 정확도에 수렴하는 모습을 관측할 수 있었다. 그러나 뉴런의 개수를 1000개로 했을 때는 학습의 반복 횟수가 19회가 될 때까지는 분류 정확도가 증가했으나 이후로는 급격하게 감소해서 최종적으로 38%에 수렴하는 결과를 관측할 수 있었다.



[그림 3] 파티셔닝 최적화 기법을 적용했을 때 처리시간 변화

기존 처리시간과 파티셔닝 최적화 기법을 적용했을 때 처리시간의 변화는 [그림 3]과 같다. 처리 시간 변화를 측정하기 위해 데이터를 모든 엑세큐터들의 코어 개수로 파티셔닝했고, 이후 처리시간의 변곡점이 관측되기까지 반복적으로 파티션 개수를 배수로 증가시켰다. 5번동안 각각의 경우를 반복적으로 실험했다. 이에 대한 평균치중 가장 처리시간 변화가 컸던 경우는 모든 코어개수의 6배수일 때였고, 감소폭은 25%였다.

6. 결론 및 논의

본 연구는 범용성을 갖는 분산 처리 프레임워크인 아파치 스파크에서 분산 학습 기반 다계층 신경망을 구현하여 모바일 빅 데이터를 학습시켰다. 또한 기존의 학습 정확도 최고치를 이루기 위한 모델 튜닝 과정을 구체적으로 다뤘고, 아파치 스파크의 작업 처리 특징을 고려해서 처리시간을 최적화할 수 있도록 파티션 병렬 최적화 기법을 적용하여 처리시간을 25% 감소시켰다. 추후 연구에서는 특수 목적형 연산 처리 장치인 GPU를 활용하여 분산 학습을 구현하여 범용 분산 처리 프레임워크와 성능 및 차이점을 분석할 것이다. 또한 모바일 빅 데이터의 연속성을 고려하여 분류 정확도를 높이기 위해 Recurrent Neural Network를 사용하여 데이터를 학습시킬 예정이다.

참 고 문 헌

[1] Yick, J., Mukherjee, B., & Ghosal, D. (2008). Wireless sensor network survey. Computer networks, 52(12), 2292-2330.

[2] Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future generation computer systems, 29(7), 1645-1660.

[3] Hinton, G. E. (2009). Deep belief networks. Scholarpedia, 4(5), 5947.

[4] Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y. (2009, June). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In Proceedings of the 26th annual international conference on machine learning (pp. 609-616). ACM.

[5] Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., & Khudanpur, S. (2010, September). Recurrent neural network based language model. In Interspeech (Vol. 2, p. 3).

[6] Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., ... & Ghodsi, A. (2016). Apache Spark: A unified engine for big data processing. Communications of the ACM, 59(11), 56-65.

[7] Isard, M., Budiu, M., Yu, Y., Birrell, A., & Fetterly, D. (2007, March). Dryad: distributed data-parallel programs from sequential building blocks. In ACM SIGOPS operating systems review (Vol. 41, No. 3, pp. 59-72). ACM.

[8] Carbone, P., Katsifodimos, A., Ewen, S., Markl, V., Haridi, S., & Tzoumas, K. (2015). Apache flink: Stream and batch processing in a single engine. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 36(4).

[9] Hillis, W. D., & Steele Jr, G. L. (1986). Data parallel algorithms. Communications of the ACM, 29(12), 1170-1183.

[10] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... & Ghemawat, S. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467.

[11] Kwapisz, J. R., Weiss, G. M., & Moore, S. A. (2011). Activity recognition using cell phone accelerometers. ACM SigKDD Explorations Newsletter, 12(2), 74-82.

[12] Yegnanarayana, B. (2009). Artificial neural networks. PHI Learning Pvt. Ltd..

[13] Alsheikh, M. A., Niyato, D., Lin, S., Tan, H. P., & Han, Z. (2016). Mobile big data analytics using deep learning and apache spark. IEEE network, 30(3), 22-29.

[14] Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks, 5(2), 157-166.

[15] Apache Spark: Spark Configuration. http://spark.apache.org/docs/latest/configuration.html