ISSN 1229-1889(Print)
ISSN 2287-9005(Online)
J Korean Soc Qual Manag Vol. 48, No.4: 553-566, December 2020
https://dx.doi.org/10.7469/JKSQM.2020.48.4.553

빅데이터를 위한 H-RTGL 기반 단일 분류기 분산 처리 프레임워크 설계

김도균* · 최진영**†

* 아주대학교 공학연구소 · ** 아주대학교 산업공학과

Design of Distributed Processing Framework Based on H-RTGL One-class Classifier for Big Data

Do Gyun Kim* · Jin Young Choi**†

* Engineering Research Institute, Ajou University
** Department of Industrial Engineering, Ajou University

ABSTRACT

Purpose: The purpose of this study was to design a framework for generating one-class classification algorithm based on Hyper-Rectangle(H-RTGL) in a distributed environment connected by network.

Methods: At first, we devised one-class classifier based on H-RTGL which can be performed by distributed computing nodes considering model and data parallelism. Then, we also designed facilitating components for execution of distributed processing. In the end, we validate both effectiveness and efficiency of the classifier obtained from the proposed framework by a numerical experiment using data set obtained from UCI machine learning repository.

Results: We designed distributed processing framework capable of one-class classification based on H-RTGL in distributed environment consisting of physically separated computing nodes. It includes components for implementation of model and data parallelism, which enables distributed generation of classifier. From a numerical experiment, we could observe that there was no significant change of classification performance assessed by statistical test and elapsed time was reduced due to application of distributed processing in dataset with considerable size.

Conclusion: Based on such result, we can conclude that application of distributed processing for generating classifier can preserve classification performance and it can improve the efficiency of classification algorithms. In addition, we suggested an idea for future research directions of this paper as well as limitation of our work.

Key Words: Distributed Machine Learning, H-RTGL, One-Class Classification, Model/Data Parallelism, Big data

● Received 30 October 2020, 1st revised 16 November 2020, accepted 7 December 2020
† Corresponding Author(choijy@ajou.ac.kr)

※ This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2017R1A2B4009841)

1. 서 론

기술의 발달로 인해 다양한 분야에서 발생하는 데이터의 양이 폭발적으로 증가하고 있으며, 이러한 데이터 크기의 증가속도는 기존의 데이터 처리 및 분석 방법의 성능 개선 속도를 상회한다. 예를 들어, 반도체 제조 공정은 산화(oxidation), 식각(etching), 박막 공정 등을 포함한 8개의 주요 공정으로 구분되지만, 실질적인 세부 공정 단계의 수는 500-600개에 달하며, 이에 따라 발생하는 공정 데이터의 양도 매우 방대하여 분석이 쉽지 않다(Chao, 2001). 이러한 빅데이터를 가공하여 분석을 수행하기 위한 대안으로서 분산 처리(distributed processing)가 고려되고 있다. 분산 처리는 네트워크를 통해 연결되어 있으며, 복수의 컴퓨팅 노드로 구성된 분산 환경(distributed environment)에서 여러 개의 노드가 동시에 데이터를 다루고 분석하는 방법이라고 할 수 있다. 이를 통해 물리적인 제약을 극복할 수 있을 뿐 아니라, 복수의 컴퓨팅 자원을 이용하여 더욱 효율적으로 데이터를 처리하고 분석하는 것이 가능하다.

한편, 이와 같은 분산 처리를 바탕으로 공정에 대한 모니터링 및 공정과 제품의 품질 관리에 매우 유용한 기계학습(machine learning)의 방법론들을 더욱 효율적으로 수행할 수 있다. Figure 1과 같이 수행되는 분산 기계 학습은 훈련을 위한 모델을 포함하고 있는 복수의 분산된 노드가 병렬적으로 데이터를 받아 분석을 수행하며, 각각의 노드로부터 수행된 분석 결과를 취합하여 최종적으로 하나의 모델을 생성하는 상위 노드가 존재한다. 올바른 분산 기계 학습의 수행을 위해서는 적절한 기반 환경이 구축될 필요가 있으며, 이러한 기반 환경은 분산 처리의 대상이 되는 학습 데이터나 모델을 교환하고 각각의 분산된 주체에서 수행된 연산의 결과를 취합하는 기능들을 수행하기 위한 지원 도구들을 포함하여 구성될 필요가 있다.

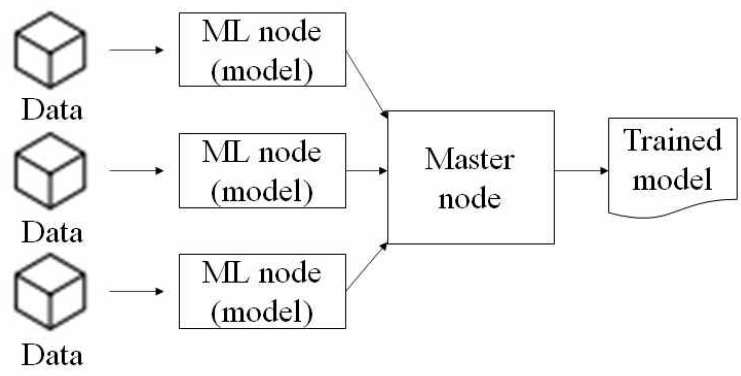

Figure 1. Concept of distributed machine learning

이러한 배경 하에서 본 논문은 높은 수준의 분류 정확도를 제공할 수 있을 뿐 아니라, 분류 요인에 대한 용이한 해석을 제공할 수 있어 사후 분석에 유용한 Hyper-Rectangle (H-RTGL) 기반의 기하학적 단일 분류(One-Class Classification, OCC) 알고리즘을 분산 환경에서 수행하기 위한 분산 처리 프레임워크를 제안하고자 한다. 보다 구체적으로는 복수의 컴퓨팅 노드에서 단일 분류기 생성을 수행하기 위한 분산 H-RTGL 기반 단일 분류 알고리즘을 모델 및 데이터의 분산화(parallelism)를 고려하여 설계한 후, 이를 수행하고 보조하기 위한 지원 환경을 설계한다. 또한, 지원 환경은 알고리즘 수행 과정에서 성능 최적화, 데이터 교환 및 분석 등을 수행하기 위한 컴포넌트들을 포함한다. 본 연구의 결과로서 설계된 분산 처리 프레임워크는 다양한 알고리즘에 대한 분산 기계 학습을 수행하기 위한

토대로 사용될 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 분산 처리를 위한 핵심 인프라에 해당하는 빅데이터 처리 및 분석을 위한 플랫폼 및 분산 기계 학습에 대한 문헌 조사 결과를 소개한다. 3장에서는 분산 처리의 대상이 되는 기반 방법론 및 분산 프레임워크를 구성하는 컴포넌트의 상세한 설계 내용을 기술한다. 이후, 4장에서는 UCI machine learning repository의 데이터 집합들에 대해 분산 기계 학습을 수행하고, 이를 분산 처리를 적용하지 않은 학습 결과와 비교함으로써 본 논문이 제안하는 프레임워크를 통해 생성된 분류기의 성능을 검증한다. 마지막으로 5장에서는 본 논문의 의의 및 결론을 제시하고, 추후 연구를 위한 방향을 제안한다.

2. 관련 연구 조사

2.1 빅데이터 처리 및 분석을 위한 플랫폼 연구 조사

빅데이터 처리 및 분석을 위한 플랫폼은 분산 환경에서의 기계 학습을 수행하기 위한 가장 중요한 기반 기술이라고 할 수 있다. 분산 환경에서의 기계 학습을 올바르게 하기 위해서는 데이터를 처리하거나 분석을 수행하는 기존의 데이터 분석 솔루션의 기능과 더불어 대규모의 빅데이터를 저장하고 관리하기 위한 컴퓨팅 인프라가 요구된다(Bekkerman et al., 2011). 가장 대표적인 사례인 Apache 사의 Hadoop (High-Availability Distributed Object-Oriented Platform)은 분산 파일 시스템인 Hadoop Distributed File System (HDFS)를 바탕으로 분산된 환경에서 데이터 처리를 수행하고, 이에 따른 결과를 취합하기 위한 MapReduce를 포함한다(Padhy, 2013). MapReduce 프레임워크에 포함되는 Mahout 라이브러리는 다양한 기계 학습 알고리즘을 제공하며, 제공되는 알고리즘의 내역은 Table 1.과 같다(Giacomelli, 2013).

Table 1. List of algorithms provided by Mahout library

Category	Detailed algorithms
Collaborative filtering (CF)	Item-based, Matrix Factorization
Classification	Naïve Bayes, Complementary Naïve Bayes, Random Forest
Clustering	K-means Clustering, Spectral Clustering, Canopy Clustering, Fuzzy k-means, Streaming k-means clustering
Dimensionality reduction	Lanczos Algorithm, Stochastic Singular Value Decomposition, Principal Component Analysis
Topic model	Latent Dirichlet Allocation
Miscellaneous	Frequent Pattern Matching, RowSimilarity Job, ConcatMatrices, Collocations

Hadoop은 HDFS를 기반으로 큰 데이터를 용이하게 처리할 수 있으며, 이를 바탕으로 분산 환경에서 복잡한 데이터 분석을 수행할 수 있다는 장점을 가진다. 그러나 물리적 디스크 작업을 수행하는 MapReduce의 특징 때문에 데이터 용량에 대한 물리적 상한선이 존재하며, 작업 용량을 할당하는 과정에서도 디스크 기반의 데이터 공유로 인한 비효율성을 가진다(Parsian, 2015). Spark는 이러한 Hadoop MapReduce의 단점을 극복하기 위해 개발되었으며, 디

스크가 아닌 메모리 기반으로 빅데이터를 처리하고 분석하기 위한 플랫폼이다(Zaharia et al., 2010). 다양한 형태로 데이터를 처리할 수 있으며, Spark 내에서 Hadoop 생태계의 다른 요소들이 수행하는 쿼리 작업, 기계 학습 알고리즘 제공 등의 역할을 수행하는 라이브러리들을 추출하여 이용할 수도 있다. Spark의 MLib는 기계 학습 알고리즘을 제공하기 위한 라이브러리로서 Hadoop의 Mahout과 같은 역할을 수행한다. 그 외에도 Spark는 다양한 솔루션과의 연계가 자유로운데, 딥러닝 수행을 위한 프레임워크인 Caffe와의 연동을 통해 분산 환경에서의 딥러닝을 제공하는 DeepSpark, SparkNet 등이 이에 대한 대표적인 예시라고 할 수 있다(Kim et al., 2016; Moritz et al., 2015). Hadoop이나 Spark와 같이, 범용적인 용도로 사용되는 분산 처리 시스템이 아닌, 기계 학습에 특화된 분산 처리 프레임워크 또한 존재한다. Petuum은 대표적인 오픈 소스 분산 기계 학습 프레임워크이며, 기계 학습을 위한 라이브러리와 데이터 분산화(data parallelism)를 담당하는 Bosen, 모델 분산화(model parallelism)를 담당하는 Strasds를 포함하고 있다(Xing et al., 2015). Bosen은 제한적인 동기화(synchronization) 모델을 기반으로 데이터를 병렬적으로 처리하는 과정에서 우수한 성능을 보이며, Strads는 보다 효율적인 연산 순서를 부여하고 모델의 매개 변수(parameter)를 세부적으로 업데이트 하는 등 분산 처리를 위해 분리된 모델들의 효율적인 스케줄링(scheduling)을 수행한다. 그 외에도, Petuum은 sparse coding과 같이 복잡한 기계 학습 및 딥러닝 알고리즘을 포함한 기계 학습 라이브러리를 제공하며, Hadoop과 호환이 용이하여 확장성과 유연성 측면에서도 장점을 가지고 있다(Peralta et al., 2017). Baek et al. (2018)은 이와 같이 분산 환경에서 기계 학습을 위해 사용될 수 있는 다양한 인공지능 기반 서비스를 제공하는 플랫폼들을 편의성(convenience)과 안전성(safety) 등의 측면에서 평가하기 위한 세부 측정 항목을 제안하였다.

2.2 분산 환경에서의 기계 학습 연구 조사

분산 환경에서의 기계 학습에 대한 연구들은 다음과 같다. Chang (2011)은 대표적인 기계 학습 알고리즘인 Support Vector Machine (SVM)을 분산 환경에서 수행하기 위한 Parallel SVM (PSVM)을 제안하였다. PSVM은 계산 과정에서 발생하는 Quadratic Programming (QP) 문제를 풀기 위해 학습 데이터를 복수의 분산된 노드에 할당하고, 각각의 노드는 내부점 접근(Interior Point Method, IPM) 방법을 통해 해를 찾아 계산 결과를 취합하는 방식으로 분산 기계 학습을 수행한다. Caruana et al. (2013)은 스팸 메일 처리를 위한 SVM 기반 필터링 시스템을 Hadoop의 MapReduce 기반으로 구현하였다. 이 과정에서, SVM 계산을 위한 QP 문제를 여러 개의 연속된(serial) 작은 QP 문제로 분리하여 계산하는 Sequential Minimal Optimization (SMO) 접근을 고려하였으며, 분산된 노드는 할당된 학습 데이터에 대한 SMO 계산 결과를 도출한다. Guo et al. (2016)은 대용량의 이미지 분류를 위한 분산 환경에서의 SVM 학습을 수행하였으며, 각기 다른 특징과 성능을 가진 계산 노드들로 구성된 분산 환경을 보다 효율적으로 이용하기 위한 부하 분산(load balancing) 방법을 설계하였다. Hodge et al. (2016)은 Hadoop 기반의 분산된 환경에서 이진 신경망의 한 종류인 Advanced Uncertain Reasoning Architecture(AURA)를 이용하여 빅데이터 수준의 데이터에 대한 변수 선택(feature selection)을 수행하였다. Liu et al. (2017)은 Hadoop과 Spark를 이용하여 일반적인 역전파 신경망(backpropagation neural network)의 가중치 학습에 대한 분산 처리를 실행하였다. Cho et al. (2015)은 자동차 부품의 부하 상태와 수명을 예측하기 위해 분산 환경의 일종인 클라우드 컴퓨팅 환경에서 기계 학습을 수행하였다. 그 외에도, 잠재적 디리클렛 할당(Latent Dirichlet Allocation)이나 결정 트리(Decision Tree), 스펙트럼 군집화(Spectral Clustering)와 같이 많은 계산 성능을 필요로 하는 기계 학습 알고리즘들을 분산된 환경에서 수행하기 위한 연구들이 이루어졌다(Huang et al., 2014; Dai and Ji, 2014; Jin et al., 2013). 또한, Kim and Choi (2020)은 본 논문에 대한 선행 연구로서 분산 환경에서 단일 분류 알고리즘을 수행하고, 이에 따른

효율 상승을 검증하였다.

상기와 같은 분산 기계 학습을 수행한 기존 연구들은 사용된 알고리즘이 해석력(interpretability)과 성능 사이에 존재하는 trade-off를 고려하지 못했다는 한계점을 가진다. 예를 들어, Dai and Ji (2014)가 사용한 결정 트리의 경우 결정 트리를 구성하는 노드에 할당된 규칙을 바탕으로 분류를 수행하기 때문에 사용자가 이러한 규칙과 분류 결과에 대해 명확하게 해석할 수 있다. 그러나 이러한 결정 트리를 학습하는 문제는 NP-hard 문제로 알려져 있어 복잡도가 높고, 변수의 수가 증가함에 따라 지나치게 복잡한 형태의 분류기가 도출되어 성능이 떨어지는 문제가 발생한다. 반면, Chang (2011)에서 사용된 SVM은 이러한 성능 저하 없이 분류를 수행할 수 있지만, 도출된 분류기의 형태가 복잡한 고차원의 함수 형태로 나타나기 때문에 사용자가 분류 요인을 파악하고 이에 기반한 지식 추출을 수행하기가 어렵다. 본 연구가 분산 환경에서 수행하고자 하는 H-RTGL 기반 단일 분류기는 명확하게 해석 가능한 기하학적 규칙인 인터벌(interval)을 이용하여 도출되기 때문에 해석력을 제공할 수 있으며, 분류 성능 또한 해석력을 제공하지 못하는 우수한 분류기들과 비슷한 수준을 유지하였다(Jeong et al., 2019).

3. H-RTGL 기반 기하학적 단일 분류를 위한 분산처리 프레임워크

3.1 H-RTGL 기반 단일 분류기 생성 방법

먼저, 단일 분류 문제는 주어진 데이터 집합 내에 오직 하나의 클래스(class)만이 존재한다고 가정하며, 이에 따라 모든 인스턴스의 소속(membership)은 목표 클래스(target class) 또는 이상치(outlier)로 나누어지게 된다. 이러한 가정 하에 미분류된 인스턴스의 소속을 판별하는 단일 분류기를 도출하는 것이 단일 분류 문제의 목적이라고 할 수 있다(Khan and Madden, 2014). n개의 인스턴스와 q개의 속성 값을 갖는 데이터 집합 $X=\{x_1,x_2,\ \ ...,x_n\}$의 인스턴스 $x_i(i=1,2,...,n)$는 목표 클래스 또는 이상치의 인스턴스이며, 인스턴스 x_i의 r번째 속성 값 $y_{ir}(r=1,2,...,q)$들을 바탕으로 분류기 도출을 수행하게 된다.

또한, 분산 처리를 적용하고자 하는 H-RTGL 기반 기하학적 단일 분류 알고리즘은 Figure 2.와 같이 1) 학습 데이터의 인스턴스들을 각각의 속성에 사영(projection)하여 인터벌(interval)을 생성하고, 2) 속성 별로 생성된 인터벌들을 결합하여 3) H-RTGL 형태의 분류기(classifier)를 획득한다. 인스턴스는 분류기 영역 안에 있을 때 목표 클래스로 판별된다.

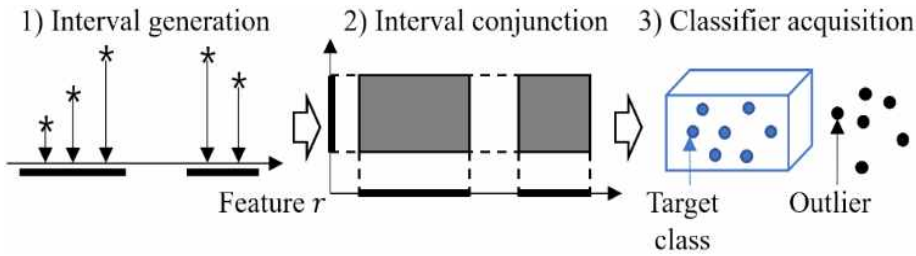

Figure 2. Concept of OCC based on H-RTGL

이와 같은 H-RTGL 기반 단일 분류기의 핵심 요소인 인터벌을 생성하기 위한 다양한 방법이 존재하며, 이에 따라 같은 사영점 집합에 대해 다른 형태의 인터벌이 도출되고 분류기로 사용되는 H-RTGL의 형태가 달라질 수 있다

(Jeong et al., 2019; Kim et al., 2018). 보다 상세한 인터벌 생성 과정 및 분류기 생성 결과의 차이는 참고 문헌들을 통해 확인할 수 있다. 그 중에서도 본 논문에서는 병합(Merging)과 분리(Partitioning), 가우시안(Gaussian)을 통해 인터벌을 생성하는 방법들을 고려하였으며, 해당 방법들의 인터벌 생성 과정은 Figure 3.과 같다. 병합과 분리 기반 인터벌 생성 방법은 각각 작은 인터벌에서 큰 인터벌을 도출하는 과정과 큰 인터벌에서 작은 인터벌을 도출하는 과정을 포함하며, 가우시안 기반 인터벌 생성은 속성의 사영점들을 발생시켰을 것으로 추정되는 가우시안 분포의 통계량을 바탕으로 인터벌을 도출한다.

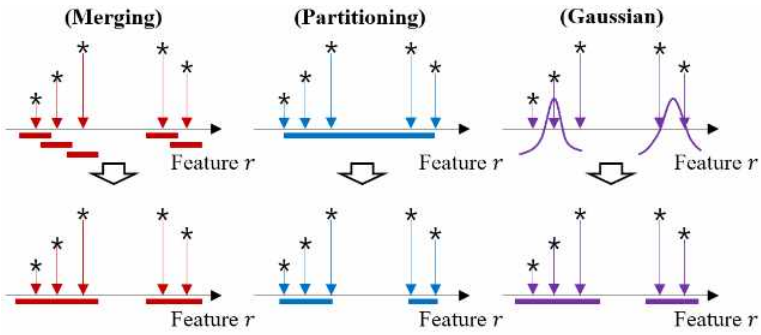

Figure 3. Various Interval Generation methods

3.2 분산 처리를 위한 프레임워크 수립

본 논문에서는 H-RTGL 기반 기하학적 단일 분류 알고리즘의 분산 처리를 위해 Figure 4.와 같은 컴포넌트들로 구성된 프레임워크를 제안하고자 한다.

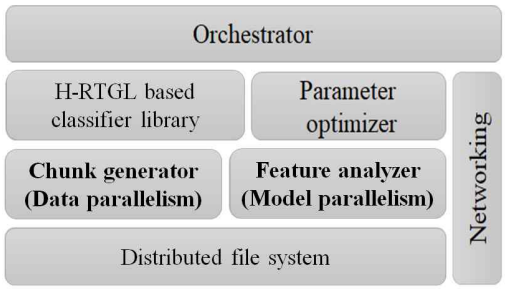

Figure 4. Designed framework for distributed processing

가장 먼저, Orchestrator는 분산 처리를 통한 H-RTGL 기반 단일 분류기 학습 과정을 총괄하며, 작업을 수행하는 worker 노드의 수행 결과들을 취합하는 상위 노드의 역할을 수행한다. 분산 환경은 물리적으로 분산된 컴퓨팅 노드가 네트워크를 통해 연결된 환경을 의미하므로, 네트워킹은 분산 처리를 위한 필수적인 기반 구조(infrastructure)라고 할 수 있다. 또 다른 중요한 기반 구조에 해당하는 분산 파일 시스템은 Hadoop의 HDFS와 같이 저장된 데이터에 대한 입력 및 출력과 같은 작업들을 수행한다. H-RTGL 기반 분류기 라이브러리는 병합, 분리, 가우시안을 이용한 H-RTGL 기반 단일 분류기 생성 알고리즘을 제공함으로써 Hadoop의 Mahout 라이브러리와 대응될 수 있다. 또한, H-RTGL 기반 단일 분류기는 분류기 생성 과정에서 사용되는 매개 변수에 따라 성능의 편차가 발생할 수 있기 때문

에, Parameter optimizer를 통해 보다 좋은 매개 변수를 탐색할 수 있도록 한다. 보다 구체적으로는 Figure 5.와 같이 매개 변수로 구성된 속성 공간에 대한 격자 탐색(Grid search)을 수행하며, 이때 영향력이 큰 매개 변수에 대한 탐색을 먼저 수행하고 나머지 매개 변수에 대한 탐색을 수행한다.

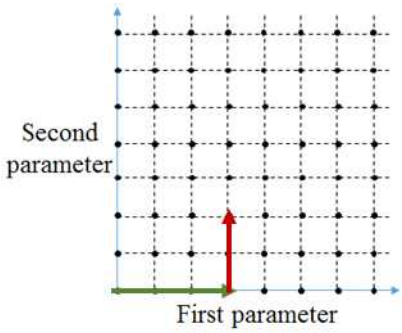

Figure 5. Grid search committed by Parameter optimizer

분산 기계 학습에서 학습을 위한 분산 처리를 수행하는 방법은 크게 모델 분산과 데이터 분산으로 구분된다. 모델 분산 방법은 각각의 노드가 학습을 위한 모델을 나누어 가진 상태에서 동일한 데이터 집합을 이용하여 학습을 수행하며, 데이터 분산 방법은 각각의 노드가 학습을 위한 데이터 집합을 나누어 가진 상태에서 동일한 학습 모델을 이용하여 학습을 수행한다. 본 논문에서는 이러한 2가지 방법에 대한 H-RTGL 기반 단일 분류기의 분산 처리 방법을 다음과 같이 정의하였다.

방법 1: H-RTGL 기반 단일 분류기 생성을 위한 모델 분산은 Feature analyzer를 통해 다음과 같이 수행된다. 먼저, 각각의 노드에 복수의 속성 값과 해당 속성에서의 인터벌 생성을 위해 필요한 매개 변수들을 입력으로 전달하면 해당 노드들이 저차원의 H-RTGL을 생성한다. 이후, 상위 노드에서는 이를 취합하여 생성된 H-RTGL에 대한 논리곱 연산을 수행하고, 추가적인 fitting 절차를 통해 최종적인 분류기에 해당하는 고차원의 H-RTGL을 확정한다. Figure 6.은 6개의 속성을 가진 데이터 집합에 대한 H-RTGL 기반 단일 분류기 생성 과정에서 3개의 노드로 구성된 분산 환경을 이용하여 모델 분산 방법을 적용하는 예시를 나타낸다. 노드 1은 속성 f_1과 f_2에 대한 2차원의 H-RTGL, 노드 2는 속성 f_3과 f_4에 대한 2차원의 H-RTGL, 노드 3은 속성 f_5와 f_6에 대한 2차원의 H-RTGL을 독립적으로 생성한다. 이후, 각각의 속성에 대해 생성된 2차원의 H-RTGL들에 대해 논리곱 연산이 수행되며, 최종적으로는 6개의 속성을 고려하여 생성된 6차원의 H-RTGL을 얻는다. 이때, 각각의 노드가 H-RTGL을 생성하는 과정에서 점선으로 처리된 영역은 인스턴스가 존재하지 않아 분류기 생성 과정에서 배제되는 영역을 나타낸다.

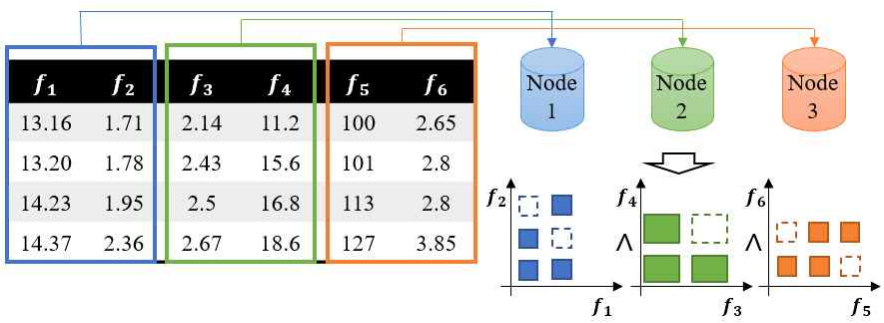

Figure 6. Example of model parallelism

방법 2: H-RTGL 기반 단일 분류기 생성을 위한 데이터 분산은 Chunk generator를 통해 다음과 같이 수행된다. 먼저, 하나의 특성을 기준으로 데이터 집합을 몇 개의 chunk로 분리할 수 있으며, 특정한 데이터 chunk에서의 H-RTGL 기반 단일 분류기 생성은 다른 chunk에 대한 분류기 생성과 독립적으로 수행된다. 즉, 각각의 노드에 분리된 데이터 집합의 일부를 할당하여 동시에 H-RTGL 기반 단일 분류기 생성을 수행하고, 상위 노드는 독립적으로 생성된 H-RTGL 기반 단일 분류기를 취합하여 최종적인 분류기를 결정할 수 있다. Figure 7.은 3개의 속성을 가진 데이터 집합에 대한 H-RTGL 기반 단일 분류기 생성 과정에서 2개의 노드로 구성된 분산 환경을 이용하여 데이터 분산 방법이 적용되는 예시를 나타낸다. 먼저, 속성 f_3을 기준으로 f_3의 값이 낮은 인스턴스로 구성된 chunk와 높은 인스턴스로 구성된 chunk로 학습 데이터 집합을 분할하여 노드 1과 2에 할당한다. 이후, 2개의 노드는 자신이 가진 데이터 집합의 chunk에 대한 H-RTGL을 동시에 병렬적으로 생성한다. 생성된 H-RTGL은 속성 f_3의 값에 따라 독립적으로 생성되었기 때문에, 최종적으로는 두 노드가 생성한 H-RTGL의 합집합(union)이 분류기로 확정된다.

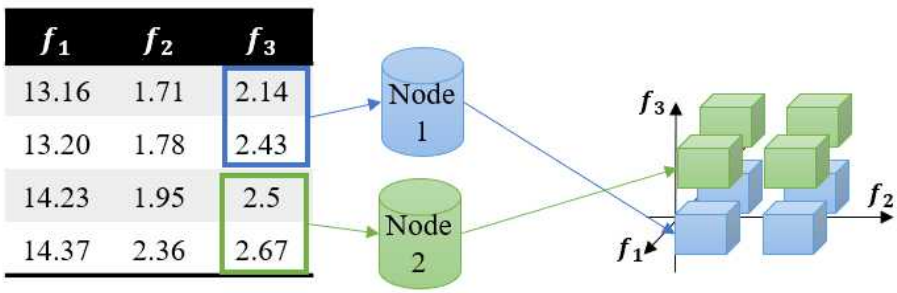

Figure 7. Example of data parallelism

4. 수치 실험을 통한 성능 분석

본 논문에서는 데이터 집합이 매우 많은 인스턴스를 포함하고 있는 빅데이터 처리를 위한 단일 분류기 분산 처리 프레임워크를 수립하고자 하므로, 빅데이터의 분산 처리에 보다 적합한 데이터 분산 방법을 고려하여 단일 분류기를 생성하고, 수치 실험을 통해 이에 대한 성능 분석을 수행하였다.

4.1 실험 설계

앞서 설계된 분산 처리 프레임워크를 이용하여 생성된 H-RTGL 기반 기하학적 분산 단일 분류기의 성능을 검증하기 위한 수치 실험을 다음과 같이 설계하였다. 먼저, 기계 학습 및 데이터 분석을 위한 대표적인 데이터베이스인 UCI machine learning repository에서 제공되는 데이터 집합을 이용하였으며, 크기와 속성의 수가 다양한 5가지의 데이터 집합인 Iris, Breast, Vehicle, Abalone, Satellite 집합들을 고려하였다. 이러한 데이터 집합들은 복수의 클래스로 구성되어 있기 때문에 특정한 클래스를 목표 클래스로 설정하고, 다른 클래스의 인스턴스들을 이상치로 가정하는 One-Versus-All (OvA) 접근을 수행하였다. Table 2.는 각각의 데이터 집합에 대한 정보를 나타내고 있으며, 해당 데이터 집합에서 목표 클래스로 사용한 클래스의 정보를 포함하고 있다.

Table 2. Information about dataset used in this paper

Dataset	Iris	Breast	Vehicle	Abalone	Satellite
# of features	4	9	18	10	36
Target class	Virginica	Malignant	Van	Class 1-8	Grey Soil
Size of target class	50	241	199	1,407	961
Size of outliers	100	458	647	2,770	3,474

본 논문에서는 이와 같은 데이터 집합에 대한 수치 실험을 통해 1) 분산 처리를 통해 생성된 단일 분류기의 분류 정확도 및 2) 분산 처리를 통한 분류기 생성 과정의 효율성에 대한 평가를 수행하였다. 먼저, 분류 정확도를 평가하기 위한 지표로서 Receiver Operating Characteristics (ROC) 곡선의 면적을 나타내는 Area Under ROC curve (AUC)를 고려하였다. ROC 곡선은 목표 클래스로 판별된 인스턴스 중 실제로도 목표 클래스인 인스턴스의 비율을 나타내는 True Positive Rate(TPR)와 실제로는 이상치 데이터인 인스턴스의 비율을 나타내는 False Positive Rate(FPR)을 축으로 사용하여 얻을 수 있다. 이 과정에서, 분류기로서 사용되는 H-RTGL의 크기를 조절하는 매개변수를 증가시키면서 ROC 곡선을 산출하였다. 또한, 목표 클래스는 포함하고 이상치는 배제하는 분류기를 보다 바람직한 분류기로 평가할 수 있도록 목표 클래스의 인스턴스 중 50%만을 분류기 학습을 위해 사용하고, 나머지 50%의 목표 클래스 인스턴스와 이상치를 분류 성능 검증을 위한 시험 데이터로 정의하였다. 분류기 생성 과정의 효율성 측정은 훈련 집합에 대해 H-RTGL 기반 단일 분류기를 생성하기 위해 소요된 시간을 바탕으로 분산 처리에 의한 처리 속도의 증가량을 계산하여 평가하고자 하였다.

실험 수행을 위한 분산 환경 정의는 다음과 같다. 네트워크로 연결된 2개의 클러스터를 고려하며, 클러스터 1은 Orchestrator 및 하나의 worker 노드를 포함하고 클러스터 2는 2개의 worker nodes만으로 구성된다. 각 클러스터는 모두 Intel ® Core (TM) i5-3570 4GHZ, RAM 16GB의 동일한 제원을 가지고 있다. 즉, 3개의 worker nodes가 동시에 H-RTGL 기반 단일 분류기 알고리즘을 수행하며, 분산 처리 방식으로는 chunk generator를 이용한 데이터 분산을 고려하였다. 분산 처리를 적용하지 않은 대조군 실험은 클러스터 1만을 이용하여 실험을 수행하였다. 또한, 대조군으로서 분산 처리를 적용하지 않은 H-RTGL 기반 단일 분류기와 더불어, 기존 연구에서 이용된 다양한 단일 분류 알고리즘의 수행 결과를 함께 기록하여 보다 다각적인 비교가 이루어질 수 있도록 한다. 기존 단일 분류 알고리즘의 수행 결과는 Tax (2010)에 기록된 AUC 값을 이용하였다,

4.2 실험 결과 및 분석

Table 3.은 본 논문에서 설계된 분산 처리 프레임워크를 바탕으로 생성된 H-RTGL 기반 단일 분류기의 성능 실험 결과와 분산 처리를 수행하지 않고 생성된 H-RTGL 단일 분류기 및 다른 단일 분류 알고리즘의 성능 실험 결과를 나타낸다. 5가지의 데이터 집합에 대해, 각 경우마다 20회의 실험을 수행하고, AUC 값의 평균과 표준 편차를 통해 분류기의 성능을 기록하였다. Merging, Partitioning, Gaussian은 인터벌 생성 방법에 따른 H-RTGL 기반 단일 분류기 종류를 나타낸다.

Table 3. AUC and Elapsed time of experiments

Classifier \ Dataset	Iris	Breast	Vehicle	Abalone	Satellite
	AUC * 100 (Standard deviation) / Elapsed time (ms) (Standard deviation)				
Distributed processing using data parallelism (Method 2)					
Merging	95.7 (0.9) / 4.54 (0.11)	94.6 (0.9) / 4.79 (0.13)	92.3 (1.9) / 6.63 (0.15)	86.3 (0.7) / 9.82 (0.24)	90.6 (1.3) / 14.38 (0.63)
Partitioning	97.2 (1.0) / 4.48 (0.09)	95.8 (0.8) / 4.71 (0.11)	83.0 (2.3) / 6.55 (0.14)	68.6 (1.8) / 9.79 (0.23)	82.9 (1.0) / 14.03 (0.58)
Gaussian	97.6 (1.1) / 4.52 (0.10)	96.1 (1.0) / 4.81 (0.13)	93.1 (1.6) / 6.69 (0.18)	87.4 (0.9) / 9.88 (0.26)	91.6 (1.1) / 14.52 (0.65)
Without distributed processing (H-RTGL based One-Class Classifier)					
Merging	96.1 (1.1) / 4.10 (0.08)	95.1 (1.2) / 5.17 (0.12)	92.7 (2.2) / 8.16 (0.17)	86.7 (0.5) / 12.33 (0.63)	91.2 (1.1) / 19.11 (0.81)
Partitioning	97.6 (0.9) / 4.02 (0.06)	96.2 (0.9) / 5.09 (0.15)	83.6 (2.1) / 7.92 (0.18)	69.0 (1.6) / 11.84 (0.59)	83.4 (0.9) / 18.65 (0.74)
Gaussian	97.9 (0.8) / 4.13 (0.09)	96.3 (0.7) / 5.21 (0.13)	93.2 (1.8) / 8.27 (0.19)	87.8 (0.8) / 12.59 (0.68)	91.9 (1.2) / 19.41 (0.77)
Without distributed processing (Other One-Class Classifier from Tax (2010), AUC only)					
Naïve Parzen	95.4 (1.1)	96.5 (0.4)	86.9 (0.2)	85.9 (0.4)	92.3 (0.1)
k-means	95.4 (0.5)	84.6 (3.5)	88.9 (0.6)	79.2 (1.1)	88.6 (18.6)
SVDD	98.1 (0.8)	70.3 (0.5)	91.3 (1.4)	80.6 (0.1)	91.7 (0.1)
Neural Network	96.6 (0.3)	79.0 (2.3)	84.6 (0.8)	81.4 (0.3)	89.8 (1.9)

Table 3.에서, 분류 정확도를 나타내는 AUC 값은 분산 처리 여부와 관계 없이 가우시안을 이용한 H-RTGL 기반 단일 분류기가 가장 높은 값을 가지고 있으며, 분산 처리 여부에 따라 같은 방법론을 이용하는 경우에도 평균적으로 약 0.5% 이내의 차이를 보이고 있음을 확인할 수 있다. 이는 데이터 분산을 수행하고, 이를 취합하는 과정에서 분산 처리를 수행하지 않는 경우의 분류기와는 조금 다른 분류기가 생성되었기 때문에 발생하는 차이라고 할 수 있다. 본 논문에서는 이러한 차이가 통계적으로 유의미한 수준인지 검증하기 위해 대응 표본 t-검정(paired t-test)을 수행하였다. 먼저, 다음과 같은 귀무 가설(null hypothesis) H_0와 대립 가설(alternative hypothesis) H_1을 정의하였다.

$$\begin{aligned} H_0 &: \mu_d = 0 \\ H_1 &: \mu_d \neq 0 \end{aligned} \tag{1}$$

이때, μ_d는 분산 처리를 수행한 경우와 그렇지 않은 경우에 대한 AUC 값 차이의 평균을 나타내며, H_0는 분산 처리에 따른 분류 정확도 차이가 통계적으로 유의미한 수준이 아님을 의미한다. 즉, 검정 결과 H_0가 채택된다면 분산 처리로 인한 분류 정확도의 차이가 없다고 판단할 수 있으며, 기각되는 경우에는 분산 처리가 분류 정확도의 차이를 유발한다고 판단한다. 총 20회의 실험을 수행하였기 때문에 검정의 자유도(degree of freedom)는 19이며, 95%

의 신뢰 구간에 따른 기각 영역(rejection region)은 검정 통계량 t가 $|t| > t_{0.025,\ 19} = 2.093$를 만족하는 영역이 된다. Table 4.는 3가지의 H-RTGL 기반 단일 분류기에 대한 대응 표본 t-검정의 결과를 나타내며, 모든 경우에서 검정 통계량 t가 기각 영역에 해당하지 않고 있기 때문에 H_0가 채택됨을 알 수 있다.

Table 4. Result of t-test for verifying difference of AUC

Classifier	t-statistics	Acceptance of H_0
Merging	1.7738	Accept
Partitioning	1.8934	Accept
Gaussian	1.6953	Accept

결과적으로, 검정의 결과를 기반으로 분산 처리를 수행하여 생성된 H-RTGL 기반 단일 분류기와 분산 처리 없이 생성된 H-RTGL 기반 단일 분류기의 분류 성능은 유사한 수준이라고 볼 수 있다. 그러나, 분리를 이용하는 H-RTGL 기반 단일 분류기는 가장 높은 t값을 가지고 있으며, scale이 큰 학습 데이터 집합에 대해서는 비교적 낮은 정확도를 보이고 있어 빅데이터를 고려한 분산 처리에 사용하기 위해서는 메커니즘의 개선을 고려할 필요가 있다.

그 외에도, Table 3.의 가장 아래쪽 부분을 통해 기존의 다른 분산합의 알고리즘과의 비교 측면에서도 분산 처리를 수행한 H-RTGL 기반 단일 분류기의 성능이 경쟁력을 가지고 있음을 알 수 있다. 일부 사례에서 H-RTGL 기반 단일 분류기에 비해 높은 분류 정확도가 관찰되는 사례가 있지만 그 차이가 크지 않다. 게다가, 해당 단일 분류 알고리즘들은 해석력을 제공하지 못한다는 한계점을 가지고 있어 사후 해석 및 지식 추출까지 가능한 H-RTGL 기반 단일 분류기에 비해 빅데이터 분석을 위한 용도로 사용되기에 적합하지 않을 수 있다.

한편, Table 3.에는 분류 정확도 측정을 위한 AUC 값과 더불어 분류기 생성에 소요된 시간이 기록되어 있다. 분류기 생성을 위해 소요된 시간 측면에서는 Iris 데이터 집합을 제외한 모든 데이터 집합에서 분산 처리를 적용하는 것이 더 효율적인 것을 알 수 있으며, 그 중에서도 특히 분리를 이용한 H-RTGL 기반 단일 분류기가 가장 짧은 시간을 통해 분류기를 생성하고 있다. 예외에 해당하는 Iris 데이터 집합의 경우, 분산 처리를 통해 발생하는 분류기 생성 시간의 차이보다 분산 환경에서 가능한 통신 지연(latency) 등으로 인한 소요 시간의 영향이 더 크기 때문에 이러한 결과가 나타난다고 볼 수 있다. 즉, scale이 적은 데이터 집합의 학습에는 분산 처리를 적용하지 않는 것이 보다 효율적이다.

또한, 분산 처리로 인한 개선 정도를 나타내는 분류기 도출 속도의 증가량(speedup)을 다음과 같이 정의할 수 있으며, 이때, 과 는 각각 분산 처리를 적용하지 않은 소요 시간과 분산 처리에 의한 소요 시간을 나타낸다.

$$speedup = \frac{p_n - p_d}{p_n} * 100\ (\%) \tag{2}$$

Table 5. Calculated speedup value of classifiers in dataset

Classifier	Iris	Breast	Vehicle	Abalone	Satellite
Merging	**-9.63%**	7.35%	18.75%	20.36%	24.72%
Partitioning	-9.52%	7.46%	17.30%	17.31%	24.77%
Gaussian	-9.48%	**7.69%**	**18.86%**	**21.53%**	**25.19%**

Table 5.로부터 이용하는 방법론에 따라 H-RTGL 기반 단일 분류기 도출 속도의 증가량 또한 달라지는 것을 확인할 수 있다. 보다 구체적으로는 분산 처리 자체가 효율적이지 못한 Iris 데이터 집합을 제외한 나머지 학습 데이터 집합에서 가우시안을 이용한 H-RTGL 기반 단일 분류기의 도출 속도 증가량이 가장 크게 나타난다. 즉, 가우시안을 이용한 H-RTGL 기반 단일 분류기는 계산을 위한 소요 시간은 다른 H-RTGL 기반 단일 분류기에 비해 긴 편이지만 분산 처리를 통해 가장 큰 폭의 효율성 개선을 보이고 있어 빅데이터의 분산 처리에 가장 적합한 방법이라고 할 수 있다.

결과적으로, 데이터 분산을 고려한 분산 처리를 통해 1) 분산 처리 이전과 동일한 수준의 분류 성능을 유지하면서도, 2) scale이 큰 데이터 집합에 대해 분류기 생성을 위한 시간을 획기적으로 단축할 수 있으며, 이는 본 논문에서 설계된 프레임워크를 이용한 분산 기계학습이 빅데이터 분석을 위한 단일 분류기 생성을 매우 효율적으로 수행할 수 있음을 의미한다.

5. 결 론

본 논문에서는 H-RTGL 기반 기하학적 단일 분류기 생성을 분산 환경에서 수행함으로써 물리적 제약을 극복하고, 복수의 컴퓨팅 노드를 활용하여 더욱 효율적인 학습을 수행하기 위한 프레임워크를 설계하였다. 보다 구체적으로는 H-RTGL 기반의 기하학적 단일 분류기 생성 알고리즘에 대한 모델 분산 및 데이터 분산 방법을 정의하였으며, 이를 분산 환경에서 수행하기 위한 기반 환경과 지원 컴포넌트들을 포함한 단일 분류기 처리 프레임워크를 수립하였다. 특히, 빅데이터 분석에 보다 적합한 데이터 분산 방법을 적용하였으며, 이를 통해 도출된 H-RTGL 기반 단일 분류기는 분산 처리를 수행하지 않은 경우와 비교하여 정확도 측면에서는 유사한 성능을, 계산 효율 측면에서는 개선된 성능을 확인할 수 있었다. 이러한 결과를 바탕으로, 데이터 집합이 매우 많은 수의 인스턴스로 구성되어 기계학습 알고리즘의 성능이 저하되는 빅데이터를 분석하는 과정에서 본 논문에서 설계된 프레임워크 기반의 접근을 통해 보다 효율적인 분산 처리가 이루어질 수 있을 것으로 기대된다.

이에 대한 추후 연구로서, 본 논문에서 사용된 데이터 집합보다 더 큰 scale을 갖는 빅데이터 집합에 대해 분산 처리의 효과도 및 효율성을 검증하고자 한다. 이에 따라 H-RTGL 기반 단일 분류기 생성을 위한 분류 모델이 복잡해질 것으로 예상되며, 이러한 모델 복잡도 상승을 해결하기 위해 본 논문에서 적용된 데이터 분산과 더불어 feature analyzer에 의해 수행되는 모델 분산을 함께 적용하고자 한다. 이때, 단순히 데이터 또는 모델을 분산 처리하는 것이 아니라, 보다 효과적으로 분류기 생성을 수행할 수 있는 데이터 분할 방법이나 계산 효율을 제고할 수 있는 모델 분산의 기준 등에 대해서도 추가적인 연구가 수행될 수 있다. 또한, H-RTGL 기반 단일 분류기는 매개 변수의 영향을 많이 받기 때문에 본 논문에서 사용된 격자 탐색 방법이 아닌 메타휴리스틱(Metaheuristic) 알고리즘의 적용 등을 통해 매개 변수 공간에 대한 효율적인 탐색과 분류 성능 최적화를 고려한 Parameter optimizer의 개선을 고려할 수 있다.

REFERENCES

Baek, C. H., Choe, J. H., and Lim, S. U. 2018. Review and suggestion of characteristics and quality measurement items of artificial intelligence service. Journal of the Korean Society for Quality Management 46(3):677-694.

Bekkerman, R., Bilenko, M., and Langford, J. 2011. Scaling up machine learning: Parallel and distributed approaches.

Cambridge University Press.

Caruana, G., Li, M., and Liu, Y. 2013. An ontology enhanced parallel SVM for scalable spam filter training. Neurocomputing 108:45-57.

Chang, E. Y. 2011. Psvm: Parallelizing support vector machines on distributed computers. In Foundations of Large-Scale Multimedia Information Management and Retrieval, 213-230.

Chao, T. S. 2001. Introduction to semiconductor manufacturing technology. SPIE PRESS.

Cho, H., Kim, K. T., Jang, Y. H., Kim, S. H., Kim, J. S., Park, K. Y., Jang, J. S., and Kim, J. M. 2015. Development of Load Profile Monitoring System Based on Cloud Computing in Automotive. Journal of the Korean Society for Quality Management 43(4):573-588.

Dai, W., and Ji, W. 2014. A mapreduce implementation of C4. 5 decision tree algorithm. International Journal of Database Theory and Application 7(1):49-60.

Giacomelli, P. 2013. Apache mahout cookbook. Packt Publishing.

Guo, W., Alham, N. K., Liu, Y., Li, M., and Qi, M. 2016. A resource aware MapReduce based parallel SVM for large scale image classifications. Neural Processing Letters, 44(1):161-184.

Hodge, V. J., O'Keefe, S., and Austin, J. 2016. Hadoop neural network for parallel and distributed feature selection. Neural Networks, 78:24-35.

Huang, F., Matusevych, S., Anandkumar, A., Karampatziakis, N., and Mineiro, P. 2014. Distributed latent dirichlet allocation via tensor factorization. In NIPS Optimization Workshop. 1-5.

Jeong, I., Kim, D.G., Choi, J. Y., and Ko, J. 2019. Geometric one-class classifiers using hyper-rectangles for knowledge extraction. Expert Systems with Applications, 117:112-124.

Jin, R., Kou, C., Liu, R., and Li, Y. 2013. Efficient parallel spectral clustering algorithm design for large data sets under cloud computing environment. Journal of Cloud Computing: Advances, Systems and Applications 2(1):18.

Khan, S. S., and Madden, M. G. 2014. One-class classification: taxonomy of study and review of techniques. The Knowledge Engineering Review 29(3):345-374.

Kim, D. G., Choi, J. Y., and Ko, J. 2018. An Efficient One Class Classifier Using Gaussian-based Hyper-Rectangle Generation. Journal of Society of Korea Industrial and Systems Engineering 41(2):56-64.

Kim, D. G., and Choi, J. Y. 2020. A distributed processing framework based on H-RTGL for an efficient One-Class Classification of Big Data. Proceedings of the Korean Society for Quality Management Conference, 137.

Kim, H., Park, J., Jang, J., and Yoon, S. 2016. Deepspark: A spark-based distributed deep learning framework for commodity clusters. arXiv preprint arXiv:1602.08191.

Liu, Y., Xu, L., and Li, M. 2017. The parallelization of back propagation neural network in mapreduce and spark. International Journal of Parallel Programming 45(4):760-779.

Moritz, P., Nishihara, R., Stoica, I., and Jordan, M. I. 2015. Sparknet: Training deep networks in spark. arXiv preprint arXiv:1511.06051.

Padhy, R. P. 2013. Big data processing with Hadoop-MapReduce in cloud systems. International Journal of Cloud Computing and Services Science 2(1):16-27.

Parsian, M. 2015. Data algorithms: Recipes for scaling up with hadoop and spark. O'Reilly Media, Inc.

Peralta, B., Parra, L., Herrera, O., and Caro, L. 2017. Distributed mixture-of-experts for Big Data using PETUUM framework. In 2017 36th International Conference of the Chilean Computer Science Society (SCCC). 1-7.

Tax, D. M. J., 2010. One-class classifier results URL http://homepage.tudelft.nl/n9d04/occ/

Xing, E. P., Ho, Q., Dai, W., Kim, J. K., Wei, J., Lee, S., Zheng, X., Xie, P., Kumar, A., and Yu, Y. (2015). Petuum: A new platform for distributed machine learning on big data. IEEE Transactions on Big Data, 1(2), 49-67.

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. 2010. Spark: Cluster computing with working sets. HotCloud, 10(10-10):95.

저자소개

김도균 아주대학교 산업공학과에서 학사, 석사, 박사 학위를 취득하였다. 현재 아주대학교 공과대학에서 박사후 연구원으로 재직 중이며, 현재 관심 분야는 Data-driven optimization, Multi-Agent Scheduling, Process Mining 등이다.

최진영 한양대학교 산업공학과에서 학사, KAIST 산업공학과에서 석사, Georgia Institute of Technology 산업시스템공학과(ISyE)에서 박사학위를 취득하였다. 현재 아주대학교 산업공학과 교수로 재직 중이며, 주요 관심분야는 Industrial AI Optimization, Quality Control, Process Mining, Modeling & Simulation 등이다.