https://doi.org/10.7840/kics.2023.48.6.733

기계학습의 하이퍼파라미터 최적화 연구 동향

원 종 현◆, 신 종 민*, 김 재 호**, 이 장 원°

A Survey on Hyperparameter Optimization in Machine Learning

Jonghyeon Won◆, Jongmin Shin*, Jae-Ho Kim**, Jang-Won Lee°

요 약

최근 다양한 분야에서 주목받고 있는 기계학습의 성능은 기계학습 모델의 하이퍼파라미터에 의존한다. 이에 따라 기계학습의 성능을 향상시키기 위해서는 최적의 하이퍼파라미터를 찾는 것이 중요하다. 기계학습의 하이퍼파라미터 최적화는 최적화 문제의 목적 함수와 결정 변수들의 특징 때문에 풀이에 어려움이 많으며, 하이퍼파라미터 최적화의 알고리즘들은 이러한 어려움을 해결하는 방향으로 연구되고 있다. 본 논문에서는 기계학습의 하이퍼파라미터 최적화의 난점을 분석하고, 이를 해결하기 위해 제안된 하이퍼파라미터 최적화 연구들의 동향을 파악한다. 또한 이를 바탕으로 기계학습의 성능을 더 향상시키기 위해 앞으로 기계학습의 하이퍼파라미터 최적화 연구가 나아갈 방향을 제시한다.

키워드 : 기계학습, 하이퍼파라미터, 하이퍼파라미터 최적화
Key Words : Machine learning (ML), hyperparameter, hyperparameter optimization (HPO)

ABSTRACT

Machine learning (ML) has recently attracted attention in various fields and the performance of an ML model highly depends on its hyperparameters. Accordingly, it is important to find the optimal hyperparameters to improve its performance. However, there are many difficulties in hyperparameter optimization (HPO) due to some characteristics of its objective functions and decision variables and lots of studies on algorithms for HPO have been conducted to solve those difficulties. In this paper, we analyze the difficulties of HPO in ML and present the trends of recent studies on HPO. In addition, we present a future direction for HPO in ML to further improve the performance of ML.

Ⅰ. 서 론

최근 인공 지능 분야에서 많은 주목을 받고 활발하게 연구되고 있는 기계학습(machine learning)은 주어진 데이터를 이용하여 특정 목적에 대한 예측이나 결정의 성능을 향상시키는 알고리즘에 대한 연구이다. 이렇게 연구된 기계학습은 통신, 금융, 영업, 의료, 제조 등의 다양한 산업 분야에 활용되어 이미지 분류, 불량품 탐지, 질병 예측, 음성 인식, 번역 등 결과 예측이나 합리적 결정 문제에서 우수한 성능을 보이고 있다[1,2]. 기계학습에는 선형 회귀(linear regression), 선형 분류(linear classification), 최근접 이웃 탐색

※ 이 연구는 2023년도 산업통상자원부 및 산업기술평가관리원(KEIT) 연구비 지원에 의한 연구임(RS-2022-00154678)
◆ First Author : Yonsei University Department of Electrical and Electronic Engineering, wonjonghyeon@yonsei.ac.kr, 정회원
° Corresponding Author : Yonsei University Department of Electrical and Electronic Engineering, jangwon@yonsei.ac.kr, 종신회원
* Yonsei University Department of Electrical and Electronic Engineering, tls0264@yonsei.ac.kr, 학생회원
** Sejong University Department of Electrical Engineering, kimjh@sejong.ac.kr
논문번호 : KICS2023-02-026-C-RN.R1, Received February 12, 2023; Revised March 27, 2023; Accepted April 3, 2023

(nearest neighbor search), k-평균 군집화(k-means clustering), 신경망(neural network) 등 다양한 모델이 존재하며[3], 주어진 데이터의 형태와 해결할 문제의 목적에 따라 적절한 모델을 선택하여 이용할 수 있다.

기계학습 모델이 특정 결과를 예측하거나 특정 문제의 결정을 내리기 위해서는 주어진 데이터를 이용하여 적절한 학습 파라미터(learning parameter) 값을 찾는 훈련(training) 과정이 필수적이다. 기계학습의 학습 파라미터와 달리, 기계학습 모델의 구조나 알고리즘 등을 구체적으로 결정하는 변수를 하이퍼파라미터(hyperparameter)라고 하며, 하이퍼파라미터는 훈련 과정 전에 값이 미리 결정되어 있어 훈련 과정에서 값이 변하지 않는다. 이러한 기계학습의 하이퍼파라미터는 크게 모델 하이퍼파라미터와 알고리즘 하이퍼파라미터로 분류할 수 있다. 모델 하이퍼파라미터는 기계학습 모델 자체의 구조를 결정하는 값으로, 신경망에서 은닉 층(hidden layer)의 개수와 각 은닉 층의 노드 개수, 합성곱 신경망(convolutional neural network)에서 kernel의 크기 등이 있다. 알고리즘 하이퍼파라미터는 기계학습의 학습 알고리즘을 결정하는 값으로, 경사하강법(gradient descent)에서의 학습률(learning rate), 반복(iteration) 횟수 등이 있다.

기계학습에서는 훈련에 앞서 하이퍼파라미터 값을 결정하여 구체적인 기계학습 모델을 생성하고, 이 기계학습 모델을 이용하여 학습(learning)을 진행한다. 각 하이퍼파라미터는 서로 다른 기계학습 모델을 결정하므로 하이퍼파라미터 값에 따라 기계학습 모델의 성능이 달라지며, 하이퍼파라미터 값을 적절하게 설정하지 않으면 기계학습 모델은 매우 낮은 성능을 보일 수 있다. 따라서 높은 성능을 가진 기계학습 모델을 구현하기 위해서는 최적의 하이퍼파라미터를 결정하여야 한다[4].

하이퍼파라미터의 성능을 측정하기 위해서는 각 하이퍼파라미터에 대응되는 기계학습 모델을 학습시켜야 하는데, 일반적으로 학습에는 많은 시간이 필요하다. 이에 따라 가능한 모든 하이퍼파라미터 값들의 성능을 측정하여 최적의 하이퍼파라미터를 찾는 것은 현실적으로 거의 불가능하다. 또한 하이퍼파라미터는 연속형(continuous), 이산형(discrete), 범주형(categorical) 등 그 종류가 매우 다양하고, 각각의 하이퍼파라미터마다 특징도 매우 다양하다. 이렇게 하이퍼파라미터의 최적화(HPO: hyperparameter optimization)에는 여러 난점이 존재하고, HPO의 알고리즘은 이러한 난점을 해결하기 위한 방향으로 연구되고 있다[5].

한편, 최근 인공 지능 분야에서는 기계학습 모델 중 하나인 신경망의 많은 장점들이 부각되고 그것의 성능 우수성이 검증되면서 심층 신경망(deep neural network), 합성곱 신경망, 순환 신경망(RNN: recurrent neural network) 등 신경망 기반의 많은 기계학습 모델들이 제안되었다. 이러한 모델들은 매우 복잡한 구조를 가지고 있어 일반적으로 그 구조를 결정하는 모델 하이퍼파라미터가 매우 많다. 따라서 HPO 중에서 특별히 신경망 기반 기계학습 모델의 성능 향상을 위해 신경망의 모델 하이퍼파라미터를 최적화하는 연구가 주목받고 있으며, 신경망 구조를 결정하기 위한 HPO 알고리즘을 neural architecture search (NAS)라고 부른다[6].

현재까지도 HPO의 효율적인 알고리즘 개발을 위한 많은 연구들이 진행되고 있다[4,5,7,8]. [4]에서는 NAS를 포함하여 대표적인 HPO 알고리즘들을 소개하고 있다. 각 알고리즘들의 기반이 되는 기법들을 Bayesian optimization (BO), 진화 알고리즘(evolutionary algorithm), 강화 학습(RL: reinforcement learning), 경사하강법 등으로 정리하고 이 기준에 따라 알고리즘들을 분류하였다. [5]에서는 선형 회귀, 최근접 이웃 탐색, Naíve Bayes, ensemble learning, clustering 등 대표적인 기계학습 모델들을 소개하고 각각의 모델에 포함된 하이퍼파라미터를 소개하였다. 이후 여러 HPO 알고리즘들의 강점과 한계점을 분석하여, 각각의 HPO 알고리즘을 적용하기 적합한 기계학습 모델들을 정리하였다. [7]에서는 HPO 알고리즘의 일반적인 절차를 하이퍼파라미터 탐색 과정과 성능 평가(performance evaluation) 과정으로 구분하고, 각 과정의 기반이 되는 알고리즘에 따라 HPO 알고리즘들을 분류하였다. 한편 기계학습 모델의 성능을 평가하기 위한 지표는 precision, mean square error (MSE), complexity, 훈련 시간 등으로 매우 다양하며 이러한 여러 지표들을 동시에 목적 함수로 고려한 HPO 연구들이 진행되고 있다. [8]에서는 이러한 다목적 HPO에 대한 연구들을 소개하고 있다.

HPO 문제의 특성상 여러 난점들이 존재하여 HPO의 알고리즘들은 이러한 난점들을 해결하기 위한 방향으로 개발되고 있지만, [4,5,7,8]에서는 각 알고리즘이 해결하고자 한 난점에 주목하고 있지 않다. 따라서 본 논문에서는 이러한 HPO 문제의 난점을 분석하고 이에 따라 HPO의 알고리즘들을 소개하고자 한다. HPO 알고리즘의 최신 연구 동향을 소개하기에 앞서 본 논문에서는 먼저 HPO의 난점을 분석한다. 이를 위해 HPO를 수식으로 표현하고 그 특징을 살펴본다. 수

식화된 최적화 문제(optimization problem)를 바탕으로 HPO의 난점을 분석하며, 각 알고리즘에서 주목한 HPO의 난점에 따라 제안된 알고리즘들을 분류한다. 최종적으로 이렇게 분류된 알고리즘들을 상세하게 분석하며 연구 동향을 파악한다.

본 논문의 구성은 다음과 같다. 2장에서는 HPO를 일반적인 최적화 문제로 형성하고 HPO의 난점을 소개한다. 3장에서는 현재까지 연구된 HPO의 알고리즘들을 2장에서 언급한 난점에 따라 분류하고 알고리즘의 내용을 설명한다. 4장에서는 HPO의 연구가 앞으로 나아갈 방향을 제시하고, 5장에서 논문의 결론을 요약한다.

Ⅱ. 하이퍼파라미터 최적화

본 장에서는 기계학습에서 HPO의 일반적인 과정을 소개하고, 이를 최적화 문제의 형태로 표현한다. 그후, 이 최적화 문제를 바탕으로 일반적으로 HPO의 최적해를 구하기 어려운 이유들을 분석한다.

2.1 하이퍼파라미터 최적화 문제

주어진 기계학습 모델에 대해 그 기계학습 모델의 하이퍼파라미터가 가질 수 있는 모든 값들의 집합을 탐색 공간(search space)이라고 한다. 탐색 공간의 원소들은 구체적인 하나의 기계학습 모델을 결정하며, 주어진 훈련 데이터와 검증(validation) 데이터에 대해 각 기계학습 모델은 하나의 성능(performance) 값에 대응된다. 즉, 탐색 공간의 원소 하나하나는 각 기계학습의 성능 값에 대응할 수 있다. 각 하이퍼파라미터 값의 기계학습 모델 성능을 평가하는 데에는 훈련 데이터와 검증 데이터를 통해 그 기계학습 모델을 훈련시키는 과정이 필수적인데, 일반적으로 기계학습의 훈련 과정에는 시간이 많이 소요된다. 그림 1은 위에서 설명한 하이퍼파라미터의 성능 평가 과정을 보여준다.

HPO는 탐색 공간 내에서 가장 좋은 성능 값을 가지는 하이퍼파라미터를 찾는 과정이라고 볼 수 있다. 따라서 HPO는 문제 (1)과 같이 목적 함수와 결정 변수를 포함한 최적화 문제의 형태로 표현할 수 있다.

$$\begin{aligned}
&\underset{\boldsymbol{\Theta}}{\text{minimize}} && J(\boldsymbol{\Theta}|\mathbf{X}_{\text{tr}},\mathbf{X}_{\text{va}}) && (1.1)\\
&\text{subject to} && m_1 \le \theta_1 \le M_1 && (1.2)\\
&&& \theta_2 \in \mathbb{N}, && (1.3)\\
&&& \theta_3 \in \{c_1, c_2, (c_3, \lambda_{3,3}), \cdots\}, && (1.4)\\
&&& \lambda_{3,3} > m_{3,3} && (1.5)\\
&&& \vdots
\end{aligned}$$

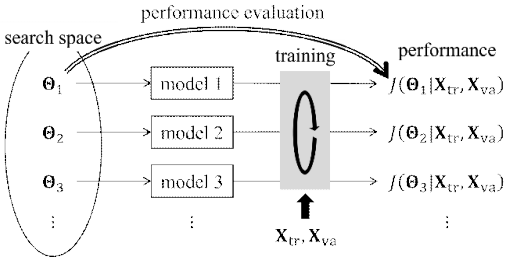

그림 1. 하이퍼파라미터 성능 평가
Fig. 1. Performance evaluation for hyperparameter.

식 (1.1)에서 $J(\cdot\,|\,\cdot,\cdot)$는 비용 함수(cost function), $\boldsymbol{\Theta}$는 하이퍼파라미터, $\mathbf{X}_{\text{tr}}$는 훈련 데이터, $\mathbf{X}_{\text{va}}$는 검증 데이터이다.

문제 (1)의 목적 함수인 $J(\boldsymbol{\Theta}|\mathbf{X}_{\text{tr}},\mathbf{X}_{\text{va}})$ 는 기계학습의 성능을 나타내는 비용 함수로, 주어진 훈련 데이터 $\mathbf{X}_{\text{tr}}$과 검증 데이터 $\mathbf{X}_{\text{va}}$에 대해 하이퍼파라미터 $\boldsymbol{\Theta}$에 관한 함수로 표현된다. 비용 함수의 예시로는, 회귀 문제에서의 평균 제곱 오차(mean squared error), 분류 문제에서의 교차 엔트로피(cross entropy) 등이 있다. 문제 (1)에서는 비용 함수를 최소화하는 것을 목적으로 한다.

문제 (1)의 결정 변수인 $\boldsymbol{\Theta} = (\theta_1, \theta_2, \theta_3, \cdots)$는 최적화 문제를 통해 최적의 값을 찾고자 하는 하이퍼파라미터들의 순서쌍이다. 하이퍼파라미터는 값의 범위에 따라 연속형, 이산형, 범주형으로 분류할 수 있다. 연속형 하이퍼파라미터는 실수 집합의 특정 구간 내의 값을 범위로 가진다. 예를 들어, 학습률은 양의 실수 값을 가질 수 있으므로 연속형 하이퍼파라미터이다. 이산형 하이퍼파라미터는 이산적인 실수 값들을 범위로 가지는 하이퍼파라미터이다. 예를 들어, 신경망의 은닉 층의 개수는 음이 아닌 정수만을 값으로 가질 수 있으므로 이산형 하이퍼파라미터이다. 연속형 하이퍼파라미터와 이산형 하이퍼파라미터는 모두 수치형 값을 가질 수 있기 때문에 둘을 통틀어서 수치형 하이퍼파라미터라고도 한다. 이와 달리, 범주형 하이퍼파라미터는 수치로 표현되지 않는 대상들을 값으로 갖는 하이퍼파라미터이다. 예를 들어, 신경망 노드의 활성 함수(activation function)가 이에 해당한다. 활성 함수는 rectified linear unit (ReLU) 함수, sigmoid 함수, hyperbolic tangent 함수 등이 될 수 있으며 이들은 모두 수치형 값이 아니므로, 활성 함수는 범주형 하이퍼파라미터이다. 한편, 기계학습에는 특정 하이퍼파라미터에 의존하여 그 하이퍼파라미터가 특정 값을 가질 때에만 존재하는 하이퍼파라미터가 있는데, 이를

조건부 하이퍼파라미터(conditional hyperparameter)라고 한다. 예를 들어, 활성 함수가 ReLU 함수, sigmoid 함수, hyperbolic 함수 등일 때에는 추가적인 하이퍼파라미터가 필요하지 않지만, parametric ReLU (PReLU) 함수일 때에는 활성 함수의 음수 구간 기울기를 결정하는 상수가 필요하다. 이 상수는 활성 함수가 PReLU일 때에만 존재하므로 조건부 하이퍼파라미터이다.

이처럼 하이퍼파라미터마다 주어진 범위가 다르며, 각 하이퍼파라미터의 주어진 범위는 문제 (1)의 제약식에 반영되어 있다. 식 (1.2), (1.3), (1.4), (1,5)는 이러한 제약식의 예시를 보여준다. θ_1은 연속형 하이퍼파라미터의 예시로, $[m_1, M_1]$의 구간을 범위로 가진다. θ_2는 이산형 하이퍼파라미터의 예시로, 자연수 값을 범위로 가진다. θ_3는 범주형 하이퍼파라미터의 예시로, 수가 아닌 c_1, c_2, c_3 등의 값을 가질 수 있다. 한편, θ_3가 c_3일 때 조건부 하이퍼파라미터인 $\lambda_{3,3}$가 존재하여 θ_3는 c_3와 조건부 하이퍼파라미터 $\lambda_{3,3}$의 순서쌍으로 표현할 수 있다. θ_3의 조건부 하이퍼파라미터인 $\lambda_{3,3}$는 연속형 하이퍼파라미터로 $(m_{3,3}, \infty)$의 구간을 범위로 가진다.

2.2 하이퍼파라미터 최적화의 난점

HPO를 위해서는 먼저 문제 (1)처럼 최소화할 비용함수와 최적의 값을 결정할 하이퍼파라미터를 정한다. 이후 이 최적화 문제의 최적해를 구해 최적의 하이퍼파라미터를 결정한다. 하지만 기계학습 및 HPO의 특성상 최적해를 구하는 데 여러 난점이 존재한다. 현재까지 개발된 많은 HPO의 알고리즘들은 이러한 난점을 해결하기 위한 방향으로 연구되었다. 3장에서 본격적으로 HPO의 알고리즘들을 소개하기에 앞서 본 절에서는 HPO의 여러 난점을 분석하고자 한다. 본 논문에서는 HPO의 난점을 크게 아래와 같이 분류하였다.

2.2.1 비용 함수의 복잡성

일반적으로 문제 (1)의 목적 함수인 $J(\boldsymbol{\Theta}|\mathbf{X}_{\text{tr}}, \mathbf{X}_{\text{va}})$는 하이퍼파라미터 $\boldsymbol{\Theta}$에 대해 closed-form으로 나타내기 어려운 black box function이다. $J(\boldsymbol{\Theta}|\mathbf{X}_{\text{tr}}, \mathbf{X}_{\text{va}})$가 $\boldsymbol{\Theta}$에 대해 closed-form으로 표현된다고 하더라도 매우 복잡한 형태로 표현되며 불연속하거나 미분불가능한 함수인 경우가 대부분이다. 이에 따라 흔히 사용되는 최적화 알고리즘인 경사하강법이나 convex optimization의 기법 등을 문제 (1)에 바로 적용하여 최적해를 구하는 것은 어렵다. 따라서 목적 함수가 black box function이거나 복잡한 꼴의 함수인 최적화 문제의 최적해를 구할 수 있도록 HPO의 알고리즘이 연구되어야 한다.

2.2.2 하이퍼파라미터의 종류

현재 최적화 문제 풀이의 기법들은 목적 함수가 연속 함수이고 결정 변수의 범위가 연속적일 때 적용 가능한 경우가 대부분이다. 예를 들어, 함수의 도함수나 기울기(gradient)를 계산해야 하는 경사하강법 등의 알고리즘들은 연속된 범위에서의 변수를 결정 변수로 가지는 연속 함수, 그 중에서도 미분가능한 함수에만 적용할 수 있다. 이산형 하이퍼파라미터의 경우, 많은 최적화 기법들을 적용하기 어려워 연속형 하이퍼파라미터보다 최적해를 구하기가 더 난해하지만, integer programming의 일반적인 알고리즘들을 적용하여 최적해를 구할 수도 있다. 하지만 범주형 하이퍼파라미터는 수치형 값을 가지지 않기 때문에 범주형 하이퍼파라미터가 변수로 포함된 목적 함수나 제약식에 해석적으로 접근하여 최적해를 구하는 것은 매우 어렵다. 특히 조건부 하이퍼파라미터는 다른 하이퍼파라미터 값에 따라 존재 여부와 형태가 제각각이라 범주형 하이퍼파라미터와 마찬가지로 해석적인 방법으로 최적해를 구하기가 매우 어렵다. 이에 따라, 다양한 종류의 하이퍼파라미터에 적용 가능한 알고리즘 개발이 필수적이고, 이에 대한 연구가 필요하다.

2.2.3 하이퍼파라미터의 개수

일반적으로 하나의 기계학습 모델에는 수십에서 수백 개 이상의 모델 하이퍼파라미터와 알고리즘 하이퍼파라미터가 포함되어 있다. 더 많은 하이퍼파라미터가 HPO의 결정 변수로 포함될수록 더 좋은 기계학습의 성능을 기대할 수 있다. 하지만 일반적으로 최적화 문제는 결정 변수의 수가 많아질수록 복잡도가 급격하게 증가한다. 이러한 상황에서 기존의 알고리즘을 그대로 적용하면 최적해를 구하는 데 시간이 매우 많이 소요되거나 최적해를 찾는 것이 불가능하다. 따라서 많은 수의 하이퍼파라미터에도 적용할 수 있는 HPO 알고리즘을 개발하는 연구가 진행되어야 한다.

2.2.4 성능 평가 시간

기본적으로 HPO를 위해서는 그림 1에서처럼 주어진 하이퍼파라미터마다 기계학습 모델의 성능을 평가하는 과정이 필수적이다. 일반적으로 기계학습의 성능 평가는 수많은 연산 과정을 요구하여 매우 긴 학습 시간을 요구한다. 학습하고자 하는 데이터의 크기가 클수록, 데이터의 수가 많을수록, 학습 알고리즘의 복잡

도가 높을수록 학습에는 더 많은 시간이 소요될 것이다. 아무리 우수한 성능의 하이퍼파라미터를 찾을 수 있는 알고리즘이라도 그 알고리즘의 수행 시간이 비현실적으로 길어 현실에서 사용할 수 없다면 그 알고리즘은 무의미하게 될 것이다. 따라서 학습에 요구되는 시간을 고려하여 성능 평가에 필요한 시간을 줄이거나 전체 알고리즘의 소요 시간을 줄일 수 있는 연구가 진행되어야 한다.

Ⅲ. 하이퍼파라미터 최적화 연구 동향

HPO 문제 풀이에는 2.2절에서 언급된 여러 난점이 있으며, HPO의 알고리즘들은 이러한 난점을 해결하기 위한 방향으로 연구되고 있다. 본 장에서는 현재까

표 1. 하이퍼파라미터 최적화 알고리즘
Table 1. Algorithms for hyperparameter optimization.

Paper	Main contribution	Search strategy	Types of hyperparameters	Number of hyperparameters	Year
[9]	Complexity of cost function	GS	Continuous	~10	1998
[10]		RS	Continuous	~10	2012
[11]		RS	Continuous	~10	2016
[12]		RS	Continuous	~10	2017
[13]		RS	Continuous	~10	2017
[14]		BO	Continuous	10~20	-
[15]		BO	Continuous	10~20	2011
[16]		BO	Continuous	10~20	2022
[17]		Interpolation	Continuous	10~20	2017
[18]	Type of hyperparameters	BO	Continuous, discrete, categorical	10~20	2011
[19]		BO	Continuous, discrete, categorical	10~20	2011
[20]		BO, multi-armed bandit	Continuous, discrete, categorical, conditional	10~20	2020
[21]		Gradient descent	Categorical	Thousands	2019
[22]	Number of hyperparameters	Reverse-mode differentiation	Continuous	Thousands	2015
[23]		Gradient descent	Continuous	Thousands	2018
[24]		Implicit function theorem	Continuous	Millions	2020
[25]		NAS	Discrete, categorical	Millions	2017
[26]	Time for performance evaluation	RS	Continuous	10~20	2018
[27]		BO, Hyperband	Continuous	10~20	2018
[28]		DE, Hyperband	Continuous	10~20	2021
[29]		BO	Continuous	10~20	2013
[30]		Genetic algorithm	Continuous	~10	2022
[31]		NAS	Categorical	Thousands	2018
[32]		NAS	Categorical	Thousands	2018
[33]		NAS	Categorical	Thousands	2018
[34]		Q-learning	Continuous, discrete	Thousands	2022

지 연구된 HPO의 알고리즘들[9-34]을 각 알고리즘이 해결하고자 한 HPO의 난점에 따라 분류하고 구체적 내용을 소개한다.

본 장에서 소개될 HPO의 알고리즘들은 표 1에서 정리되어 있다. 2.2절에서 언급된 HPO의 네 가지 난점 중 각 알고리즘이 가장 크게 해결에 기여한 난점에 따라 알고리즘을 분류하였다. 또한, 각 알고리즘이 복잡한 비용 함수 문제를 해결하기 위해 활용한 기법과 최적화할 수 있는 하이퍼파라미터의 종류도 표 1에 나타내었다. 한편, 일반적으로 최적화할 하이퍼파라미터의 개수가 많아질수록 복잡도가 증가하여 알고리즘의 수행 시간이 길어지게 된다. 이에 따라 실제적으로 각 알고리즘을 사용하기에 적합한 하이퍼파라미터의 개수 또한 표 1에 정리하였다.

3.1 비용 함수의 복잡성

HPO의 최적화 문제에서 목적 함수인 비용 함수 $J(\boldsymbol{\Theta}|\mathbf{X}_{\text{tr}},\mathbf{X}_{\text{va}})$는 일반적으로 결정 변수인 하이퍼파라미터 $\boldsymbol{\Theta}$에 대해 black box function, 불연속 함수, 미분 불가능 함수 등 그 형태가 상당히 복잡하여 일반적인 최적화 문제의 기법들을 적용하기 어렵다. 본 절에서는 이러한 HPO의 최적화 문제의 특수성을 고려하여 개발된 알고리즘들[9-17]을 소개한다.

임의의 하이퍼파라미터에 대해 그 성능(비용 함수의 값)을 모르는 경우에 최적해를 구하는 가장 원초적인 방법은 탐색 공간 내 모든 하이퍼파라미터의 성능을 평가하는 것이다. 하지만 대부분의 경우 탐색 공간의 원소는 무수히 많아 모든 하이퍼파라미터의 성능을 평가하는 것이 현실적으로 불가능할 뿐만 아니라, 일반적으로 성능 평가에 매우 많은 시간이 소요되어 너무 많은 수의 하이퍼파라미터 성능을 평가하는 것도 현실적으로 어렵다. 따라서 최대한 좋은 성능을 보이는 하이퍼파라미터를 효율적으로 찾을 수 있도록 탐색 공간 내 수많은 원소들 중 성능을 평가할 하이퍼파라미터들을 효율적으로 선별하여야 한다.

초기 단계의 연구에서는 이에 대한 간단한 해결책으로 grid search (GS)[9]와 random search (RS)[10]를 활용한 알고리즘이 사용되었다. GS는 각 하이퍼파라미터의 구간마다 일정한 간격으로 값을 선택하고 선택된 값들로 구성된 모든 조합들의 성능을 평가하여 가장 좋은 성능을 보이는 조합을 선택하는 방법이다. 그림 2는 범위가 각각 $[m_1,M_1]$과 $[m_2,M_2]$로 주어진 두 연속형 하이퍼파라미터 θ_1과 θ_2를 최적화하고자 하는 상황이다. 그림 2(a)는 GS의 예시를 보여주는데, θ_1과 θ_2의 구간에서 각각 일정한 간격으로 3개의 값을 선택하여 총 9개의 하이퍼파라미터 조합이 생성되었다. GS는 성능을 평가할 하이퍼파라미터들을 탐색 공간 전체에서 고르게 선별할 수 있고 복잡한 알고리즘이 필요 없어 구현이 매우 간단하다는 장점이 있다. 반면, 설정한 간격이 작을수록 더 좋은 성능의 하이퍼파라미터를 찾을 가능성은 높아지지만, 그만큼 더 많은 하이퍼파라미터에 대한 성능 평가가 필요하므로 그만큼 더 긴 성능 평가 시간을 필요로 한다는 한계점도 있다. 또한 최적화할 하이퍼파라미터의 개수가 많아질수록 성능을 평가해야 하는 하이퍼파라미터 조합의 수가 기하급수적으로 많아지게 된다는 단점도 있다.

이에 따라, 훨씬 적은 수의 하이퍼파라미터 조합들만으로도 GS와 비슷한 성능을 보일 수 있는 RS가 주목받기 시작하였다. RS는 탐색 공간 전체에서 무작위로 하이퍼파라미터들을 추출하여 각각의 성능을 평가한 뒤 그중 가장 좋은 성능을 보이는 하이퍼파라미터를 선택하는 방법이다. 그림 2(b)에서는 그림 2(a)에서와 같은 개수인 9개의 조합을 탐색 공간 내에서 무작위로 선택한 결과를 보여준다. [10]에서는 같은 수의 하이퍼파라미터 조합을 선택할 때, GS보다 RS가 더 좋은 성능의 조합을 찾을 가능성이 더 높다는 것을 수식적으로 분석하고 실험적으로 검증하였다. RS 또한 GS와 마찬가지로 복잡한 알고리즘이 요구되지 않아 구현이 간단하다는 장점을 갖는다.

성능을 평가할 하이퍼파라미터를 선별하기 위한 GS나 RS 기반의 알고리즘은 모든 하이퍼파라미터의 성능을 평가하는 것이 아니기 때문에 최적의 하이퍼파라미터가 성능 평가 대상으로 선별되지 않을 경우 최종적으로 구한 하이퍼파라미터의 성능이 최적해의 성능 값과 차이가 클 수 있다는 근본적인 한계점을 지니고 있다. 따라서 보다 정교한 알고리즘을 이용하여 최적해에 가까운 하이퍼파라미터를 찾기 위한 기법들이 개발되었다. [11]에서는 이러한 기법들 중 하나로

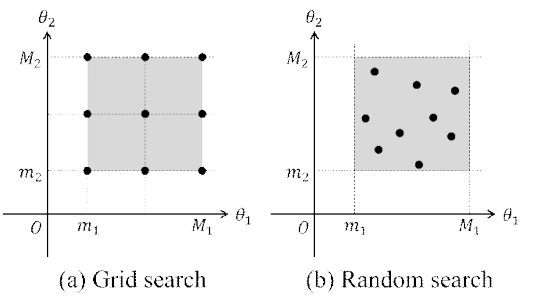

그림 2. GS와 RS의 예시
Fig. 2. Examples of grid search and random search.

covariance matrix adaptation evolution strategy (CMA-ES) 알고리즘이 제안되었다. CMA-ES에서는 매 단계마다 정해진 개수의 하이퍼파라미터들을 다변수 정규 분포(multivariate normal distribution)에 따라 추출한다. 매 단계마다 추출된 하이퍼파라미터들의 성능을 기반으로 다음 단계의 추출이 따를 다변수 정규 분포의 평균 벡터(mean vector)와 공분산 행렬(covariance matrix)을 결정한다. 매 단계를 반복할 때마다 추출된 하이퍼파라미터들은 점점 최적해에 가까워지는 경향성을 띠며, 수렴 조건을 만족할 때까지 이 시행을 반복하여 최종적으로 가장 좋은 성능을 보이는 하이퍼파라미터를 선택한다.

[12]에서는 particle swarm optimization (PSO) 알고리즘을 활용하여 성능을 평가할 하이퍼파라미터들을 선별하는 기법을 제안하였다. PSO 알고리즘에서는 CMA-ES와 마찬가지로 매 단계마다 일정한 개수의 하이퍼파라미터들의 성능을 평가한다. 현재 단계까지 측정된 성능 값들을 기반으로 더 좋은 성능의 하이퍼파라미터를 찾을 수 있도록 다음 단계에서 성능 평가를 할 하이퍼파라미터들이 업데이트되며, 수렴 조건을 만족할 때까지 이 과정을 반복한다.

[13]에서는 sequential optimization과 parallel search의 기법을 결합하여 성능을 평가할 하이퍼파라미터를 찾는 Population Based Training (PBT) 알고리즘을 제안하였다. PBT에서는 여러 개의 CPU를 이용하여 기계학습 모델의 학습 파라미터와 하이퍼파라미터를 동시에 업데이트한다. CMA-ES, PSO, PBT 알고리즘은 일반적으로 GS나 RS 기반 알고리즘보다 최적해에 더 가까운 하이퍼파라미터를 찾을 가능성이 높지만, 성능 평가가 더 많이 필요하여 더 긴 수행 시간을 필요로 한다는 단점이 있다.

비록 CMA-ES, PSO, PBT 알고리즘이 최적해에 가까운 해를 찾기 위해 보다 정교한 알고리즘을 활용할지라도, 여전히 탐색 공간 내 모든 원소의 성능을 비교한 것이 아니기 때문에 구한 성능이 최적 성능과 큰 차이가 있을 가능성이 여전히 존재한다. 이에 따라 최근 대부분의 연구들에서는 성능이 평가되지 않은 하이퍼파라미터에 대해서도 성능을 예측하여 탐색 공간 내 모든 원소들을 최적해의 대상으로 할 수 있는 알고리즘들이 연구되고 있다. 이러한 알고리즘들은 정확한 함숫값을 모르는 하이퍼파라미터들에 대해서도 비용 함수의 값을 예측하는 모델을 필요로 하며, 순차적으로 학습하며 비용 함수를 업데이트하여 sequential model-based optimization (SMBO)라고도 부른다.

최근에는 SMBO 중에서도 BO[14] 기반의 알고리즘이 가장 많이 주목받고 연구되고 있다. BO에서는 매 단계마다 하나의 하이퍼파라미터에 대한 성능을 평가하고, 현재 단계까지 성능이 평가된 하이퍼파라미터들을 기반으로 성능 값을 모르는 하이퍼파라미터의 성능 값을 예측하는 모델을 세운다. 이때 이 예측 모델의 기반이 되는 확률적 모델을 surrogate model이라고 한다. 그림 3은 결정 변수가 실수 구간 $[m,M]$을 범위로 가지는 하나의 연속형 하이퍼파라미터 θ인 최적화 문제에 Gaussian process를 surrogate model로 사용하여 BO를 적용한 예시를 보여준다. 세 단계 동안 세 개의 하이퍼파라미터 θ_1, θ_2, θ_3의 비용 함수 값을 계산하였고, 이를 기반으로 비용 함수 $J(\theta|\mathbf{X}_{\text{tr}},\mathbf{X}_{\text{va}})$를 Gaussian process로 모델링하였다. 정확한 비용 함수 값을 아는 θ_1, θ_2, θ_3에서는 비용 함수 값이 확정적으로 결정되며 나머지 값들에서는 Gaussian random variable로 표현된다. 성능의 예측 모델을 세운 후에는 다음 단계에서 성능을 평가할 하이퍼파라미터를 선별한다. 성능이 좋을 것으로 기대되는 하이퍼파라미터를 선별하기 위해서, exploitation과 exploration 사이의 trade-off를 잘 조절하여야 한다. 즉, 현재까지 평가한 성능 중 가장 좋은 성능을 보이는 하이퍼파라미터 주변에서 더 좋은 성능의 하이퍼파라미터가 관측될 가능성과 아직 탐색하지 않은 미지의 영역에서 더 좋은 성능의 하이퍼파라미터가 관측될 가능성 모두를 고려하여야 한다. 다음 단계의 하이퍼파라미터를 결정하는 역할을 하는 함수를 acquisition function이라고 하며, 대표적인 예로 expected improvement (EI), probability improvement (PI), upper confidence bound (UCB) 등이 있다. BO는 탐색 공간 내 모든 원소들의 성능 예측이 가능하고 그 원리가 수학적으로 뒷받침되고 있어, 최근 HPO 연구에서는 BO를 기반으로 한 알고리즘이 활발하게 연구되고 있다.

BO 기반 알고리즘인 GP-Hedge[15]는 acquisition

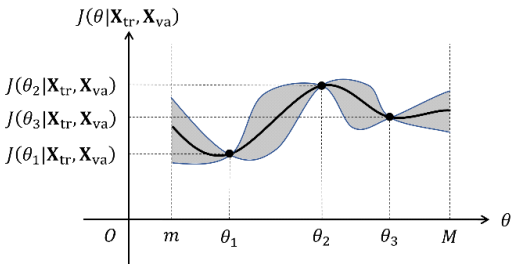

그림 3. BO의 surrogate model 예시
Fig. 3. Example of surrogate model in Bayesian optimization.

function을 하나로 고정하지 않고 여러 acquisition function 중에서 매 단계마다 가장 좋은 성능의 하이퍼파라미터를 찾을 것으로 예상되는 하나의 acquisition function을 선택한다. 매 단계마다 선택된 acquisition function은 성능 평가를 마친 뒤 평가된 성능을 바탕으로 보상을 받고, 현재 단계까지 각 acquisition function의 누적 보상에 softmax function을 적용하여 각 acquisition function의 확률 모델을 만든다. 매 단계마다 이 확률 모델에 기반하여 하나의 acquisition function을 추출하고 그 acquisition function에 의해 선택된 하이퍼파라미터에 대해 다음 단계에서 성능 평가를 한다. GP-Hedge 하나가 아닌 여러 acquisition function을 사용함으로써 exploitation과 exploration 사이의 trade-off를 반영하여 성능이 더 좋은 하이퍼파라미터를 선택할 가능성을 높였다는 데 의의가 있다.

[16]에서는 최적의 하이퍼파라미터 경향성에 대해 사용자가 미리 알고 있는 선험적인 정보를 반영하여 BO의 매 단계에서 하이퍼파라미터를 추출하는 πBO라는 알고리즘을 개발하였다. 즉, 탐색 공간 내 각 원소가 최적의 하이퍼파라미터일 확률 분포가 주어졌을 때, 매 단계의 acquisition function에 이 확률 분포를 곱하여 새로운 acquisition function을 정의하였다. 이때, BO의 단계가 진행됨에 따라 선험적인 확률 분포의 영향이 점점 줄어들도록 하였다. 임의로 추출한 하이퍼파라미터에 기반하여 acquisition function을 계산한 기존의 BO 기법에 비해, 하이퍼파라미터에 대한 선험적인 정보를 반영한 πBO가 더 좋은 성능을 보이는 것을 실험을 통해 검증하였다.

BO 기반 알고리즘은 surrogate model로 Gaussian process 등의 확률 모델을 사용한다. 이와 달리, Hyperparameter Optimization using RBF-based surrogate and DYCORS (HORD)[17]는 surrogate model로서 결정적 모델인 radial basis function (RBF)을 사용한다. 비용 함수는 현재 단계까지 성능 평가를 완료한 하이퍼파라미터들에 RBF로 interpolation을 적용하여 예측한다. 다음 단계에서 성능을 평가할 하이퍼파라미터는 현재 단계까지 성능을 평가한 하이퍼파라미터들과 정규 분포를 이용하여 결정한다. HORD는 BO 기반의 알고리즘에 비해 복잡도가 매우 낮아 예측된 비용 함수를 계산하는 데 걸리는 시간이 짧다는 데 의의가 있다.

3.2 하이퍼파라미터의 종류

하이퍼파라미터는 연속형, 이산형, 범주형, 조건부 등 그 종류가 다양하다. 연속형 하이퍼파라미터에 대해 적용 가능한 HPO 알고리즘이 가장 많이 연구되어 왔지만, 다른 종류의 하이퍼파라미터도 최적화할 수 있도록 하이퍼파라미터의 종류를 고려한 알고리즘들[18-21]도 연구되고 있다.

Gaussian process를 surrogate model로 사용하는 기존의 BO 기반의 알고리즘은 연속형 하이퍼파라미터에 대해서만 최적해를 구할 수 있다. 이에 [18,19]에서는 BO에서 Gaussian process가 아닌 다른 surrogate model을 사용하여 다른 종류의 하이퍼파라미터에 대해서도 최적해를 구할 수 있는 알고리즘들을 개발하였다. Sequential Model-based Algorithm Configuration (SMAC)[18]에서는 수치형 하이퍼파라미터에만 적용 가능한 Gaussian process의 kernel function에 추가적으로 범주형 하이퍼파라미터에도 적용 가능한 Hamming distance 기반의 kernel function을 함께 고려하여 수치형과 범주형 하이퍼파라미터를 모두 고려할 수 있게 하였다. SMAC에서는 또다른 방법으로 수치형 하이퍼파라미터에도 적용할 수 있는 regression tree들의 모임인 random forest를 surrogate model로 활용하여 수치형과 범주형 하이퍼파라미터에 모두 적용 가능한 BO 기반 알고리즘을 제안하였다. Tree-structured Parzen estimator (TPE)[19]에서도 수치형과 범주형 하이퍼파라미터에 모두 적용 가능한 Parzen estimator를 surrogate model로 사용하여 다양한 종류의 하이퍼파라미터에 적용 가능하게 하였다.

기계학습의 조건부 하이퍼파라미터는 특정 하이퍼파라미터가 특정 값을 가질 때에만 존재하여 일반적인 최적화 기법으로 HPO의 최적해를 구하기 어렵다. [20]에서는 범주형 하이퍼파라미터와 이에 의존하는 연속형 조건부 하이퍼파라미터를 최적화하는 BO 기반 알고리즘 Bandit-BO를 제안하였다. Bandit-BO에서는 multi-armed bandit 기법을 활용하여 범주형 하이퍼파라미터의 값마다 대응되는 arm들을 설정한다. Bandit-BO는 최적의 arm과 각 arm에서의 최적의 연속형 하이퍼파라미터 값을 정하는 것을 목적으로 하며, multi-armed bandit의 Thompson sampling 기법을 BO에 적용하였다.

한편, 함수의 기울기를 이용하는 경사하강법은 알고리즘이 간단하여 기계학습에서 많이 활용되고 있다. 함수의 기울기는 그 함수가 연속함수일 때에만 구할 수 있으므로 범주형 하이퍼파라미터의 최적화에는 경사하강법을 바로 적용할 수 없다. [21]에서는 범주형 하이퍼파라미터를 연속형 하이퍼파라미터로 변환한 후 경사하강법을 적용하여 최적해를 구하는

Differentiable Architecture Search (DARTS)라는 이름의 알고리즘을 제안하였다. 신경망의 각 노드에 쓰이는 활성 함수를 결정하기 위해 DARTS에서는 각 활성 함수 후보에 실수 값을 대응시키고 이 실수 값들에 softmax function을 적용한 값을 각 활성 함수 후보의 가중치로 사용한다. 이후 활성 함수 후보들에 각각의 가중치 값을 곱한 함수들을 모두 더한 가중합 함수를 정의하여 알고리즘에 이 함수를 사용한다. 이 함수는 각 활성 함수마다 대응되는 실수 값들에 대한 미분가능함수이며, 따라서 이 함수에 경사하강법을 적용하여 실수 값들을 결정하고 최종적으로 가장 큰 가중치 값을 가지는 활성 함수를 선택한다. DARTS는 특정 하이퍼파라미터를 다른 종류의 하이퍼파라미터로 변환하는 아이디어를 제시하였다는 점에서 의의가 있다.

3.3 하이퍼파라미터의 개수

기계학습 모델의 종류마다 하이퍼파라미터의 개수는 다르지만, 심층 학습에서는 하이퍼파라미터가 수백에서 수천 개 이상으로 많은 경우도 있다. HPO의 결정 변수로 쓰이는 하이퍼파라미터의 개수가 많을수록 더 좋은 기계학습의 성능을 기대할 수 있다. 하지만 일부 알고리즘들은 결정 변수의 개수가 적을 때에만 적용할 수 있거나 결정 변수의 수가 많을 때에는 실질적으로 적용이 불가능한 경우가 많다. 따라서 하이퍼파라미터의 개수가 많을 때에도 적용 가능한 알고리즘[22-25]이 개발되고 있다.

경사하강법은 수식적으로 표현된 함수의 기울기를 이용하기 때문에 일반적으로 변수의 개수가 많아도 성능에 크게 영향을 미치지 않는다. 이에 [22-24]에서는 하이퍼파라미터에 대한 비용 함수의 기울기를 이용하여 많은 수의 하이퍼파라미터를 최적화할 수 있는 경사하강법 기반의 알고리즘을 개발하였다.

역전파 학습(reversible learning)을 HPO에 적용한 [35]에서의 알고리즘은 기계학습 모델의 구조가 복잡할수록 역전파 학습의 중간 단계 연산 값을 저장하기 위한 메모리가 많이 필요하다는 문제점이 있다. [22]에서는 이러한 문제점을 해결하기 위해 중간 단계의 연산 값을 저장할 필요 없이 학습된 하이퍼파라미터로부터 초기 하이퍼파라미터 값을 추적할 수 있는 방법을 제안하였다. 따라서 큰 메모리가 없어도 역전파 학습을 통해 HPO에 경사하강법을 적용할 수 있게 하였으며, 실험을 통해 수천 개의 하이퍼파라미터에도 알고리즘이 잘 동작하는 것을 확인하였다.

[23]에서는 기계학습 모델의 최적의 학습 파라미터 값이 하이퍼파라미터에 의해 결정된다는 점에 주목하여, 주어진 하이퍼파라미터에 대한 학습 파라미터 값을 예측하는 hypernetwork 모델을 제안하였다. 하이퍼파라미터에 대한 비용 함수는 hypernetwork와 가중치에 대한 비용 함수의 합성 함수로 표현되며, 연쇄법칙에 의해 하이퍼파라미터에 대한 비용 함수의 기울기는 hypernetwork의 기울기와 가중치에 대한 비용 함수의 기울기의 곱으로 표현된다. 따라서 최적의 하이퍼파라미터는 경사하강법 알고리즘을 이용하여 업데이트할 수 있다. [23]에서는 이 알고리즘을 hypertraining이라고 부르며, 하이퍼파라미터의 개수가 많을 때 기존의 BO 기반 알고리즘이 제대로 된 성능을 보이지 못하는 것에 비해 hypertraining은 하이퍼파라미터의 개수가 많아도 잘 동작하는 것을 실험을 통해 검증하였다.

[24]에서는 수학적인 방법으로 기울기를 근사하는 방법을 제안하였고 HPO에 경사하강법을 적용하기 위해 근사된 기울기를 이용하였다. [23]에서와 비슷하게 하이퍼파라미터에 대한 비용 함수의 기울기를 두 기울기의 곱으로 나타낸 뒤, 하이퍼파라미터에 대한 가중치의 기울기에 implicit function theorem (IFT)을 적용하여 근사된 기울기 식을 유도하였다. 근사된 식은 Hessian의 역행렬 계산을 필요로 하는데 하이퍼파라미터의 개수가 많아질수록 역행렬의 계산 시간이 매우 길어지기 때문에 역행렬은 Neumann series를 활용하였다. 최종적으로 얻은 근사된 기울기에 경사하강법을 적용하여 최적의 하이퍼파라미터를 구하였다. 실험을 통해, 수백만 개의 HPO 상황에서도 알고리즘이 좋은 성능을 보이며 학습 파라미터와 하이퍼파라미터를 동시에 최적화함에도 시간이 많이 소요되지 않은 것을 확인하였다.

[25]에서는 순환 신경망과 강화 학습 기법을 활용한 NAS를 위한 알고리즘을 제안하였다. 이 알고리즘에서는 신경망 각 층의 filter 개수와 크기, stride 크기를 출력 값으로 하는 순환 신경망을 설계하고, 이 순환 신경망의 가중치 값들은 강화 학습을 이용하여 업데이트한다. [25]에서의 순환 신경망처럼, NAS를 위한 알고리즘에는 controller 역할을 하는 별도의 모델이 신경망의 모델 하이퍼파라미터 값을 결정하는 프레임워크가 많이 활용되고 있다.

3.4 성능 평가 시간

하이퍼파라미터의 성능 평가를 위해서는 기계학습 모델을 주어진 훈련 데이터와 검증 데이터에 학습시키는 과정이 필수적이다. 일반적으로 기계학습 모델의 학습에는 시간이 많이 소요되기 때문에 성능 평가를

효율적으로 진행하여 수행 시간을 최소화하고자 하는 알고리즘들[26-34]이 연구되고 있다.

HPO에서 활발하게 연구되고 있는 BO 기반 알고리즘들은 최적해에 가까운 하이퍼파라미터를 찾는 데에는 유리하지만, 매 단계마다 순차적으로 학습을 해야 하기 때문에 수행 시간이 길다는 단점이 있다. Hyperband[26]는 수행 시간을 줄이는 것에 초점을 맞춰 RS를 기반으로 제안된 새로운 알고리즘이다. RS에서는 선별된 모든 하이퍼파라미터들마다 각각의 성능 값이 수렴 조건을 만족할 때까지 학습 파라미터의 업데이트 과정을 반복하여야 한다. 반면, Hyperband에서는 모든 하이퍼파라미터들에 대해 학습 파라미터 업데이트 과정을 조금만 반복한 뒤, 현재 단계에서의 성능을 비교하여 성능이 떨어지는 하이퍼파라미터들을 제거하고 나머지 하이퍼파라미터들에 대해서만 더 많은 업데이트 과정을 반복한다. 하나의 하이퍼파라미터가 남을 때까지 이 과정을 반복하여 그 하이퍼파라미터를 최적해로 채택한다. 더 많은 하이퍼파라미터들의 성능을 평가할수록 실제 최적해에 더 가까운 성능 값의 하이퍼파라미터를 찾을 가능성이 높아지지만, 그만큼 각 하이퍼파라미터들의 성능을 평가하는 데 투입되는 학습 파라미터 업데이트 과정의 수도 적어져 각 하이퍼파라미터의 현재 단계에서의 값과 실제 성능 값의 오차가 클 가능성도 높아진다. Hyperband 알고리즘에서는 이러한 trade-off 관계를 고려하여 초기에 성능을 평가할 하이퍼파라미터의 개수와 매 단계에서 제거할 하이퍼파라미터의 비율을 조절할 수 있도록 한다. Hyperband 알고리즘은 BO 기반 알고리즘보다 5~30배 빠른 결과를 보여주었다.

Hyperband도 성능을 평가한 하이퍼파라미터들 중에서만 최종 하이퍼파라미터 값을 선택하므로 실제 최적해의 성능과 차이가 큰 하이퍼파라미터 값이 선택될 가능성이 있다는 RS의 본질적인 한계를 여전히 지니고 있다. [27]에서는 이러한 한계를 보완하기 위해 높은 성능을 보장하는 BO 기반 알고리즘을 Hyperband와 함께 사용하였다. 전체적인 알고리즘은 Hyperband를 따르되, 성능을 측정할 하이퍼파라미터들을 선별할 때 BO 기법을 활용하여 최적해에 가까운 하이퍼파라미터를 선별할 가능성을 높였다. Hyperband 기반 알고리즘들은 기본적으로 수행 시간이 짧기 때문에 이 알고리즘도 BO가 가지고 있는 긴 수행 시간의 문제점을 Hyperband를 통해 보완하였다고 할 수 있다.

[27]에서 BO 기반 알고리즘을 통해 하이퍼파리미터 추출에 기인하는 Hyperband의 근본적인 한계를 보완하였다면, [28]에서는 하이퍼파라미터를 추출할 때 Differential Evolution (DE) 기법을 활용하여 이 한계를 보완하였다. DEHB라고 명명된 이 알고리즘은 하이퍼파라미터들을 임의로 추출하는 대신 최적해에 가까운 하이퍼파라미터들을 추출할 수 있도록 DE 알고리즘을 활용하고, 이후 Hyperband를 적용하였다. DEHB도 기본적으로 Hyperband의 알고리즘을 따르기 때문에 짧은 수행 시간을 보인다.

[26-28]에서는 수행 시간을 줄이기 위해 Hyperband라는 새 알고리즘을 활용하였다. 반면 [29]에서는 BO 기반 알고리즘을 변형하여 수행 시간을 줄이는 방법을 제안하였다. BO 기반 알고리즘에서는 새로운 데이터에 대한 학습을 할 때 기존의 데이터로 학습된 값들을 전혀 이용하지 못하고 초기 상태에서 다시 학습하여야 한다. 또한 성능 평가에는 시간이 많이 소요되고, 특히 데이터의 크기가 큰 경우에는 시간이 더 많이 필요하기 때문에 여러 데이터들에 대한 학습이 필요한 경우 시간이 매우 많이 소모된다. 이에 따라 [29]에서는 다음 단계에서 새 하이퍼파라미터의 성능을 평가할 때 이전의 하이퍼파라미터들의 학습 정보를 이용하여 학습 시간을 줄이고자 하였다. [29]에서는 기존의 BO 기반에서 사용하던 surrogate model인 Gaussian process 대신 이를 확장한 multi-task Gaussian process를 활용하였다. Multi-task Gaussian process를 이용하면 여러 task들끼리의 상관관계를 고려하여 서로 다른 학습 데이터를 사용하는 각 task마다 학습된 정보들을 서로 공유할 수 있다. 이런 방식으로 여러 데이터들에 대해 학습하면 기존의 BO에서 서로 다른 데이터로부터의 학습들끼리 정보를 공유하지 않을 때보다 학습에 필요한 시간이 훨씬 줄어들게 된다.

[30]에서는 유전 알고리즘(genetic algorithm)을 기반으로 하여 보다 적은 횟수의 성능 평가로 최적의 하이퍼파라미터를 찾을 수 있는 simple deterministic selection genetic algorithm (SDSGA)라는 알고리즘을 개발하였다. SDSGA는 기존의 유전 알고리즘에서 exploration을 약화시키고 exploitation을 좀 더 강화하였다. 즉, 이전 단계에서 성능을 측정한 하이퍼파라미터들 중 성능이 좋은 하이퍼파라미터들 주변에서 crossover와 mutation을 진행하였다. 다만, exploration 측면이 너무 약해지지 않도록 mutation 확률을 높여 좀 더 넓은 영역의 탐색 공간을 탐색하도록 하였다. 제안된 SDSGA 알고리즘은 기존에 잘 알려진 많은 기법들에 비해 더 좋은 성능을 보이며 수렴 속도도 빨라진 것을 실험을 통해 확인하였다.

한편, [25]에서 소개한 알고리즘은 탐색 공간의 크기가 너무 커서 ImageNet[36]과 같은 크기가 큰 데이터에 적용하면 연산이 복잡하여 수행 시간이 매우 길다는 문제점이 있다. [31]에서는 이러한 문제점을 해결하기 위해 NAS에 전이 학습(transfer learning) 기법을 활용하고자 하였다. 이를 위해 NAS의 탐색 공간의 크기를 줄여 작은 데이터에서 학습하고 그 결과를 큰 데이터에 사용하는 알고리즘을 제안하였다. [31]에서는 하나의 신경망을 여러 블록 구조로 나눈 뒤 이 블록들의 연결 방식과 각 블록 내에서 사용할 연산을 결정하는 방식으로 탐색 공간을 축소하였으며, 축소된 새로운 탐색 공간을 NASNet 탐색 공간이라고 불렀다. 이런 방식으로 축소된 구조를 CIFAR-10과 같은 비교적 크기가 작은 데이터에 학습시키고, 학습 결과를 크기가 큰 데이터에 사용할 있게 하였다. 이렇게 전이된 학습 결과는 큰 데이터에서 기존 알고리즘을 직접 적용하는 것보다 수행 시간이 훨씬 단축된다는 것을 실험을 통해 검증하였다.

일반적으로 HPO의 성능 평가에서는 각 하이퍼파라미터마다 독립적으로 기계학습 모델의 학습 파라미터를 업데이트한다. 각 하이퍼파라미터에 대응되는 기계학습 모델의 학습 파라미터를 구하는 데에는 시간이 많이 소요되지만, 한 하이퍼파라미터의 성능 평가에서는 다른 하이퍼파라미터에 대응되는 학습 파라미터 정보를 활용하지 않기 때문에 최종적으로 최적해가 아닌 하이퍼파라미터에서에 대응되는 학습 파라미터 값들은 성능 값을 계산하기 위해서만 사용된 뒤 버려진다. [32,33]에서는 한 하이퍼파라미터로부터 계산된 학습 파라미터를 다른 하이퍼파라미터의 성능 평가에 활용하여 성능 평가에 소요되는 시간을 줄이고자 하는 아이디어를 NAS에 활용하였다. Efficient Neural Architecture Search (ENAS)[32]에서는 신경망의 구조를 결정하기 위해 directed acyclic graph (DAG)를 고려하였다. 하나의 DAG는 어떤 변들을 선택하는지에 따라 여러 신경망 구조에 대응시킬 수 있다. ENAS는 각각의 신경망 구조를 직접 학습하는 대신 DAG의 각 노드의 학습 파라미터 값을 학습하여, 탐색 공간 내 여러 신경망 구조들끼리 같은 학습 파라미터 값을 공유하도록 한다. [33]에서는 one-shot architecture search 알고리즘을 제안하였는데, 신경망 구조를 결정하기 위해 신경망의 각 노드의 입력 값과 노드 내 연산을 결정하고자 하였다. 한 노드에 대한 탐색 공간 내 모든 가능한 조합을 one-shot model을 이용하여 하나의 모델로 표현하였고, 신경망 구조들을 하나하나 학습하는 대신 one-shot model을 학습하여 탐색 공간의 원소들끼리 학습 파라미터 값을 공유하도록 하였다. [32,33]에서는 한 하이퍼파라미터의 성능 평가 과정에서 구한 기계학습 모델의 학습 파라미터 값을 다른 하이퍼파라미터들에도 활용할 수 있도록 하는 패러다임을 제시하여 알고리즘의 수행 시간을 단축하였다는 데 의의가 있다.

한편, 기계학습 모델 중 신경망의 구조를 결정하는 하이퍼파라미터의 최적화는 NAS라는 알고리즘으로 따로 활발하게 연구되고 있지만, 신경망에는 구조를 결정하는 하이퍼파라미터 외에도 많은 하이퍼파라미터가 포함되어 있다. 이러한 하이퍼파라미터에는 학습률, momentum, dropout rate, batch의 크기 등이 있으며, 심층 신경망에서 은닉 층의 개수가 많아지고 구조가 복잡할수록 이러한 하이퍼파라미터들을 최적화하는 데에 필요한 연산량이 매우 많아진다. [34]에서는 이러한 신경망에서의 HPO의 특징에 주목하여 신경망의 최적의 하이퍼파라미터를 구하기 위해 Q-learning을 활용한 알고리즘을 개발하였다. 이 알고리즘에서는 최적의 하이퍼파라미터를 찾기 위해 Q-table을 지속적으로 업데이트하며, 효율적인 최적화를 위한 초기 상태와 종료 조건이 제시되었다. 이러한 Q-learning을 활용한 알고리즘은 기존의 RS, BO, 유전 알고리즘으로 신경망의 최적의 하이퍼파라미터를 구할 때보다 더 좋은 효율성을 보인다.

Ⅳ. 하이퍼파라미터 최적화 미래 연구 방향

본 논문에서 소개한 HPO의 알고리즘들은 하이퍼파라미터와 기계학습의 특징을 고려하여 기계학습 모델에 사용될 최적의 하이퍼파라미터를 결정하며, 이렇게 결정된 하이퍼파라미터는 기계학습 모델의 성능 향상에 도움을 준다. 하지만 기기의 제한된 연산 능력에 의해 최적의 하이퍼파라미터를 찾는 데 시간이 오래 걸리고, 과적합과 같은 학습 모델 자체의 한계에 의해 HPO의 알고리즘을 통해 구한 하이퍼파라미터가 학습 모델의 성능 향상에 크게 도움을 주지 못할 수 있다. 따라서 기기의 제한된 연산 능력과 기계학습의 문제점을 고려하여 HPO의 알고리즘을 발전시킬 필요가 있다.

4.1 최적화할 하이퍼파라미터 결정

심층 학습에서 기계학습 모델의 하이퍼파라미터는 매우 다양하여 모든 하이퍼파라미터를 최적화하기에는 연산 시간 및 메모리 등에 한계가 있는 경우가 많다. 이는 최적화할 하이퍼파라미터가 많을수록 HPO

알고리즘의 복잡성이 높아지기 때문이다. 따라서 최적화할 하이퍼파라미터의 개수를 줄여 HPO 알고리즘의 복잡성을 줄일 필요가 있다. 하이퍼파라미터마다 기계학습 모델의 성능에 미치는 영향은 다르며[10,37], HPO 알고리즘은 학습 모델의 성능 향상을 목표로 하므로 학습 모델의 성능 향상에 영향을 많이 미치는 하이퍼파라미터들을 최적화할 대상으로 선택하고 영향을 덜 끼치는 하이퍼파라미터들을 최적화할 대상으로 선택하지 않는 식으로 최적화 문제의 결정 변수를 조정할 수 있다. 이렇게 최적화할 하이퍼파라미터를 결정한다면 성능이 거의 하락하지 않으면서 최적화할 하이퍼파라미터의 개수를 줄여 알고리즘의 복잡도를 낮출 수 있다. 예를 들어, 신경망의 가중치 계산에 직접적으로 사용되는 학습률은 기계학습의 정확도에 직접적인 영향을 미쳐 최적화할 대상으로 설정하고 간접적으로 사용되는 은닉 층의 개수는 최적화하지 않는 것이다. 이처럼 학습 모델의 성능에 미치는 영향에 따라 하이퍼파라미터에 우선 순위를 적용하여 결정 변수로서 사용될 하이퍼파라미터의 개수를 줄이는 연구를 할 수 있다.

4.2 탐색 공간의 크기 조절

하이퍼파라미터의 탐색 공간의 크기는 알고리즘 수행 시간에 크게 영향을 주어[6] 이를 적절하게 줄일 필요가 있다. 개별 하이퍼파라미터의 범위를 줄임으로써 전체 하이퍼파라미터의 탐색 공간의 크기를 줄일 수 있다. 예를 들어, 학습률의 범위를 (0,10]의 구간에서 (0,1]의 구간으로 줄임으로써 전체 하이퍼파라미터의 탐색 공간의 크기를 줄일 수 있다. 하지만 개별 하이퍼파라미터의 범위가 너무 좁다면 이 탐색 공간에서 찾은 최적의 하이퍼파라미터는 학습 모델의 성능 향상에 도움을 주지 못할 수도 있다. 예를 들어, 학습률의 범위를 (0,0.0001]의 좁은 범위로 더 축소한다면 이 범위에서 선택한 최적의 학습률이 학습 모델의 성능 향상에 영향을 미치지 못할 수도 있다. 따라서 이러한 trade-off 관계를 고려하여 각 하이퍼파라미터의 범위를 적절하게 조절하여 전체 하이퍼파라미터의 탐색 공간의 크기를 조절하는 연구가 진행될 수 있다.

4.3 과적합 방지

기계학습에서 특정 데이터만으로 기계학습 모델을 학습시킨다면 그 데이터에 대한 과적합[38] 현상이 발생할 수 있기 때문에 특정 학습 데이터에 치우치지 않도록 기계학습 모델을 학습시키는 것이 매우 중요하다. 하이퍼파라미터 역시 이러한 기계학습의 특징을 고려하여 학습 모델의 HPO 알고리즘이 연구돼야 한다. 하지만 현재의 HPO 알고리즘은 고정된 데이터에 대해 학습 모델의 하이퍼파라미터를 최적화하는 경우가 대부분이다. 즉, 기존의 HPO 알고리즘은 특정 데이터에 치우친 기계학습의 성능이 반영되었을 수 있다[39]. 이렇게 특정 데이터에 치우친 성능으로 하이퍼파라미터를 최적화한다면 그 하이퍼파라미터는 특정 데이터에서만 좋은 성능을 보이는 값일 수 있다. 따라서 특정 데이터에 치우치지 않고 일반적인 데이터에 대해 기계학습 모델의 하이퍼파라미터를 최적화하는 연구가 필요하다.

V. 결 론

최근 많은 분야에서 기계학습이 사용됨에 따라, 기계학습 모델의 성능 향상을 위해 HPO 알고리즘이 활발하게 연구되고 있다. 본 논문에서는 HPO 연구에서의 난점을 분석하였고, 이를 해결하기 위해 제안된 알고리즘들을 정리하였다. 현재까지도 HPO 알고리즘 연구는 활발하게 진행되고 있지만, 개발된 알고리즘들에도 여전히 한계들은 존재하고, 해결해야 할 HPO의 문제점들도 많이 존재한다. 현재까지 연구된 내용을 바탕으로 본 논문에서는 기계학습 모델의 HPO 연구가 앞으로 나아갈 수 있는 미래 연구 방향을 제시하였다.

References

[1] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255-260, Jul. 2015. (https://doi.org/10.1126/science.aaa8415)

[2] J. Lee, S. Shin, S. Yoon, and T. Kim, "Survey on artificial intelligence & machine learning models and datasets for network intelligence," *J. KICS*, vol. 47, no. 4, pp. 625-643, Apr. 2022. (https://doi.org/10.7840/kics.2022.47.4.625)

[3] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd Ed., Springer, 2009. (https://doi.org/10.1198/tech.2003.s770)

[4] Y.-H. Moon, I.-H. Shin, Y. J. Lee, and O. G. Min, "Recent research & development trends

in automated machine learning," *Electron. Telecommun. Trends*, vol. 34, no. 4, pp. 32-42, Aug. 2019. (https://doi.org/10.22648/ETRI.2019.J.340404)

[5] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295-316, Nov. 2020. (https://doi.org/10.1016/j.neucom.2020.07.061)

[6] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, pp. 1-21, Mar. 2019. (https://dl.acm.org/doi/10.5555/3322706.3361996)

[7] T. Yu and H. Zhu, "Hyper-parameter optimization: A review of algorithms and applications," *arXiv:2003.05689,* 2020. (https://doi.org/10.48550/arXiv.2003.05689)

[8] A. Morales-Hernández, I. V. Nieuwenhuyse, and S. R. Gonzalez, "A survey on multi-objective hyperparameter optimization algorithms for machine learning," *Artif. Intell. Rev.*, Dec. 2022. (https://doi.org/10.1007/s10462-022-10359-2)

[9] G. Montavon, G. B. Orr, and K.-R. Müller, *Neural Networks: Tricks of the Trade*, 2nd Ed., Springer, 2012. (https://doi.org/10.1007/978-3-642-35289-8)

[10] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *JMLR*, vol. 13, pp. 281-305, Feb. 2012. (https://dl.acm.org/doi/10.5555/2188385.2188395)

[11] N. Hansen, "The CMA evolution strategy: A tutorial," *arXiv:1604.00772,* 2016. (https://doi.org/10.48550/arXiv.1604.00772)

[12] P. R. Lorenzo, J. Nalepa, M. Kawulok, L. S. Ramos, and J. R. Pastor, "Particle swarm optimization for hyper-parameter selection in deep neural networks," in *Proc. GECCO 2017*, pp. 481-488, Berlin, Germany, Jul. 2017. (https://doi.org/10.1145/3071178.3071208)

[13] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, C. Fernando, and K. Kavukcuoglu, "Population based training of neural networks," *arXiv:1711.09846,* 2017. (https://doi.org/10.48550/arXiv.1711.09846)

[14] J. Mockus, V. Tiesis, and A. Zilinskas, "The application of Bayesian methods for seeking the extremum," *Toward Global Optim*, vol. 2, pp. 117-129, Amsterdam, Netherlands, 1978. (https://www.researchgate.net/publication/248818761)

[15] M. W. Hoffman, E. Brochu, and N. de Freitas, "Portfolio allocation for Bayesian optimization," in *Proc. UAI 2011*, pp. 327-336, Barcelona, Spain, Jul. 2011. (https://dl.acm.org/doi/10.5555/3020548.3020587)

[16] C. Hvarfner, D. Stoll, A. Souza, M. Lindauer, F. Hutter, and L. Nardi, "πBO: Augmenting acquisition functions with user beliefs for Bayesian optimization," *arXiv:2204.11051,* 2022. (https://doi.org/10.48550/arXiv.2204.11051)

[17] I. Ilievski, T. Akhtar, J. Feng, and C. Shoemaker, "Efficient hyperparameter optimization for deep learning algorithms using deterministic RBF surrogates," in *Proc. 31st AAAI Conf. Artif. Intell. (AAAI)*, pp. 822-829, San Francisco, USA, Feb. 2017. (https://doi.org/10.1609/aaai.v31i1.10647)

[18] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *Proc. LION 5*, pp. 507-523, Rome, Italy, Jan. 2011. (https://doi.org/10.1007/978-3-642-25566-3_40)

[19] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Proc. Adv. NeurIPS 2011*, pp. 2546-2554, Granada, Spain, Dec. 2011. (https://dl.acm.org/doi/10.5555/2986459.2986743)

[20] D. Nguyen, S. Gupta, S. Rana, A. Shilton, and S. Venkatesh, "Bayesian optimization for categorical and category-specific continuous inputs," in *Proc. 34st AAAI Conf. Artif. Intell.*,

pp. 5256-5263, New York, USA, Feb. 2020. (https://doi.org/10.1609/aaai.v34i04.5971)

[21] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," *arXiv:1806.09055,* 2019. (https://doi.org/10.48550/arXiv.1806.09055)

[22] D. Maclaurin, D. Duvenaud, and R. Adams, "Gradient-based hyperparameter optimization through reversible learning," in *Proc. ICML 2015*, pp. 2113-2122, Lille, France, Jul. 2015. (https://dl.acm.org/doi/10.5555/3045118.3045343)

[23] J. Lorraine and D. Duvenaud, "Stochastic hyperparameter optimization through hypernetworks," *arXiv:1802.09419,* 2018. (https://doi.org/10.48550/arXiv.1802.09419)

[24] J. Lorraine, P. Vicol, and D. Duvenaud, "Optimizing millions of hyperparameters by implicit differentiation," in *Proc. 23rd AISTATS 2020*, pp. 1540-1552, virtual, Aug. 2020. (https://dl.acm.org/doi/abs/10.5555/3586589.3586738)

[25] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv:1611.01578,* 2017. (https://doi.org/10.48550/arXiv.1611.01578)

[26] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *JMLR*, vol. 18, no. 1, pp. 6765-6816, Apr. 2018. (https://dl.acm.org/doi/10.5555/3122009.3242042)

[27] J. Wang, J. Xu, and W. Wang, "Combination of Hyperband and Bayesian optimization for hyperparameter optimization in deep learning," *arXiv:1801.01596*, 2018. (https://doi.org/10.48550/arXiv.1801.01596)

[28] N. Awad, N. Mallik, and F. Hutter, "DEHB: Evolutionary hyperband for scalable, robust and efficient hyperparameter optimization, *arXiv:2105.09821,* 2021. (https://doi.org/10.48550/arXiv.2105.09821)

[29] K. Swersky, J. Snoek, and R. P. Adams, "Multi-task bayesian optimization," in *Proc. Adv. NIPS 26*, pp. 2006-2014, Lake Tahoe, USA, Dec. 2013. (https://dl.acm.org/doi/10.5555/2999792.2999836)

[30] I. D. Raji, H. Bello-Salau, I. J. Umoh, A. J. Onumanyi, M. A. Adegboye, and A. T. Salawudeen, "Simple deterministic selection-based genetic algorithm for hyperparameter tuning of machine learning models," *Appl. Sci.*, vol. 12, no. 3, p. 1186, Jan. 2022. (https://doi.org/10.3390/app12031186)

[31] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE Conf. CVPR 2018*, pp. 8697-8710, Salt Lake City, USA, Jun. 2018. (https://doi.org/10.1109/CVPR.2018.00907)

[32] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," in *Proc. ICML 2018*, pp. 4095-4104, Stockholm, Sweden, Jul. 2018. (http://proceedings.mlr.press/v80/pham18a.html)

[33] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le, "Understanding and simplifying one-shot architecture search," in *Proc. ICML 2018*, pp. 550-559, Stockholm, Sweden, Jul. 2018. (https://proceedings.mlr.press/v80/bender18a.html)

[34] X. Qi and B. Xu, "Hyperparameter optimization of neural networks based on Q-learning," *Signal Image Video Process.*, Oct. 2022. (https://doi.org/10.1007/s11760-022-02377-y)

[35] Y. Bengio, "Gradient-based optimization of hyperparameters," *Neural Comput.*, vol. 12, no. 8, pp. 1889-1900, Aug. 2000. (https://doi.org/10.1162/089976600300015187)

[36] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. CVPR 2009*, pp. 248-255, Miami, USA, Jun. 2009.

(https://doi.org/10.1109/CVPR.2009.5206848)

[37] F. Hutter, H. Hoos, and K. Leyton-Brown, "An efficient approach for assessing hyperparameter importance," in *Proc. 31th ICML 2014*, pp. 754-762, Beijing, China, Jun. 2014. (https://proceedings.mlr.press/v32/hutter14.html)

[38] X. Ying, "An overview of overfitting and its solutions," *J. Phys., Conf. Ser.*, vol. 1168, Feb. 2019. (https://doi.org/10.1088/1742-6596/1168/2/022022)

[39] A. Makarova, H. Shen, V. Perrone, A. Klein, J. B. Faddoul, A. Krause, M. Seeger, and C. Archambeau, "Automatic termination for hyperparameter optimization," in *Proc. Int. Conf. Automated Mach. Learn.*, pp. 1-21, Baltimore, USA, Jul. 2022. (https://proceedings.mlr.press/v188/makarova22a.html)

원 종 현 (Jonghyeon Won)

2020년 8월 : 연세대학교 전기전자공학과 졸업

2020년 9월~현재 : 연세대학교 전기전자공학과 석박사통합과정

<관심분야> 최적화 이론, 하이퍼파라미터 최적화, 무인항공기 네트워크, radio access network

[ORCID:0000-0003-3883-0379]

신 종 민 (Jongmin Shin)

2022년 2월 : 연세대학교 전기전자공학과 졸업

2022년 3월~현재 : 연세대학교 전기전자공학과 석박사통합과정

<관심분야> 무선통신네트워크, 하이퍼파라미터 최적화, 네트워크 슬라이싱

[ORCID:0000-0003-1447-7371]

김 재 호 (Jae-Ho Kim)

2017년 8월 : 연세대학교 전기전자공학과 박사 졸업

2000년~2020년 : 한국전자기술연구소 자율지능IoT연구센터장

2020년~현재 : 세종대학교 전자정보통신공학과 교수

2019년~현재 : TTA 사물인터넷/스마트시티 플랫폼 표준그룹(PG1001)의장

2021년~현재 : ITRC 메타버스 자율트윈 연구센터 센터장

<관심분야> 사물인터넷, 자율지능시스템, 스마트시티플랫폼, 디지털트윈 및 메타버스

[ORCID:0000-0001-6597-7988]

이 장 원 (Jang-Won Lee)

1994년 2월 : 연세대학교 전자공학과 학사

1996년 2월 : 한국과학기술원전기 및 전자공학과 석사

2004년 8월 : Dept. of ECE Purdue University 박사

2004년 9월~2005년 8월 : Dept. of EE Princeton University 박사 후 연구원

2005년 9월~현재 : 연세대학교 전기전자공학부 조교수/부교수/교수

<관심분야> 통신망 자원 할당, 통신망 최적화, 통신망 성능 분석

[ORCID:0000-0002-5627-5914]