KIPS Tr. Software and Data Eng.
Vol.5, No.11 pp.593~606 pISSN: 2287-5905

http://dx.doi.org/10.3745/KTSDE.2016.5.11.593

Development of a Simulation Prediction System Using Statistical Machine Learning Techniques

Ki Yong Lee[†] · YoonJae Shin[††] · YeonJeong Choe[††] · SeonJeong Kim[†††] · Young-Kyoon Suh[††††] · Jeong Hwan Sa[†††††] · JongSuk Luth Lee[††††††] · Kum Won Cho[†††††††]

ABSTRACT

Computer simulation is widely used in a variety of computational science and engineering fields, including computational fluid dynamics, nano physics, computational chemistry, structural dynamics, and computer-aided optimal design, to simulate the behavior of a system. As the demand for the accuracy and complexity of the simulation grows, however, the cost of executing the simulation is rapidly increasing. It, therefore, is very important to lower the total execution time of the simulation especially when that simulation makes a huge number of repetitions with varying values of input parameters. In this paper we develop a simulation service system that provides the ability to predict the result of the requested simulation without actual execution for that simulation: by recording and then returning previously obtained or predicted results of that simulation. To achieve the goal of avoiding repetitive simulation, the system provides two main functionalities: (1) storing simulation-result records into database and (2) predicting from the database the result of a requested simulation using statistical machine learning techniques. In our experiments we evaluate the prediction performance of the system using real airfoil simulation result data. Our system on average showed a very low error rate at a minimum of 0.9% for a certain output variable. Using the system any user can receive the predicted outcome of her simulation promptly without actually running it, which would otherwise impose a heavy burden on computing and storage resources.

Keywords : Simulation, Simulation Result Prediction, Statistical Machine Learning

통계적 기계학습 기술을 이용한 시뮬레이션 결과 예측 시스템 개발

이 기 용[†] · 신 윤 재[††] · 최 연 정[††] · 김 선 정[†††] · 서 영 균[††††] · 사 정 환[†††††] · 이 종 숙[††††††] · 조 금 원[†††††††]

요 약

컴퓨터 시뮬레이션은 전산유체역학, 나노 물리, 계산화학, 구조 동역학, 전산설계 등 여러 계산과학공학 분야에서 시스템의 움직임을 예측하기 위해 널리 사용되고 있다. 하지만 시뮬레이션의 정밀도와 복잡도가 점점 증가함에 따라 시뮬레이션을 수행하는 비용 역시 크게 증가하고 있다. 따라서 시뮬레이션의 수행비용을 줄이는 것은 특히 입력 변수들의 값을 변화시켜가며 시뮬레이션을 반복적으로 수행하는 경우, 시뮬레이션 수행 시간 단축을 위해 매우 중요하다. 본 논문은 어떤 시뮬레이션의 수행이 요청되었을 때, 해당 시뮬레이션을 실제로 수행하지 않고도 기존에 수행된 시뮬레이션의 결과를 저장하여 이전에 획득되거나 혹은 예측된 결과를 반환하는 시스템을 개발한다. 이를 위해 본 논문에서 개발된 시스템은 크게 다음 2가지 기능을 제공한다: (1) 수행이 완료된 시뮬레이션의 결과를 데이터베이스에 저장하는 기능, (2) 사용자가 요청한 시뮬레이션의 결과를 통계적 기계학습 기술을 사용하여 예측하는 기능. 본 논문에서 개발한 예측 시스템의 예측 성능을 실제 유체역학 시뮬레이션 데이터를 사용하여 평가한 결과, 출력변수에 따라 0.9%의 매우 낮은 평균 예측 오차율을 보였다. 본 논문에서 개발한 시스템을 통하여 사용자들은 계산 및 저장 자원에 큰 부하를 주는 시뮬레이션을 실제 수행하지 않고도, 수행을 원하는 시뮬레이션의 결과를 빠르게 예측해 볼 수 있다.

키워드 : 시뮬레이션, 시뮬레이션 결과 예측, 통계적 기계학습

※ 이 논문은 2015년도 정부 (미래창조과학부)의 재원으로, 한국연구재단 첨단사이언스 및 교육 허브 개발 사업(EDISON)의 지원을 받아 수행된 연구임(No. NRF-2011-0020576).

† 종신회원 : 숙명여자대학교 컴퓨터과학부 부교수
†† 준 회 원 : 숙명여자대학교 컴퓨터과학부 석사과정
††† 비 회 원 : 숙명여자대학교 컴퓨터과학부 연구원
†††† 정 회 원 : 한국과학기술정보연구원 슈퍼컴퓨팅융합연구센터 선임연구원
††††† 비 회 원 : 한국과학기술정보연구원 슈퍼컴퓨팅융합연구센터 박사후 연구원
†††††† 정 회 원 : 한국과학기술정보연구원 슈퍼컴퓨팅융합연구센터 책임연구원
††††††† 비 회 원 : 한국과학기술정보연구원 슈퍼컴퓨팅융합연구센터 책임연구원

Manuscript Received : October 5, 2016
Accepted : October 13, 2016

* Corresponding Author : Young-Kyoon Suh(yksuh@kisti.re.kr)

1. 서 론

컴퓨터 시뮬레이션은 전산유체역학, 물리, 계산화학, 구조동역학, 전산설계 등 여러 계산과학공학 분야에서 어떤 시스템의 움직임을 예측하기 위해 널리 사용되는 기술이다[1]. 예를 들어 전산유체역학 분야에서 컴퓨터 시뮬레이션은 항공기 날개 주변의 공기 흐름을 예측하는데 널리 사용된다. 이 경우 시뮬레이션 프로그램은 주어진 입력 변수들의 값을 받아 수치해석 기법으로 방정식들의 근을 풀어 시스템의 상태를 나타내는 변수 값들이 어떻게 변하는지를 계산한다.

하지만 시뮬레이션의 정밀도와 복잡도가 점점 증가함에 따라 시뮬레이션을 수행하는 비용 자체도 역시 크게 증가하고 있다. 특히 입력 변수들의 값을 변화시켜가며 반복적으로 시뮬레이션을 수행하는 경우, 시뮬레이션 수행에 드는 비용이 커질수록 전체 수행 시간은 크게 늘어나게 된다. 예를 들면, 온라인 계산과학공학 시뮬레이션 플랫폼[2]에 탑재된 다양한 고성능 계산(HPC) 시뮬레이션 소프트웨어에 대해 제출된 시뮬레이션은 작업 대기(Job Queuing) 시간 및 입력 변수들의 값 조합에 따라 수 시간, 혹은 며칠 때로는 몇 주 이상 걸리기도 한다. 따라서 시뮬레이션의 수행 요청이 들어 왔을 때, 요청된 시뮬레이션을 가능한 적은 비용으로 처리하는 것이 매우 중요하다.

시뮬레이션의 수행 시간을 단축하기 위해서 지금까지 크게 두 가지 접근 방법이 사용되어 왔다. 첫 번째는 시뮬레이션을 수행하는 컴퓨터 하드웨어의 성능을 높이는 것이다. 이를 위해 멀티코어 CPU나 멀티코어 GPU를 사용하여 시뮬레이션 프로그램을 병렬적으로 수행하거나, 다수의 컴퓨터를 네트워크로 연결한 컴퓨터 클러스터를 구축하여 컴퓨터 하드웨어의 성능을 높이는 방법 등이 사용되어 왔다. 두 번째는 시뮬레이션을 수행하는 수치해석 알고리즘을 개선하는 것이다. 이것은 알고리즘의 불필요한 계산을 제거하거나 병렬성을 증가시킴으로서 가능하다. 예를 들어 방정식의 근을 푸는 데에는 문제 형태에 따라 서로 다른 성능을 가진 여러 알고리즘들이 존재한다.

사용자가 온라인으로 요청한 시뮬레이션을 수행하고, 그 결과를 온라인으로 반환하는 시뮬레이션 서비스 시스템은 지금까지 많이 개발되어 왔다. 하지만 지금까지의 시뮬레이션 서비스 시스템은 시뮬레이션의 수행 시간을 단축하기 위해 앞서 언급한 두 가지 접근 방법만을 사용해 왔으며, 이미 수행된 시뮬레이션들의 결과를 활용하여 이후에 요청되는 시뮬레이션의 수행 비용을 줄이는 연구는 거의 이루어지지 않았다. 즉, 사용자가 어떤 시뮬레이션의 수행을 요청했을 때, 대부분의 시뮬레이션 서비스 시스템들은 기존의 시뮬레이션 결과를 활용하지 않고, 사용자가 요청한 시뮬레이션을 처음부터 새로 수행한다. 또는 기존의 결과를 활용하더라도 사용자가 직접 원하는 결과를 검색하는 제한적인 방법만을 제공하고 있다.

따라서 본 논문은 이미 수행이 완료된 기존 시뮬레이션의 결과를 활용하여, 이후 사용자가 요청한 시뮬레이션을 수행하는 비용을 낮추는 시뮬레이션 서비스 시스템을 개발한다. 본 논문에서 개발한 시스템은 크게 다음 2가지 방식으로 기존 시뮬레이션 결과를 활용한다. (1) 재활용(reuse): 기존 시뮬레이션의 결과를 데이터베이스에 저장하고, 이후 사용자가 동일한 시뮬레이션의 수행을 요청하면 해당 시뮬레이션을 수행하지 않고 데이터베이스에 저장된 기존 시뮬레이션 결과를 반환한다. (2) 예측(prediction): 만약 사용자가 요청한 시뮬레이션의 결과가 데이터베이스에 없을 경우, 기존에 저장된 시뮬레이션의 결과를 분석하여 시뮬레이션을 실제 수행하지 않고도 사용자가 요청한 시뮬레이션의 결과를 예측하는 기능을 제공한다.

따라서 본 논문의 시스템은 시뮬레이션의 중복적인 수행을 제거하거나 시스템에 부담을 주는 불필요한 시뮬레이션의 수행을 방지함으로써, 보다 많은 사용자가 보다 효율적으로 시뮬레이션 서비스 시스템을 사용할 수 있도록 한다. 특히 본 논문의 시스템은 사용자가 요청한 시뮬레이션 결과를 예측하기 위해 통계적 기계학습(statistical machine learning)에 기반한 다양한 방법을 사용한다. 본 논문에서 개발한 예측 시스템은 통계 분석 프로그래밍 환경인 R[3]과 연동하여 선형 회귀(linear regression), 일반 가산 모델(generalized additive model), 지지 벡터 머신(support vector machine), 결정 트리(decision tree), 회귀 스플라인(regression spline), k-최근접 이웃 회귀(k-nearest neighbor regression), 다층 신경망(multi-layer neural network) 등의 다양한 예측 모델에 기반한 예측 기능을 제공한다. 실제 시뮬레이션 데이터를 활용한 예측 성능 실험 결과, 제안 시스템은 출력 변수에 따라 최소 0.9%의 예측 오차율을 보임을 확인하였다. 따라서 본 논문에서 개발한 시스템을 통하여 사용자들은 계산 및 저장 자원에 큰 부하를 주는 시뮬레이션을 실제 수행하지 않고도, 수행을 원하는 시뮬레이션의 결과를 빠르게 예측해 볼 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 기존 시뮬레이션 서비스 시스템과, 본 논문에서 사용한 통계적 기계학습 기법들을 간략히 살펴본다. 3장에서는 본 논문에서 개발한 시스템의 기능 및 구조를 설명하고, 4장에서는 본 논문에서 개발한 예측 시스템의 예측 성능 실험 결과를 보인다. 5장에서는 결론을 맺는다.

2. 관련 연구

본 장에서는 먼저 기존의 시뮬레이션 서비스 시스템들의 현황을 알아보고, 본 논문에서 시뮬레이션 결과를 예측하기 위해 사용한 통계적 기계학습 방법들을 간략히 살펴본다.

2.1 기존 시뮬레이션 서비스 시스템

웹을 통해 시뮬레이션의 수행을 요청하고 결과를 받을 수 있는 시스템은 이미 다수 존재한다. PhET[4]는 물리, 화학, 생물 등에 대한 다양한 시뮬레이션 기능을 제공하는 교육용

시스템이며, ALF[5]는 유전자 시뮬레이션 기능을 제공한다. BiDaS[6]는 생물학 관련 대규모 시뮬레이션 기능을 제공하는 시스템이며, WebArrayDB[7]는 웹을 통해 미세배열(microarray) 데이터에 대한 시뮬레이션 결과를 제공하는 시스템이다. 하지만 이들 시스템 모두 기존 데이터를 활용하지는 않고 요청된 시뮬레이션을 매번 새로 수행한다.

수행이 완료된 기존 시뮬레이션 결과를 공유하는 시스템도 이미 다수 개발되어 있다. DataSpaces[8]는 시뮬레이션 데이터 공유를 위한 질의 및 색인 API를 제공하는 시스템이다. BIGNASim[9]은 분자 시뮬레이션 결과를 NoSQL 데이터베이스에 저장하여 검색이 가능하며, SciDrive[10]는 다양한 형태의 파일로 저장된 과학 데이터에 대한 검색이 가능한 시스템이다. DCMS[11]는 분사 시뮬레이션 결과를 관계형 데이터베이스에 저장하여 검색이 가능하도록 한 시스템이며, iBIOMES[12]는 바이오 분자 및 계산화학 시뮬레이션 결과를 MySQL에 저장함으로써 검색 기능을 제공한다. SciBox[13]는 시뮬레이션 데이터를 저장하는 클라우드 시스템으로서 DropBox와 유사한 인터페이스를 통해 데이터를 쉽게 접근할 수 있도록 한다. 하지만 이들 시스템은 모두 기존의 시뮬레이션 결과를 저장하고 공유하기 위한 시스템이며, 새로 요청된 시뮬레이션의 수행 비용을 기존 시뮬레이션 결과를 활용하여 낮추는 기능은 제공하지 않는다. [14, 15]는 본 연구의 사전연구로서, 새로 요청된 시뮬레이션의 결과가 이미 데이터베이스에 존재하는 경우 이를 활용하는 시뮬레이션 서비스 시스템의 구조를 제안하였으나, 예측 기능을 실제로 구현하여 제공하지는 않는다.

2.2 시뮬레이션 결과 예측을 위한 통계적 기계학습 기법

일반적으로 시뮬레이션 프로그램은 입력변수 X_1, X_2, ..., X_n의 값을 받아 수치해석 알고리즘을 수행하고, 그의 결과로서 출력변수 Y_1, Y_2, ..., Y_m의 값들을 출력하며 이것이 시뮬레이션 결과에 해당한다. 따라서 많은 시뮬레이션 결과가 누적되면 될수록 X_1, X_2, ..., X_n의 값이 주어졌을 때 Y_1, Y_2, ..., Y_m의 값을 예측하는 것이 가능해지며, 이를 위해 통계적 기계학습 방법을 사용할 수 있다.

통계적 기계학습 방법 중 회귀 분석(regression analysis)이란 주어진 독립변수 $X = (X_1, X_2, ..., X_n)$와 종속변수 $Y = (Y_1, Y_2, ..., Y_m)$이 있을 때 다음 형태의 함수를 찾는 기법이다[14].

$$Y = f(X, \beta)$$

여기서 β는 알려지지 않은 매개변수들을 나타내며, 회귀분석은 예측 오차를 최소화하는 f와 β를 찾는 것을 목표로 한다. 본 논문의 시뮬레이션 결과 시스템에서는 시뮬레이션 입력변수 X_1, X_2, ..., X_n이 독립변수 X에 해당하고, 시뮬레이션 출력변수 Y_1, Y_2, ..., Y_m이 종속변수 Y에 해당한다. 여기서 함수 f의 형태를 어떻게 가정하느냐에 따라 다양한 통계적 기계학습 방법이 사용될 수 있다. 본 논문에서는 시뮬레이션 결과를 예측하기 위해 여러 대표적인 통계적 기계학습 방법들을 사용한다. 본 논문에서는 각 시뮬레이션 출력변수 Y_i에 대해 각각 독립적인 예측 모델을 생성한다. 즉, 각 Y_i에 대해 $Y_i = f_i(X, \beta_i)$를 만족하는 f_i와 β_i를 각각 찾는다. 아래에서는 본 논문에서 시뮬레이션 결과 예측을 위해 사용한 통계적 기계학습 기법들을 간략히 설명한다.

1) 다중 선형 회귀(multiple linear regression)

이 방법은 가장 기본적인 회귀 분석 기법으로서, 시뮬레이션 출력변수 Y의 값이 입력변수 X_1, X_2, ..., X_n들의 값과 선형적인 관계를 가진다고 가정한다. 즉, 다음 식에서 관측값들에 대한 오차를 최소화하는 β를 찾는 것을 목표로 한다.

$$Y = \beta_0 + \sum_{i=1}^{n} X_i \beta_i$$

2) 일반 가산 모델(generalized additive model, GAM)

선형 회귀를 확장하여, 출력변수 Y의 값이 각 입력변수 X_i에 대해 미지의 매끄러운 함수(smooth function) f_i를 적용하여 나온 값의 합으로 표현될 수 있다고 가정한다. 따라서 이 방법은 다음 식에서 관측값들에 대한 오차를 최소화하는 β_0와 f_i를 찾는 것을 목표로 한다.

$$Y = \beta_0 + \sum_{i=1}^{n} f_i(X_i)$$

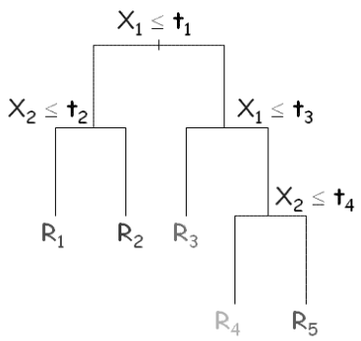
3) 지지 벡터 머신(support vector machine, SVM)

주어진 훈련 데이터들에 대해 가장 가까운 데이터와의 거리(margin)를 최대로 만드는 선을 찾아 이를 회귀 모델로 한다. 따라서 다음 식으로 표현된 모델에서 가장 가까운 데이터와의 거리를 최대화하는 β_0와 $w = (w_1, w_2, ..., w_n)$를 찾는 것을 목표로 한다. 특히 아래 모델에서 데이터를 다양한 함수 Φ를 적용하여 고차원 데이터로 변형함으로써 더 복잡한 모델을 찾을 수도 있다. 따라서 어떤 함수 Φ를 사용하느냐에 따라 예측 성능이 크게 달라질 수 있다.

$$Y = \beta_0 + w^T \Phi(X)$$

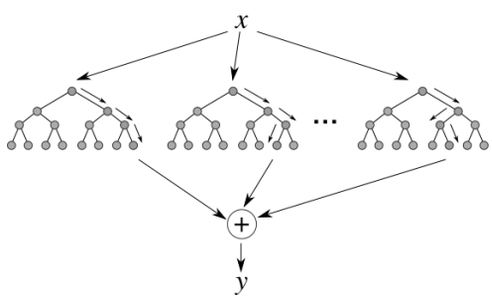
4) 분류 및 회귀 트리(classification and regression tree, CART)

분류(classification)에 널리 사용되는 기법인 결정 트리(decision tree)를 회귀 분석에 적용한 방법이다. 따라서 주어진 데이터가 어느 클래스에 속하는지를 예측 결과로 출력하는 분류와는 달리 실수(real number)를 출력하는 것이 다르다. 다음은 CART의 예를 나타내는 그림이다. 이 방법의 성능을 좌우하는 중요한 매개변수는 언제까지 노드를 분할해 나가느냐를 결정하는 데 사용되는 복잡도(complexity) 매개변수이다.



5) 랜덤 포레스트(random forest)

주어진 훈련 데이터에서 임의로 데이터를 추출하여 하나가 아닌 많은 수의 결정 트리들을 만든다. 입력변수 X가 주어지면 이를 모든 결정 트리에 적용하여 각각의 예측치를 구하고, 이들 예측치들의 평균으로서 최종 예측값을 얻는다. 이 방법의 성능을 좌우하는 중요한 매개변수는 트리의 개수보다는 노드를 분할하기 위해 임의로 선택하는 변수의 개수이다.



6) 일반 부스팅 모델(generalized boosted model, GBM)

다수의 결정 트리를 만드는 것은 랜덤 포레스트와 유사하다. 하지만 각 결정 트리를 $h_1(X)$, $h_2(X)$, ..., $h_M(X)$라 하면 다음 식으로 최종 예측 값을 모델링하며, 관측값에 대한 오차를 최소화하는 γ_i를 찾는다. 이 방법의 정확도는 결정 트리의 개수 M을 몇 개로 하느냐에 크게 좌우되며, 일반적으로 M이 클수록 정확도가 증가한다.

$$Y = \beta_0 + \sum_{i=1}^{M} \gamma_i h_i(X)$$

7) 다변수 적응 회귀 스플라인(multivariate adaptive regression spline, MARS)

선형 회귀 모델에 비선형성과 변수들 간의 상호작용을 자동으로 반영하는 방법으로서, Y 값을 다음 식과 같이 여러 개의 기저 함수 $B_i(X)$들의 가중치 합으로 나타낸다. 아래 모델에서 관측값에 대한 오차를 최소화하는 k와 c_i를 찾는 것이 목표이다. 기저 함수는 상수 1, 경첩(hinge) 함수, 둘 이상의 경첩 함수의 곱 3가지 형태 중 하나를 가지며, 어떤 기저 함수를 사용하느냐에 따라 성능이 달라진다.

$$Y = \sum_{i=1}^{k} c_i B_i(X)$$

8) 국소 회귀(local regression)

주어진 입력변수의 값 X에 대해 그와 거리가 가장 가까운 일부 데이터들만 사용하여 해당 데이터들에 대해 오차의 제곱의 합을 최소화하는 다항식을 구하여 이를 X에 대한 근사식으로 사용한다. 다항식을 구할 때는 X에 더 가까운 데이터일수록 오차의 크기를 더 크게 반영한다. 주어진 X의 근처의 데이터를 사용하여 얻어진 근사 다항식을 $\mu(X)$라고 할 때, 국소 회귀식은 다음과 같이 표현된다. 일반적으로 다항식은 2차 다항식을 많이 사용하며, 주어진 X에 대해 전체 데이터 중 X에 가까운 데이터를 얼마큼 사용하여 근사 다항식을 만드느냐에 따라 성능이 크게 좌우된다.

$$Y = \mu(X)$$

9) k-최근접 이웃 회귀(k-nearest neighbor (k-NN) regression)

이 방법은 주어진 입력변수의 값 X에 대해 그와 거리가 가장 가까운 k개의 데이터를 구한 후, 그들에 대한 출력변수 Y의 값을 평균하여 최종 예측 값을 구한다. 평균값을 구할 때는 거리가 가까운 데이터에 더 큰 가중치를 주는 가중치 평균값을 사용할 수도 있다. 이 방법에서 예측 값을 구하는 식은 다음과 같이 표현될 수 있다. 여기서 $N_k(X)$는 X와 가장 가까운 k개의 데이터로 이루어진 집합을 나타내며, y_i는 입력변수 값 x_i에 대한 출력변수의 값을 나타낸다. 이 방법의 성능은 k의 값에 따라 크게 변할 수 있다.

$$Y = \frac{1}{k} \sum_{x_i \in N_k(X)} y_i$$

10) 다층 신경망(multilayer neural network)

다층 신경망은 인간의 뇌신경을 모델링한 방법으로서, 모델은 일반적으로 여러 층(layer)로 구성되어 있고, 각 층은 또한 여러 개의 뉴런(neuron)으로 구성된다. i번째 층과 i+1번째 층의 뉴런들은 서로 연결되어 있다. 어떤 뉴런에게 입력으로 전달되는 값들이 x_1, x_2, ..., x_m이고 x_i에 대한 가중치는 w_i이며, 해당 뉴런이 출력하는 값을 y라 할 때 y는 다음과 같은 식으로 표현된다. 이 모델에서는 관측값에 대해 오차를 최소화하는, 모델에 포함된 모든 w_i 값을 구하는 것이 목표가 된다. 여기서 $K(x)$는 사전에 정의된 함수로서 보통 활성화(activation) 함수라 부른다. 이 방법은 층의 개수와 뉴런의 개수에 따라 성능이 크게 달라진다.

$$y = K(\sum_{i=1}^{m} w_i x_i)$$

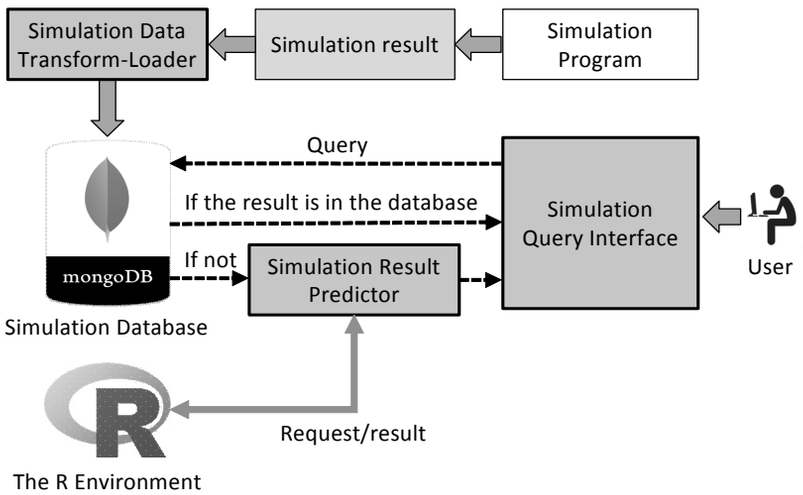


Fig. 1. The Overall System Architecture

지금까지 본 논문에서 시뮬레이션 결과 예측을 위해 사용한 여러 종류의 통계적 기계학습 기법들을 살펴보았다. 물론 모든 종류에 데이터에 대해 항상 최고의 성능을 보이는 방법은 있을 수 없으며, 주어진 데이터에 대해 훈련을 통해 예측 모델을 생성한 뒤, 해당 모델의 예측 성능을 평가하여 가장 좋은 예측 성능을 가지는 모델을 선택하는 것이 바람직하다. 4장에서는 실제 시뮬레이션 데이터를 가지고 다양한 예측 모델을 생성하여 실제 예측 성능을 측정한 실험 결과를 보인다.

3. 제안 시뮬레이션 서비스 시스템

본 장에서는 본 논문에서 개발한 시뮬레이션 서비스 시스템의 전체 아키텍처와 주요 기능을 설명한 후, R과 연동하여 시뮬레이션 결과를 예측하는 시뮬레이션 결과 예측 모듈의 구현 방법을 상세히 기술한다.

3.1 전체 시스템 아키텍처 및 주요 기능

Fig. 1은 본 논문에서 개발한 시뮬레이션 서비스 시스템의 전체 아키텍처이다. 본 논문에서 개발한 시스템은 기존 시뮬레이션 서비스 시스템과 달리 사용자가 요청한 시뮬레이션을 매번 새로 수행하는 대신, 가능하면 기존의 시뮬레이션 결과를 활용하여 새로 요청된 시뮬레이션을 처리한다. 이를 위해 본 논문에서 구현된 시스템은 크게 3개의 컴포넌트로 구성된다.

- **시뮬레이션 데이터 변환 및 적재기(Simulation Data Transform-Loader)**

일반적으로 시뮬레이션 프로그램은 시뮬레이션을 수행하고 나면 그의 수행 결과를 파일 형태로 출력한다. 해당 결과를 추후에 재활용하기 위해 시뮬레이션 데이터 변환 및 적재기는 시뮬레이션 프로그램이 출력한 파일로부터 필요한 정보를 추출하여 이를 데이터베이스에 적재하는 기능을 제공한다. 더 구체적으로 시뮬레이션 데이터 변환 및 적재기는 시뮬레이션 프로그램의 출력 파일을 인자로 받아 해당 출력 파일로부터 시뮬레이션 수행에 사용된 입력변수들의 값과 시뮬레이션 수행 결과로 출력된 출력변수들의 값을 추출한다. 그리고 추출한 값을 바탕으로 (1) 시뮬레이션 프로그램 정보, (2) 시뮬레이션 입력변수들의 값, (3) 시뮬레이션 출력변수들의 값들을 포함하고 있는 JSON 문서를 생성한다. 그리고 생성된 JSON 문서를 현재 널리 사용되고 있는 대표적인 NoSQL 데이터베이스 중의 하나인 MongoDB[16]에 적재한다. MongoDB는 문서 형태의 대규모 데이터를 저장하는 기능을 제공하는 오픈 소스 NoSQL 데이터베이스로서, 미리 스키마가 정의되어 있지 않은 임의의 형태를 가진 JSON 문서를 저장하는 동적 스키마(dynamic schema) 기능을 제공한다. 또한 NoSQL의 대표적 특징 중의 하나인 다수의 컴퓨터로 구성된 클러스터를 사용한 대규모 데이터의 저장이 가능하다. 이러한 이유로 본 논문에서 개발한 시뮬레

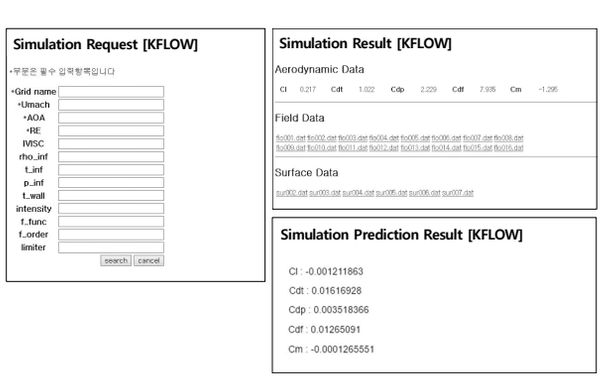


Fig. 2. Simulation Query Interface

이션 서비스 시스템은 시뮬레이션 결과를 저장하는데 MongoDB를 사용한다. 또한 시뮬레이션 데이터 변환 및 적재기는 대량 삽입(bulk insert) 기능을 제공한다. 만약 데이터베이스에 적재할 시뮬레이션 결과가 많을 경우, 시뮬레이션 데이터 변환 및 적재기는 이들을 일괄적으로 JSON 문서로 변환하여 순차적으로 MongoDB에 적재함으로써 사용자에게 편의를 제공한다.

- **시뮬레이션 질의 인터페이스(Simulation Query Interface)**

사용자는 시뮬레이션 질의 인터페이스를 통해 웹으로 시뮬레이션의 수행을 요청하고 그의 결과를 전달받을 수 있다. Fig. 2는 시뮬레이션 질의 인터페이스의 수행 예를 보여준다. 사용자가 시뮬레이션 프로그램과 그의 수행에 필요한 입력변수들의 값을 입력하면, 시뮬레이션 질의 인터페이스는 요청된 시뮬레이션에 대한 결과가 이미 데이터베이스에 저장되어 있는지 먼저 확인한다. 만약 해당 데이터가 이미 존재하면 시스템은 시뮬레이션을 새로 수행하지 않고 해당 데이터를 사용자에게 바로 반환한다. 따라서 시스템이 시뮬레이션을 수행해야 하는 비용이 제거된다. 만약 해당 데이터가 없으면 사용자에게 두 가지 옵션을 선택할 수 있도록 한다. 첫 번째는 해당 시뮬레이션을 실제로 수행하는 기능이고, 두 번째는 해당 시뮬레이션을 수행하지 않고 예측 결과만을 반환하는 기능이다. 만약 사용자가 두 번째 옵션을 선택하면 시스템은 시뮬레이션 결과 예측기를 호출하여 요청된 시뮬레이션의 결과를 예측하고 해당 예측 결과를 반환한다.

- **시뮬레이션 결과 예측기(Simulation Result Predictor)**

사용자가 시뮬레이션 데이터 변환 및 적재기를 통해 데이터베이스에 적재한 시뮬레이션 결과 데이터가 많이 쌓이게 되면 해당 데이터를 사용하여 새로 요청된 시뮬레이션의 결과를 예측하는 것이 가능해진다. 이때 다양한 예측 모델을 사용할 수 있다. 물론 어떤 예측 모델이든 시뮬레이션의 결과 예측 정확도를 확실히 보장해 주기는 어렵기 때문에, 이 기능은 100% 확실한 결과를 보장한다기보다는 시스템에 부하를 주지 않고 아주 빠르게 결과를 짐작해보는데 사용되는 것이 바람직하다. 따라서 입력변수를 다양하게 변화시켜가며 결과의 패턴을 빠르게 파악해보는데 매우 유용하다. 시뮬레이션의 결과를 예측하기 위해 본 시스템은 현재 총 10개의 예측 모델을 제공하고 있으며 이후로도 계속 추가하는 것이 가능하다. 현재 본 논문에서 구현한 시뮬레이션 결과 예측기가 시뮬레이션 결과 예측을 위해 사용하는 예측 모델은 2.2절에서 설명한 10개의 모델이다. 즉, 다중 선형 회귀, 일반 가산 모델, 지지 벡터 기계, 분류 및 회귀 트리, 랜덤 포레스트, 일반 부스팅 모델, 다변수 적응 회귀 스플라인, 국소 회귀, k-최근접 이웃 회귀, 다층 신경망을 통해 시뮬레이션 결과를 예측할 수 있다. 각 모델은 데이터에 따라 예측 성능이 서로 다르며, 4장에서 실제 시뮬레이션 데이터에 적용해본 실험 결과를 보인다. 본 시스템은 각각의 예측 모델을 직접 구현하는 대신 통계적 기계학습 프로그래밍에 널리 사용되고 있는 R 라이브러리[2]를 사용하여 예측 기능을 구현하였다. 이에 대해서는 3.2절에서 좀 더 자세히 설명한다.

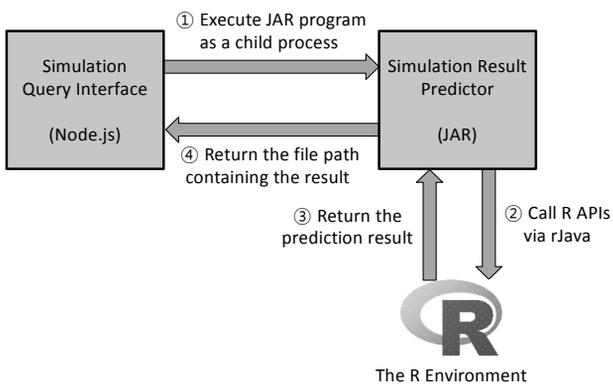


Fig. 3. Call Flow of the Simulation Result Predictor

3.2 시뮬레이션 결과 예측기 구현

만약 사용자가 시뮬레이션 질의 인터페이스를 통해 어떤 시뮬레이션의 결과 예측을 요청하면 시뮬레이션 결과 예측기가 호출된다. Fig. 3은 시뮬레이션 결과 예측기의 호출 흐름을 나타낸다. 본 논문의 시뮬레이션 질의 인터페이스는 Node.js[17]를 사용하여 구현되었으며, 시뮬레이션 결과 예측기는 Java 클래스로 구현되었다. 사용자가 시뮬레이션 결과 예측을 요청하면 시뮬레이션 질의 인터페이스에 해당하는 Node.js 프로그램은 자식 프로세스(child process)를 생성하여 시뮬레이션 결과 예측기에 해당하는 JAR 파일의 수행을 시작한다. 이때 결과를 예측하고자 하는 시뮬레이션의 모든 입력변수들의 값을 시뮬레이션 결과 예측기에 전달한다. Java 클래스로 구현된 시뮬레이션 결과 예측기는 수행이 시작되면, R 패키지 형태로 제공되는 rJava 인터페이스[18]를 통해 R에게 시뮬레이션 결과 예측에 필요한 명령을 차례대로 전달하고 R로부터 결과를 전달받는다. 시뮬레이션 결과 예측기는 R로부터 모든 예측 결과를 전달받으면 예측 결과를 파일에 기록한 뒤 수행을 종료한다. 시뮬레이션 결과 예측기의 수행이 종료되면 시뮬레이션 질의 인터페이스에 해당하는 Node.js 프로그램은 파일에 기록된 시뮬레이션 결과 예측기의 수행결과를 읽어와 이를 화면에 출력한다. 따라서 시뮬레이션 결과 예측기는 예측에 필요한 모든 작업을 R에서 제공하는 다양한 패키지를 사용하여 수행할 수 있으며, 따라서 추후로도 유연한 기능 확장이 가능하다.

4. 시뮬레이션 결과 예측 성능 실험

본 장에서는 본 논문에서 구현한 시뮬레이션 결과 예측기를 실제 시뮬레이션 데이터에 적용하여 예측 성능을 평가한 실험 결과를 보인다.

4.1 실험 환경 및 방법

본 실험에서 시뮬레이션 결과 데이터는 KISTI가 제공한 EDISON[2] KFLOW 시뮬레이션 프로그램의 실행 결과 데이터를 사용하였다. EDISON KFLOW는 KISTI에서 개발한 에어포일(airfoil) 시뮬레이터로서 비행기 날개 주변의 공기 흐름을 시뮬레이션하여 양력, 항력, 압력, 모멘트 등을 계산한다[19]. EDISON KFLOW 시뮬레이션 프로그램의 주요 입력변수는 다음과 같다.

- Thickness: NACA 에어포일[20]의 두께를 나타낸다. 주어진 시뮬레이션 결과 데이터에서는 9 ~ 12의 범위를 가진다.
- MACH: 마하 수(Mach number)로서 음속에 비하여 속도가 얼마가 되는지를 나타내는 수이다. 주어진 시뮬레이션 결과 데이터에서는 0.05 ~ 0.6의 범위를 가진다.
- AOA: 받음각(angle of attack)으로서 물체의 중심선과 물체로 흐르는 공기 흐름 간의 각도 크기를 나타낸다. 주어진 시뮬레이션 결과 데이터에서는 0° ~ 10°의 범위를 가진다.
- RE: 레이놀즈 수(Reynolds number)로서 관성에 의한 힘과 점성에 의한 힘의 비를 나타낸다. 주어진 시뮬레이션 결과 데이터에서는 10^5 ~ 10^6의 범위를 가진다.

EDISON KFLOW 시뮬레이션 프로그램의 시뮬레이션 결과로 출력되는 주요 출력변수는 다음과 같으며, 이것이 상기 입력변수들의 값이 주어졌을 때 본 시뮬레이션 결과 예측기가 값을 예측해야 하는 변수가 된다.

- Cl: 양력 계수(lift coefficient)로서 유체가 물체 주위로 흐를 때 물체의 윗면과 아랫면의 압력 차이로 발생하는 수직 성분의 힘을 나타낸다.
- Cdt: 항력 계수(total drag coefficient)로서 유체가 물체 주위로 흐를 때 물체 주위의 압력 분포 및 표면 마찰로 인해 물체의 전진을 방해하는 수평 성분의 힘을 나타낸다.
- Cdp: 압력 항력 계수(pressure drag coefficient)로서 물체 주위의 압력 차이로 인해 발생하는 항력 성분을 나타낸다.
- Cdf: 마찰 항력 계수(skin friction drag coefficient)로서 물체 표현의 마찰력 때문에 발생하는 항력 성분을 나타낸다.
- Cm: 피칭 모멘트 계수(pitching moment coefficient)로서 유체가 물체 주위로 흐를 때 물체 주위의 압력 분포의 차이로 인해 발생하는 회전력을 나타낸다.

따라서 본 실험에서 시뮬레이션의 입력변수는 총 4개(Thickness, MACH, AOA, RE)가 되고 이것이 예측모델의 독립변수가 된다. 한편 시뮬레이션의 출력변수는 총 5개(Cl, Cdt, Cdtp, Cdf, Cm)가 되며 이것이 예측모델의 종속변수가 된다.

KISTI가 제공한 EDISON KFLOW 시뮬레이션 결과 데이터는 총 7,680건이며, 본 실험에서는 이중 약 80%에 해당하는 6,200건의 데이터를 임의로 추출하여 예측모델 훈련 데이터로 사용하고, 나머지 약 20%에 해당하는 1,480건의 데이터를 훈련된 예측모델의 예측성능을 평가하는 테스트 데이터로 사용하였다. 따라서 훈련 데이터는 6,200건이고, 테스트 데이터는 1,490건이다. 예측모델을 훈련할 때는 10-겹 교차 검증(10-fold cross validation) 기법[21]을 사용하였다. 이 기법은 우선 훈련 데이터를 동일한 개수로 구성된 10개의 그룹으로 나눈다. 그리고 주어진 예측모델을 1개의 그룹을 제외한 9개의 그룹으로 훈련시키고 제외시킨 1개의 그룹으로 테스트하여 그의 예측성능을 측정한다. 이를 서로 다른 9개의 그룹에 대해 총 10번 반복하여 가장 좋은 성능을 보이는 예측모델을 선택한다. 훈련된 예측모델의 성능을 측정하기 위해서는 1,490건의 테스트 데이터를 사용하였다. 즉, 각 테스트 데이터 케이스에 대해 예측모델로 예측한 값과 참값을 비교하여 오차를 구한 후, 이들 오차의 평균값을 최종적인 예측성능의 지표로 사용하였다. 일반적으로 예측모델의 성능을 평가할 때는 RMSE(root-mean-square error)가 많이 사용되지만, 이 지표는 오차의 절대적인 크기를 나타내기 때문에 상대적인 크기를 잘 표현하지 못한다는 단점이 있다. 따라서 본 실험에서는 오차의 크기를 직관적으로 나타낼 수 있도록 참값에 상대적인 비율로 나타내는 % 오차(percent error)를 사용하였다. 다음 식은 % 오차의 정의이며 여기서 v_{true}는 참값을 $v_{predicted}$는 예측값을 나타낸다.

$$Error = \left| \frac{v_{true} - v_{predicted}}{v_{true}} \right| \times 100\%$$

또한 예측모델을 훈련하기 전에 필요에 따라 데이터의 정규화(normalization)를 수행하였다. 본 실험에서는 시뮬레이션 결과 데이터에 포함된 입력변수들의 값을 [0, 1] 범위의 값으로 정규화하였으며, 입력변수 X의 어떤 값 x의 정규화된 값 $x_{normalized}$를 구하기 위해 다음과 같은 식을 사용하였다. 여기서 X_{max}와 X_{min}는 각각 X의 최댓값과 최솟값을 의미한다.

$$x_{normalized} = \frac{x - X_{\min}}{X_{\max} - X_{\min}}$$

마지막으로 본 실험에서는 총 10개의 예측모델을 훈련시켜 예측성능을 비교하였으며, 각 예측모델은 R 라이브러리를 사용하여 구현되었다. 본 실험에서 비교대상으로 삼은 예측모델과, 해당 모델의 구현에 사용된 R 라이브러리는 각각 다음과 같다.

예측 모델	R 라이브러리
다중 선형 회귀 (Multiple linear regress)	lm {stats}
일반 가산 모델 (Generalized additive model)	gam {mgcv}
지지 벡터 기계 (Support vector machine)	svm {e1071}
분류 및 회귀 트리 (Classification and regression tree)	rpart {rpart}
랜덤 포레스트 (Random forest)	randomForest {randomForest}
일반 부스팅 모델 (General boosted model)	gbm {gbm}
다변수 적응 회귀 스플라인 (Multivariate adaptive regression spline)	earch {earth}
국소 회귀 (Local regression)	locfit {locfit}
k-최근접 이웃 회귀 (k-nearest neigbor regression)	knn.reg {FNN}
다층 신경망 (Mulilayer neural network)	h2o.deeplearning {h2o}

4.2 예측 모델 별 실험 결과

Fig. 4는 다중 선형 회귀의 예측 성능을 나타낸다. Cdp의 경우 평균 오차율 153%로 매우 나쁜 성능을 보였으며, 가장 성능이 좋은 Cdf의 경우에도 평균 오차율이 10.9%로 비교적 나쁜 성능을 보인다.

Fig. 5는 일반 가산 모델의 예측 성능을 나타낸다. 다중 선형 회귀와 유사하게 Cdp에 가장 나쁜 성능을, Cdf에 가장 좋은 성능을 보였다. 다중 선형 회귀의 확장된 모델이지만 다중 선형 회귀에 비해 비교적 적은 성능 향상에 그치고 있음이 관찰된다.

Fig. 6은 지지 벡터 기계 회귀의 예측 성능을 나타낸다. 2.1절에서 설명한 바와 같이 지지 벡터 기계 회귀에서는 다양한 커널(kernel) 함수를 사용할 수 있다. 본 실험에서는 3가지 커널 함수를 사용했을 때의 성능을 각각 측정하였으며, 실험 결과 방사(radial) 커널이 가장 좋은 성능을 보였다. 이 경우 다중 선형 회귀나 일반 가산 모델에 비해 훨씬 좋은 성능을 보인다.

Fig. 7은 분류 및 회귀 트리의 예측 성능을 나타낸다. 2.1절에서 설명한 바와 같이 분류 및 회귀 트리의 성능은 결정 트리의 노드 분할 기준이 되는 복잡도 매개변수(cp)에 따라 크게 달라진다. 본 실험에서는 cp의 값이 각각 0.01, 0.001, 0.0001일 때의 성능을 측정하였다. 일반적으로 노드를 더 상세히 분할할수록, 즉 cp값이 작을수록 예측 값이 더 정확해짐을 관찰할 수 있다. 본 실험에서는 cp=0.0001일 때지지 벡터 기계 회귀와 유사한 성능을 보인다.

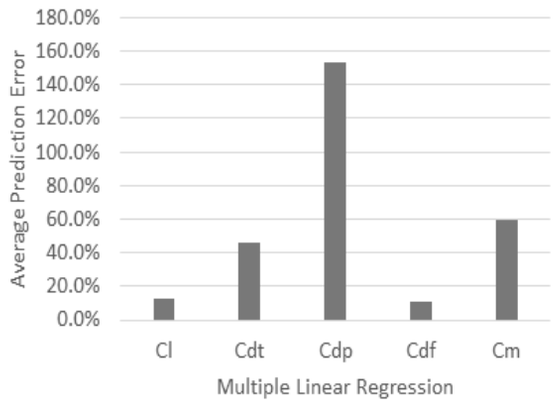


Fig. 4. Average Prediction Error of Multiple Linear Regression

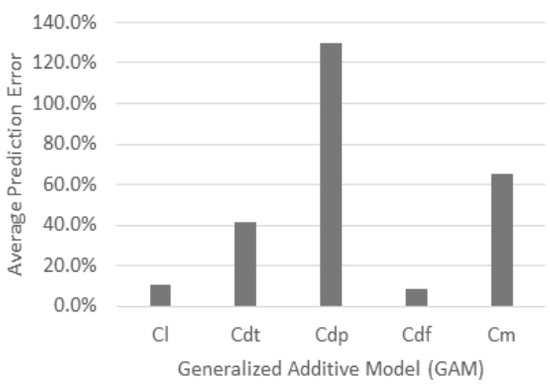


Fig. 5. Average Prediction Error of Generalized Additive Model (GAM)

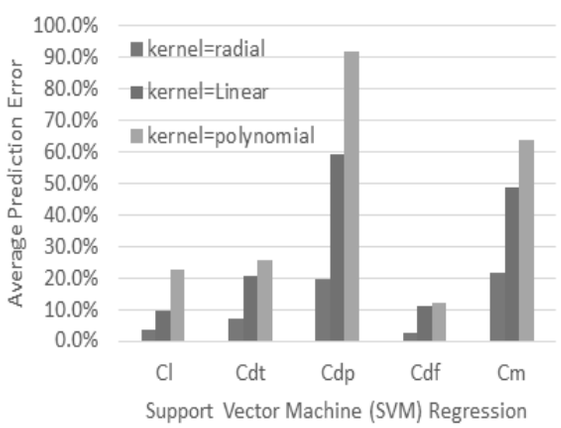


Fig. 6. Average Prediction Error of Support Vector Machine (SVM) Regression

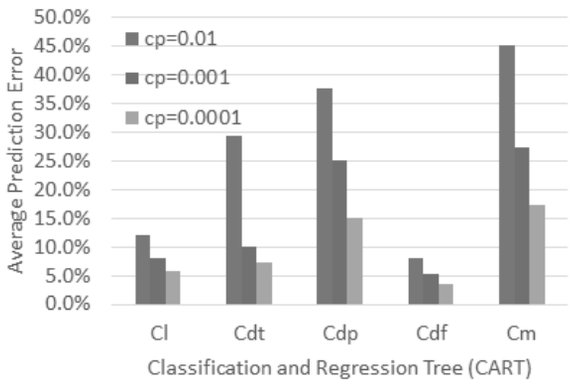


Fig. 7. Average Prediction Error of Classification and Regression Tree (CART)

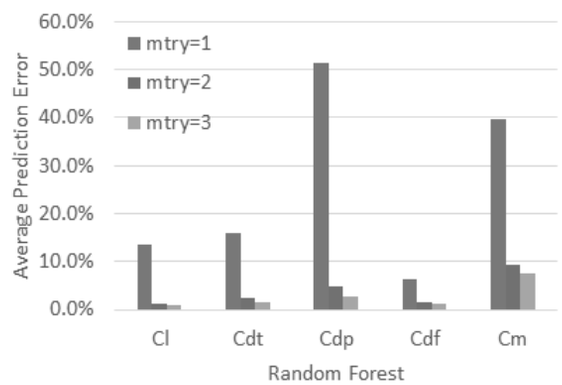


Fig. 8. Average Prediction Error of Random Forest

Fig. 8은 랜덤 포레스트의 예측 성능을 나타낸다. 2.1절에서 설명한 바와 같이 랜덤 포레스트는 노드의 분할을 결정하기 위해 임의로 선택하는 변수의 개수(mtry)에 따라 성능에 큰 차이를 보인다. 본 실험에서는 mtry의 값이 각각 1, 2, 3일 때의 성능을 측정하였으며, mtry = 3일 때 모든 예측 모델을 통틀어 가장 좋은 성능을 보였다. 다만 트리의 개수를 500개에서 5,000개로 증가시켜도 성능에 별 영향을 미치지는 않았다.

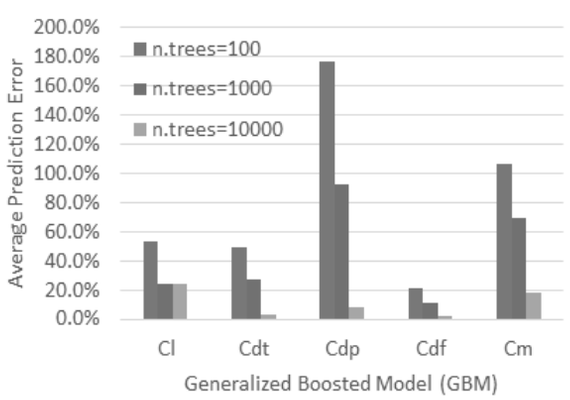


Fig. 9. Average Prediction Error of Generalized Boosted Model (GBM)

Fig. 9는 일반 부스팅 모델의 예측 성능을 나타낸다. 이 방법은 모델에서 더하는 결정 트리의 개수(n.trees)를 총 몇 개로 하느냐에 따라 성능이 크게 변한다. 본 실험에서는 n.trees = 100, 1,000, 10,000일 때의 성능을 각각 측정하였다. 본 실험에서 관찰된 바와 마찬가지로 트리의 개수가 증가하면 일반적으로 정확도가 증가한다. 하지만 다른 출력변수들과 달리 Cl에 대해서는 다른 예측 모델과 비교하여 가장 나쁜 성능을 보였다.

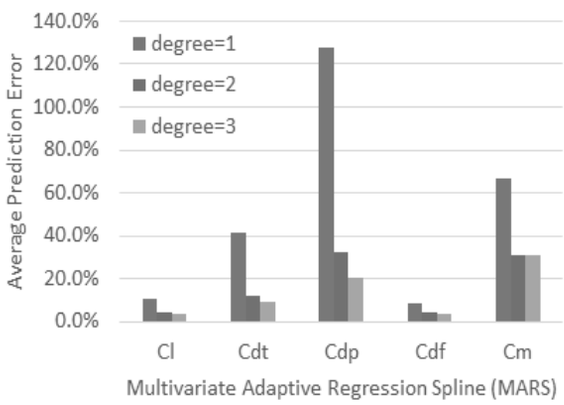


Fig. 10. Average Prediction Error of Multivariate Adaptive Regression Spline (MARS)

Fig. 10은 다변수 적응 회귀 스플라인의 예측 성능을 나타낸다. 2.1절에서 설명한 바와 같이 다변수 적응 회귀 스플라인은 어떤 기저 함수(degree)를 사용하느냐에 따라 성능이 달라진다. 본 실험에서는 기저 함수가 상수 1일 때(degree = 1), 경첩 함수일 때(degree = 2), 경첩 함수의 곱일 때(degree = 3)의 성능을 각각 측정했다. 상수 형태의 기저 함수(degree = 1)를 사용할 때보다 변수간의 상호작용을 많이 반영하는 기저 함수를 사용할수록(degree = 3) 성능이 향상됨을 관찰할 수 있었다. 하지만 출력변수 Cm에 대해서는 다른 예측 모델들에 비해 비교적 나쁜 성능을 보였다.

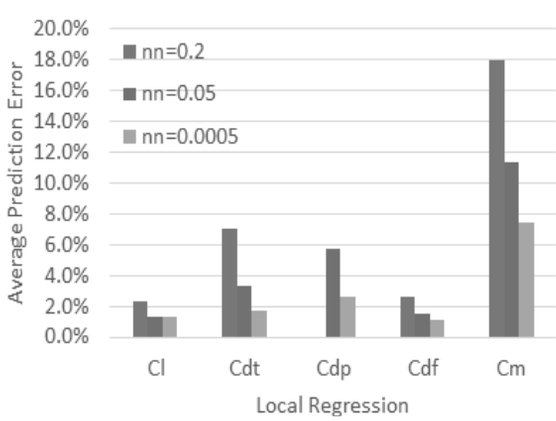


Fig. 11. Average Prediction Error of Local Regression

Fig. 11은 국소 회귀의 예측 성능을 나타낸다. 국소 회귀는 주어진 데이터에 대해 그 주변의 데이터만 고려해서 근사 다항식을 구하기 때문에 전체 데이터 중 주변 데이터 몇 %까지만 볼 것이냐(nn)에 따라 성능이 크게 변한다. 본 실험에서는 nn의 값이 낮을수록 (즉, 가까운 데이터만 고려할수록) 더 좋은 성능을 보였으며, 작은 nn의 값에 대해서는 모든 예측 모델 중 최고 수준의 성능을 보인다. 이것은 주어진 시뮬레이션 데이터가 전체적으로 매우 변동이 큰 데이터임을 의미한다.

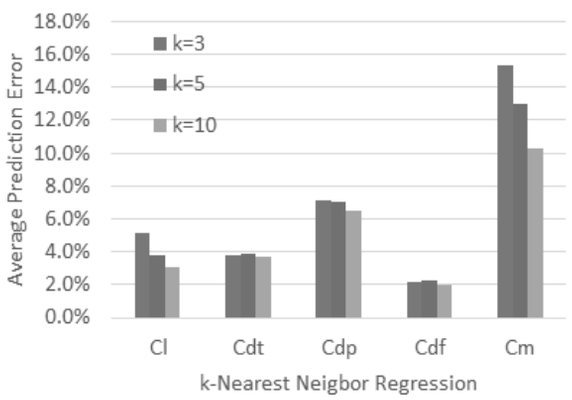


Fig. 12. Average Prediction Error of *k*-Nearest Neighbor Regression

Fig. 12는 *k*-최근접 이웃 회귀의 예측 성능을 나타낸다. 이 방법도 국소 회귀와 유사하게 주변 데이터만 고려하지만 회귀 다항식을 만들지 않고 단순 평균값을 구한다는 차이가 있다. 이 방법도 국소 회귀와 마찬가지로 최근접 이웃을 몇 개까지 사용하느냐(*k*)에 따라 성능이 변한다. 본 실험에서는 k = 10일 때 가장 좋은 성능을 보였으며, k가 너무 크거나 너무 적으면 오히려 성능이 저하된다. 이 방법도 전체 예측 모델 중 가장 성능이 좋은 편에 속하며, 그 이유는 국소 회귀와 동일하다.

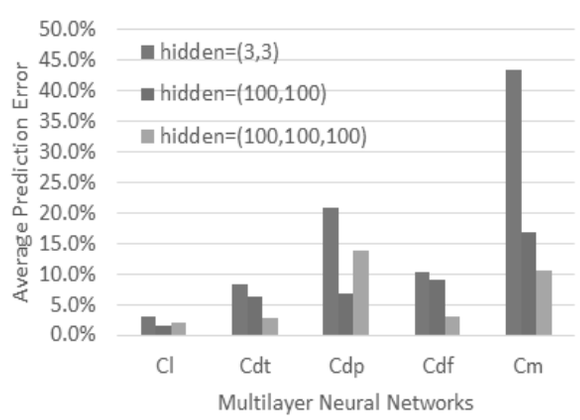


Fig. 13. Average Prediction Error of Multilayer Neural Network

Fig. 13은 다층 신경망의 예측 성능을 나타낸다. 2.1절에서 설명한 바와 같이 다층 신경망의 성능은 층 및 뉴런의 개수에 따라 크게 좌우된다. 하지만 일반적으로 층과 뉴런의 개수가 무조건 많을수록 좋은 것은 아니며, 너무 많으면 과적합(overfitting)이 발생하거나 훈련 비용이 과도하게 커질 수 있다. 본 실험에서는 은닉층의 개수가 2개 또는 3개, 각 층의 뉴런의 수가 3개 또는 100개일 때의 성능을 각각 측정했다. 본 실험에서는 은닉층이 3개이고 각 층의 뉴런 개수가 100개씩일 때(hidden=(100,100,100)) 가장 좋은 성능을 보였으며, 여기서 층의 개수나 뉴런을 더 증가시켜도 성능이 더 향상되지는 않았다. 이때의 성능은 모든 예측 모델들 중 비교적 좋은 성능을 보이는 편에 속한다.

4.3 예측 모델 간 성능 비교

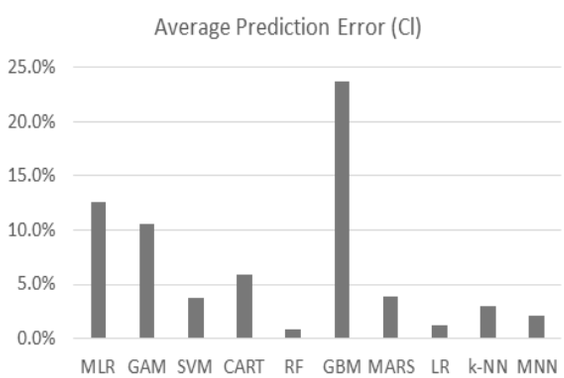


Fig. 14. Performance Comparison for Cl

Fig. 14는 출력변수 Cl에 대한 10개 모델의 예측 성능을 비교한 결과이다. 각 모델은 약자로 표현하였으며, 좌측부터 차례대로 다중 선형 회귀(MLR), 일반 가산 모델(GAM), 지지 벡터 기계(SVM), 분류 및 회귀 트리(CART), 랜덤 포레스트(RF), 일반 부스팅 모델(GBM), 다변수 적응 회귀 스플라인(MARS), 국소 회귀(LR), *k*-최근접 이웃 회귀(k-NN), 다층 신경망(MNN)을 나타낸다. 각 모델의 예측 성능은 앞서 기술한 실험 결과에서 가장 성능이 좋은 경우를 사용하였다. Cl에 대해서는 랜덤 포레스트가 가장 좋은 성능을 나타냈으며, 국소 회귀도 좋은 성능을 보였다. 반면에 선형 모델에 기반한 방법(MLR, GAM, GBM)들은 좋지 않은 성능을 보임을 알 수 있다.

Fig. 15는 출력변수 Cdt에 대한 10개 모델의 예측 성능을 비교한 결과이다. 이 경우도 Fig. 14와 유사하게 랜덤 포레스트와 국소 회귀가 가장 좋은 성능을 보였으며 다중 선형 회귀와 일반 가산 모델은 나쁜 성능을 보였다. *k*-최근접 이웃 회귀와 다층 신경망도 Fig. 14에서와 유사하게 비교적 좋은 성능을 보이고 있다.

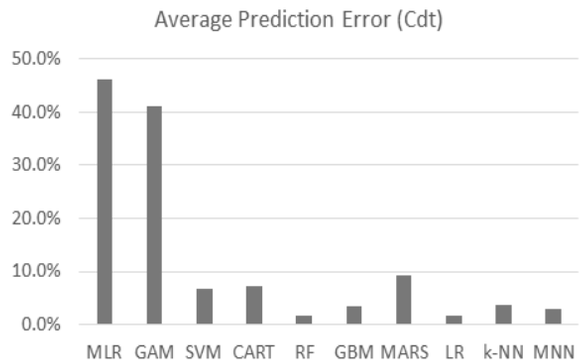


Fig. 15. Performance Comparison for Cdt

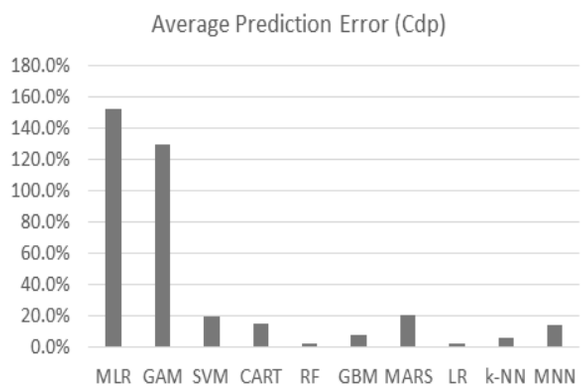


Fig. 16. Performance Comparison for Cdp

Fig. 16은 출력변수 Cdp에 대한 10개 모델의 예측 성능을 비교한 결과이다. 역시 이 결과에서도 랜덤 포레스트와 국소 회귀가 가장 좋은 성능을 보였으며, k-최근접 이웃 회귀와 다층 신경망이 그 다음으로 좋은 성능을 보였다. 선형 모델에 기반한 방법(MLR, GAM)은 이 경우에도 좋지 못한 성능을 보였다. 이것은 지금까지의 출력 변수들이 선형 모델로 표현하기에 적합하지 않다는 것을 의미한다.

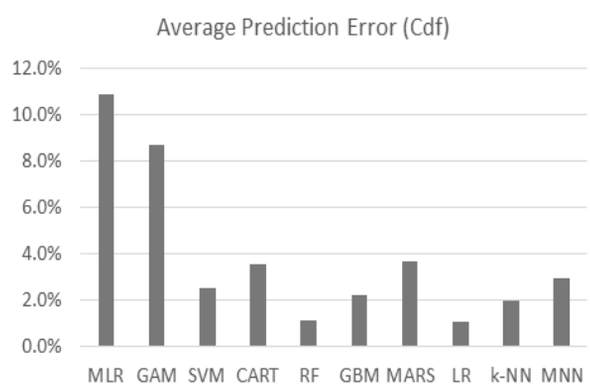


Fig. 17. Performance Comparison for Cdf

Fig. 17은 출력변수 Cdf에 대한 10개 모델의 예측 성능을 비교한 결과이다. Cdf는 지금까지의 출력변수들과 달리 간단한 선형 모델로도 비교적 잘 모델링되는 변수인 것으로 파악된다. 따라서 모든 모델이 다른 출력변수들에 비해 비교적 좋은 성능을 보이고 있다. 하지만 이 경우에도 랜덤 포레스트와 국소 회귀가 가장 좋은 성능을 보이고, k-최근접 이웃 회귀와 다층 신경망이 그 다음으로 좋은 성능을 보이며, 다중 선형 회귀와 일반 가산 모델이 가장 나쁜 성능을 보이는 패턴을 보인다.

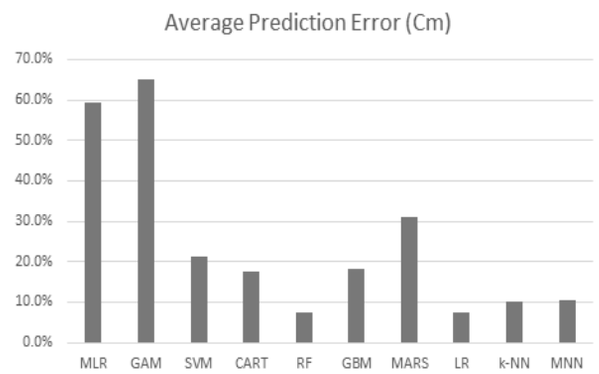


Fig. 18. Performance Comparison for Cm

Fig. 18은 출력변수 Cm에 대한 10개 모델의 예측 성능을 비교한 결과이다. Cm은 값의 분포가 매우 불규칙해서 다른 출력변수에 비해 모든 예측 모델의 성능이 가장 떨어지는 현상이 관찰된다. 가장 좋은 성능을 보이는 랜덤 포레스트와 국소 회귀도 대략 7.4~7.5% 정도의 비교적 큰 평균 오차율을 보였다. 지금까지의 결과를 보면 시뮬레이션 출력변수에 따라 모델링이 비교적 쉽게 되는 변수와 그렇지 않은 변수가 혼재함을 알 수 있다. 따라서 변수에 따라 예측 모델을 세워 예측 성능을 평가한 뒤, 성능에 따라 사용자에게 예측 기능의 사용을 추천하거나 비추천하는 것이 바람직하다고 판단된다.

4.4 실험 결과 요약

4.2절과 4.3절을 통해 다양한 예측 모델들의 예측 성능 평가 결과를 살펴보았다. 그 결과 예측 모델들의 예측 성능에 대해 다음과 같은 사항들을 관찰할 수 있다.

- 대부분의 출력 변수들은 값의 분포가 전체적으로 어떤 패턴을 보이기보다는 국소적으로 매우 불규칙한 경우가 많으며, 이 경우 예측하고자 하는 데이터의 주변 데이터만을 활용하는 근접 이웃 기반의 회귀 방법이 비교적 좋은 성능을 보인다. 근접 이웃 기반의 회귀 방법에는 국소 회귀와 k-최근접 이웃 회귀가 포함되며, 대

부분의 시뮬레이션 결과 예측 시 다른 예측 모델들에 비해 비교적 좋은 성능을 내는 기법이라 예상된다.

- 상기 이유로, 전체 훈련 데이터를 모두 포괄하는 단 하나의 예측 함수를 찾아내는 방식의 회귀 방법 (다중 선형 회귀, 일반 가산 모델, 지지 벡터 기계, 다변수 적응 회귀 스플라인 등)은 비교적 좋지 않은 성능을 보인다. 특히 넓은 범위의 입력변수 범위에 대해 예측 모델이 필요한 경우, 이러한 방식의 회귀 방식은 더욱 좋지 않은 성능을 낼 것으로 예상된다. 다만, 지지 벡터 기계는 커널을 사용하여 더욱 복잡한 모델을 만들 수 있고, 다변수 적응 회귀 스플라인은 비선형성과 변수 간의 상호작용을 모델에 추가하는 기법을 추가로 사용하므로 성능의 저하가 비교적 덜하다.
- 일반적으로 결정 트리 기반의 회귀 기법은 최종 예측값을 단 1개의 결정 트리로 결정하는 방법(분류 및 회귀 트리)보다 많은 수의 결정 트리를 활용하는 방법(랜덤 포레스트, 일반 부스팅 모델)이 더 좋은 성능을 보인다. 또한 많은 수의 결정 트리를 사용하더라도 어떤 시점 이상에서는 트리의 수를 증가시키는 보다는 각 트리의 깊이(depth)를 더 깊게 만들수록 더 정확한 결과를 보인다.
- 다층 신경망의 경우 무조건 층을 늘리거나 뉴런의 수를 늘리는 것이 성능에 도움이 되지 않는다. 일반적으로 최적의 층이나 뉴런의 수를 찾는 방법은 알려져 있지 않지만, 보통 2~3개의 은닉층을 사용하면 아무리 복잡한 함수라도 표현이 가능하다고 알려져 있다.

5. 결 론

본 논문에서는 사용자가 요청한 시뮬레이션을 실제 수행하지 않고도 통계적 기계학습 방법을 사용하여 그 결과를 빠르게 예측할 수 있는 시스템을 개발하였다. 본 논문에서 개발한 시뮬레이션 서비스 시스템은 기존 시스템과는 달리 이미 수행이 완료된 시뮬레이션 결과 데이터를 데이터베이스에 저장하고, 이를 예측 모델의 학습을 위한 훈련 데이터로 활용한다. 여러 사용자가 다수의 시뮬레이션을 동시에 요청하는 경우, 이를 모두 수행하는 것은 시스템에 큰 부담이 된다. 이 경우 본 논문에서 개발한 시뮬레이션 결과 예측 시스템을 활용하면 시스템에 큰 부담을 주지 않고도 사용자는 다양한 입력변수의 값에 대해 시뮬레이션 결과를 빠르게 예측해 볼 수 있다. 물론 모든 데이터에 대해 항상 최고의 성능을 내는 예측 방법은 존재하지 않으므로, 본 시스템은 R과 연동하여 다양한 예측 모델을 제공한다. 따라서 이후 새로운 R 패키지가 개발되면 이를 손쉽게 시스템의 기능에 포함시킬 수 있다. 추후 연구로는 최고의 예측성능을 보이는 예측 모델은 시뮬레이션 데이터에 따라 서로 다르므로, 이미 누적된 시뮬레이션 데이터로 여러 예측 모델을 훈련시켜 각 예측 모델의 예측성능을 예측하고, 이를 기반으로 사용자에게 최선의 예측 모델과 예상되는 오차의 크기를 자동으로 제공하는 것을 목표로 한다.

참 고 문 헌

[1] Angela B. Shiflet and George W. Shiflet, “Introduction to Computational Science: Modeling and Simulation for the Sciences,” 2nd edition, Princeton University Press, 2014.

[2] Y.-K. Suh, et. al., “EDISON: A Web-based HPC Simulation Execution Framework for Large-scale Scientific Computing Software,” in *Proc. of CCGrid'16*, pp.608-612, May 2016.

[3] R: The R Project for Statistical Computing [Internet], https://www.r-project.org/.

[4] PhET [Internet], https://phet.colorado.edu/.

[5] ALF: Simulating Genome Evolution [Internet], http://alfsim.org/.

[6] BiDaS [Internet], http://bioserver-3.bioacademy.gr/Bioserver/BiDaS/.

[7] WebArrayDB [Internet], http://www.webarraydb.org/webarray/.

[8] Cipran Docan, Manish Parashar, and Scott Klasky, “DataSpaces: an interaction and coordination framework for coupled simulation workflows,” *Cluster Computing*, Vol.15, No.2, pp.163-181, 2012.

[9] Adam Hospital, Pau Andrio, Cesare Cugnasco, Laia Codo, Yolanda Becerra, Pablo D. Dans, Federica Battistini, Jordi Torres, Ramon Goni, Modesto Orozco, and Josep Ll. Gelpı, “BIGNASim: a NoSQL database structure and analysis portal for nucleic acids simulation data,” *Nucleic Acids Research*, Vol.44, 2016.

[10] D. Mishin, D. Medvedev, A. S. Szalay, R. Plante, and M. Graham, “Data Sharing and Publication Using the SciDrive Service,” *Astronomical Data Analysis Software and Systems*, Vol.485, 2014.

[11] Anand Kumar, Vladimir Grupcev, Meryem Berrada, Joseph C. Fogarty, Yi-Cheng Tu, Xingquan Zhu, Sagar A Pandit, and Yuni Xia, “DCMS: A data analytics and management system for molecular simulation,” *Journal of Big Data*, Vol.1, No.9, 2014.

[12] Julien C. Thibault, Julio C. Facelli, and Thomas E. Cheatham, III, “iBIOMES: Managing and Sharing Biomolecular Simulation Data in a Distributed Environment,” *Journal of Chemical Information and Modeling*, Vol.53, pp.726-736, 2013.

[13] Jian Huang, Xuechen Zhang, Greg Eisenhauer, Karsten Schwan, Matthew Wolf, Stephane Ethier, and Scott Klasky, “Scibox: Online Sharing of Scientific Data via the Cloud,” in *Proceedings of the 28th IEEE International Parallel & Distributed Processing Symposium*, pp.145-154, 2014.

[14] Ki Yong Lee, Yoonjae Shin, Yeonjeong Choe, Young-kyoon Suh, Jeonghwan Sa, and Kum Won Cho, “Design of a Simulation Data Management System for Efficient Computational Science and Engineering Simulations,” in *Proc. of KIPS Spring Conference*, April, 2016.

[15] Ki Yong Lee, Yoonjae Shin, Yeonjeong Choe, SeonJeong Kim, Young-kyoon Suh, Jeonghwan Sa, and Kum Won Cho, “Design and Implemenation of a Data-Driven Simulation Service System,” in *Proc. of 6th International Conference on Emerging Databases (EDB 2016)*, October, 2016.

[16] MongoDB [Internet], https://www.mongodb.com/.

[17] Node.js [Internet], https://nodejs.org/.

[18] rJava [Internet], https://www.rforge.net/rJava/.

[19] EDISON-CFD, [Internet], https://cfd.edison.re.kr/.

[20] NACA airfoil [Internet], https://en.wikipedia.org/wiki/NACA_airfoil/.

[21] T. Hastie, R. Tibshirani, and J. Friedman, “The Elements of Statistical Learning,” 2nd Edition, Springer, 2008.

이 기 용

e-mail : kiyonglee@sookmyung.ac.kr
1998년 KAIST 전산학과(학사)
2000년 KAIST 전산학과(석사)
2006년 KAIST 전산학과(박사)
2006년~2008년 삼성전자 책임연구원
2008년~2010년 KAIST 전산학과 연구조교수
2010년~현 재 숙명여자대학교 컴퓨터과학부 부교수
관심분야 : 데이터베이스, 데이터마이닝, 데이터스트림, 빅데이터

신 윤 재

e-mail : yoonjaeshin@sookmyung.ac.kr
2015년 숙명여자대학교 컴퓨터과학부 (학사)
2015년~현 재 숙명여자대학교 컴퓨터과학부 석사과정
관심분야 : 데이터베이스, 데이터마이닝

최 연 정

e-mail : cyj@sookmyung.ac.kr
2015년 숙명여자대학교 컴퓨터과학부 (학사)
2015년~현 재 숙명여자대학교 컴퓨터과학부 석사과정
관심분야 : 데이터베이스, 배열데이터

김 선 정

e-mail : sjkim@sookmyung.ac.kr
2016년 숙명여자대학교 컴퓨터과학부 (학사)
2016년~현 재 숙명여자대학교 컴퓨터과학부 연구원
관심분야 : 데이터베이스

서 영 균

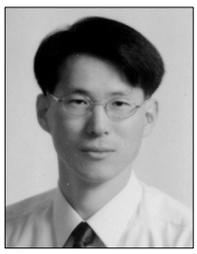
e-mail : yksuh@kisti.re.kr
2015년 Univ. of Arizona(박사)
2005년~현 재 한국과학기술정보연구원 (KISTI) 슈퍼컴퓨팅융합연구센터 선임연구원
관심분야 : 데이터베이스 시스템/설계, 컴퓨팅의 과학, 빅데이터, HPC

사 정 환

e-mail : sa_c@kisti.re.kr
20015년 건국대학교 항공우주정보공학과 (박사)
2015년~현 재 한국과학기술정보연구원 (KISTI) 슈퍼컴퓨팅융합연구센터 박사후연구원
관심분야 : 계산과학, 항공우주, 유체해석

이 종 숙

e-mail : jsruthlee@kisti.re.kr
2000년 Univ. of Canterbury(박사)
2000년~현 재 한국과학기술정보연구원 (KISTI) 슈퍼컴퓨팅융합연구센터 책임연구원(실장)
관심분야 : 계산과학, HPC, 이공계 교육・연구 융합



조 금 원

e-mail : ckw@kisti.re.kr
2000년 KAIST 항공우주공학과(박사)
2000년~현 재 한국과학기술정보연구원 (KISTI) 슈퍼컴퓨팅융합연구센터 책임연구원(센터장)
관심분야 : 계산과학, 항공우주, 유체해석, 이공계 교육・연구 융합