공 학 박 사 학 위 논 문

AutoML의 성능개선을 위한 Hyperparameter 최적화 연구

2020년 2월

부 경 대 학 교 대 학 원

컴 퓨 터 공 학 과

김 용 훈

공 학 박 사 학 위 논 문

AutoML의 성능개선을 위한 Hyperparameter 최적화 연구

지도교수 정 목 동

이 논문을 공학박사 학위논문으로 제출함.

2020년 2월

부 경 대 학 교 대 학 원

컴 퓨 터 공 학 과

김 용 훈

김용훈의 공학박사 학위논문을 인준함.

2020년 2월 21일

위 원 장 공학박사 여 정 모 (인)

위 원 이학박사 김 태 균 (인)

위 원 공학박사 송 하 주 (인)

위 원 공학박사 한 영 선 (인)

위 원 공학박사 정 목 동 (인)

목 차

그 림 목 차

표 목 차

AutoML의 성능개선을 위한 Hyperparameter 최적화 연구

김 용 훈

부 경 대 학 교 대 학 원 컴 퓨 터 공 학 과

요 약

기계학습 성능을 향상시키기 위하여, 다양한 학습 계층을 설계하거나, 특정 매개변수를 최적화 하는 연구가 활발히 진행되고 있다. 하지만, 기계학습과 관련하여 인문, 사회 및 다양한 위치에서 이와 같은 효율적인 시스템을 적용하고자 하나, 실제 시스템을 구축하는데 있어서, 기존의 연구 결과를 그대로 반영하여 학습하는 방법에는 문제가 있어왔다. 왜냐하면, 데이터의 특성이나, 사용자의 환경에 따라, 학습의 성능을 높이기 위해서는 학습계층을 합리적으로 설계하여야 하며, 또한 각 매개변수에 대하여 적절한 값의 정의가 필요하기 때문이다.

이와 관련하여, 최근 연구는 자율적 기계학습과 관련한 연구가 진행되고 있으며, 자율적 기계학습은 사용자가 학습에 필요한 여러 매개변수 설정 또는 성능향상을 위한 학습계층 설계와 같은 전문적 지식이 부족하여도 학습을 진행하는데 어려움이 없도록 하는 학습방법에 대한 연구를 의미한다. 따라서 자율적 기계학습은 학습에 사용되는 각 매개변수를 자동으로 특정 데이터 및 사용자 환경에 따라 설정하거나, 학습 계층을 구현하는 방법을 해결하는 것이고, 이러한 문제를 해결하기 위하여 자율적 기계학습에서는 크게 세부분으로 연구되고 있다. 첫 번째는 자율적 특징 학습(Automated Feature Learning)이며, 둘째는 학습계층에 대한 자동 설계(Architecture Search), 마지막으로 학습 모델에 대한 매개변수 자동 설정(Hyper-parameters Optimization)이다.

본 논문에서는 학습 모델에 대한 매개변수 자동설정과 관련이 있으며, 매개변수 자동설정은 기계 학습의 성능과 직접적인 관련이 있는 학습률(Learning Rate), Mini-batch size 및 정규화 계수(Regularization Coefficient) 와 같은 Hyperparameters를 말한다. 특히, 학습 성능과 직접 관련된 학습률에 중점을 두고 이를 개선 한다. 따라서 Exploration에 의존적인 구현을 위해 Gaussian 분포 대신 Gamma 분포를 이용한 Bayesian 최적화 기법을 사용한다. 또한, 목적함수의 결과 값을 Convex 한 형태로 제시하는 방법을 통하여 정확한 학습률을 예측함으로써, 최종적으로 성능이 개선되는 시스템에 대하여 설명하고자 한다.

Research on Hyperparameter Optimization

for Improving AutoML Performance

Yong Hoon Kim

Department of Computer Engineering, Graduate School,
Pukyong National University

Abstract

Recently, in order to improve the performance of machine learning, researches are being actively conducted to design various learning layers or to optimize specific parameters. In particular, while attempting to apply such an efficient system in the humanities, society, and various positions in relation to machine learning, there has been a problem in the method of learning to reflect the results of existing researches in constructing the actual system.

This is because, in order to improve the performance of learning according to the characteristics of the data and the user's environment, the learning layers must be reasonably designed, and the definitions of appropriate values are required for each parameter. In this regard, recent research has been conducted on autonomous machine learning, and autonomous machine learning refers to the study of the learning method that does not have difficulty in learning even if the user lacks professional knowledge such as setting various

parameters for learning or designing the learning layer for improving performance. Therefore, autonomous machine learning can be regarded as a method of solving problems by automatically setting each parameter used for learning according to the their environment or providing an algorithm of necessary to autonomously implement the machine learning layers. Therefore, autonomous machine learning is to set up each parameter used for learning automatically according to specific data and user environment, or to solve a method of implementing the learning layers. To solve this problem, autonomous machine learning has been studied in three parts. The first is Automated Feature Learning, the second is Architecture Search for the learning layer, and the last is the Hyperparameters Optimization.

In this paper, we are concerned with the parameter auto-setting for the learning model, which is directly related to the performance of machine learning, the learning rate, mini-batch size, and normalization coefficient. In particular, we focus on and improve learning rates directly related to learning performance. Therefore, the Bayesian optimization method using a Gamma distribution is used instead of Gaussian distribution for implementation dependent on exploration. In addition, we will describe a system that finally improves performance by predicting the correct learning rate through the method which is approximated convex hulls to the result of the objective function.

I. 서 론

서론에서는 본 논문에 대한 전체적인 흐름에 대한 설명과 연구의 필요성 및 목적, 연구의 범위에 대하여 설명하고자 한다.

1.1 연구의 개요

최근 AutoML(Automated Machine Learning)[1]에 대한 논의가 더욱 가속화되고 있으며, 이와 관련하여 AutoWEKA는 기계학습 알고리즘과 Hyperparameter를 동시에 선택할 수 있는 방법에 대하여 서비스를 진행하고 있다. 또한, Auto-sklearn은 AutoWEKA(Auto Waikato Environment for Knowledge Analysis)를 확장하여 분류와 회귀에 대한 라이브러리를 제공하고 있다. 추가로 TPOT(Tree-based Pipeline Optimization Tool)는 유전자 Programing을 사용하여 파이프라인을 최적화하는 연구를 진행하고 있고, 그 외 H2O(Oxdata) AutoML, TransmogrifAI, MLBox등 다양한 시스템에서 이와 관련된 연구를 진행하고 있으며, 또한 라이브러리를 제공하고 있다.

특히, Google에서는 그림 및 동영상과 관련한 Sight, 자연어 분석과 관련한 Language 및 정형화 데이터와 관련된 Structured Data의 세 가지 항목에 대하여 자동화된 기계 학습을 제공하는 Cloud AutoML 사이트를 개설하여 서비스를 진행하고 있다. 하지만, 기계학습의 매개 변수는 많은 테스트에 의해 결정되는, 학습 후 최종 결과로 간주 될 수 있고, 학습 계층구조

도 연구에 대한 결과물이며, 특징 추출과 관련한 다양한 모델 설계 또한 같은 맥락으로 이해할 수 있다. 따라서 AutoML에서는 기계학습을 위한 변수를 미리 추정하거나 검토할 수 있어야 한다.

즉, 학습을 위해 미리 설정해야 하는 모델의 학습률, Mini-Batch 크기 및 정규화 계수에 가장 적합한 값을 결정해야 한다. 그러나 대부분의 경우 기존 연구에서 제공하는 매개 변수가 그대로 사용되고 있다.

이러한 문제를 해결하기 위한 방법 중에서 Grid 검색 및 Random 검색[2]을 사용하여 Hyperparameter를 최적화하는 것을 고려할 수 있다. 그러나 많은 수의 결과 값을 도출하고 비교해야 하는 문제가 있으며, 다른 방법으로는 최소 사전 결과 값으로 적절한 학습 속도를 계산할 수 있는 모델 연구로써 TPE(Tree-structured Parzen Estimator)[3] 연구와 Taub 검색[4]이 있다. 또한, Bayesian 최적화와 관련한 다른 연구의 실험적 설계로 적응형 모델[5-7]에 사용되고 있으며, 최근에는 AutoML에 Bayesian 최적화를 적용한 연구[8-10]가 활발히 진행되고 있다.

Bayesian 최적화는 이전 값을 사용하여 모수 분포를 추정하며, 일반적으로 Gaussian 분포가 사용된다. 이를 Gaussian Process[3, 6]라고 한다. 이를 활용한 Bayesian 최적화는 더 높은 차원에서의 문제점을 해결하기 위한 연구로 mGP(Manifold Gaussian Process)[11]가 있고, 지수 분포를 이용한 최적화 및 학습률과 같은 설정 계수를 자동으로 검색하기 위한 다양한 연구가 수행되고 있다. 일반적인 알고리즘은 손실 값[12]을 출력값으로 사용하는 Hyperparameter 추정이다.

본 논문에서는 Bayesian 최적화를 기반으로 여러 매개 변수를 최적화하며, Gamma 분포를 활용하여 각 Epoch의 학습률을 자동으로 선택하는 방법에 중점을 둔다. 또한 최적화에서 중요시되는 목적함수의 출력값을 손실 값과 정확도 값을 적용하는 방법과 이 두 출력값을 Convex하게 하는 기법

에 대하여 정의하며, 통계적 기법의 검증을 통하여, 기존 연구에서 사용되는 Random 검색 및 Grid 검색과 비교함으로써, 제안하는 기법의 성능이 우수함을 보이고자 한다.

2장에서는 Bayesian 최적화 및 Acquisition Function과 분포에 대한 관련 연구를 설명하며, 3장에서 목적 함수와 목적함수의 출력값과 제안하고자 하는 Convex 기법에 대하여 정의한다. 4장에서는 기존 연구 결과와 제시하는 연구의 검증 방법 및 비교실험을 통하여 결과를 도출하며, 마지막으로 5장에서 결론과 향후 연구 방향에 대하여 논한다.

1.2 연구의 필요성 및 목적

AutoML에 대한 관심은 점점 더 증가하고 있으나, 현재 대부분의 개발 모델 즉, AlexNet[13] 모델과 2011년 이후 CNN(Convolutional Neural Network)을 사용한 다양한 학습 모델에서 사용된 매개변수는 기존 연구에서 정의된 기법을 설정값의 변경 없이 사용하는 경우가 많으며, 표 1은 현재 설계된 몇몇 기계학습 방법에 관련된 매개변수를 보여준다.

표 1 학습 모델별 Hyperparameter의 예

구분	Learning Rate	Epochs	Batch Size	Regularization Coefficient
AlexNet	Gradient Descent(lr = 0.01)	10	128	Random Normalization
GoogleNet	SGD(lr=0.1, decay = 1e-6, momentum=0.9)	Related to steps	64	Alpha = 0.001 beta = 0.75
ResNet	SGD(if batch size <8192, plr = 0.5, epochs=5 elif batch size <16384, plr = 10.0 epochs =5 elif batch size <32768 plr = 25.0 epochs = 5 else plr = 32.0 epochs =1			Mean = 127 Stddev = 60
LSTM	SGD(lr = 20 Setp by step /4.0	Related to steps	20~128	exception

그러나, 기존연구에서 사용된 매개변수를 그대로 사용하면 이전 연구에 사용된 데이터 세트가 실제 데이터 Set과 다르기 때문에 최적의 학습 결과를 도출하기가 어려운 문제가 있다[14].

이러한 문제는 Network 모델에서 학습률만 간단히 변경하여 Loss 값을

관찰하여도 쉽게 알 수 있다.

그림 2은 MNIST 데이터를 이용하여 간단한 DNN(Deep Neural Network) 학습의 결과를 나타내며 SGD0에서 SGD4까지 Random 하게 학습률을 반영한 손실 값의 결과를 보여준다.

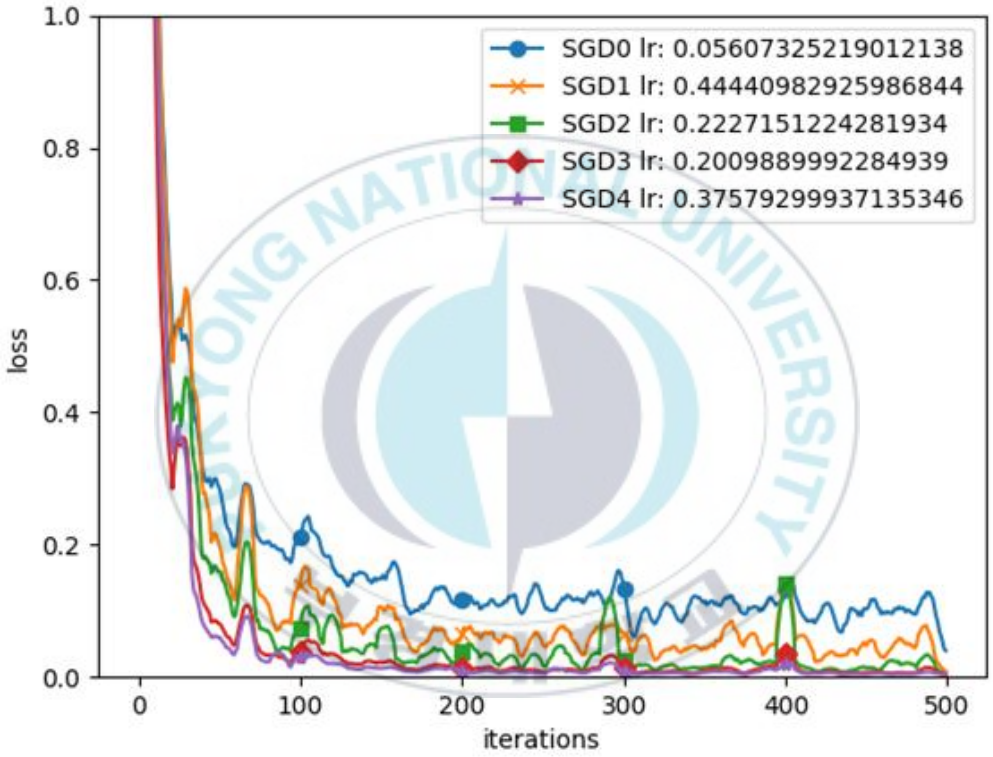

그림 2. 학습률의 차이에 따른 Loss 값의 비교

특히, SGD0과 SGD4의 손실 값 차이가 높음을 알 수 있다. 즉 학습률에 따라 실제 학습 반영은 다르게 나타난다. 그렇다면, 최적의 학습모델을 구현하기 위해서는 학습률의 결정과 반영은 상당히 중요한 문제임을 예상할 수 있다.

이와 함께 그림 1은 각 Epoch에서의 일정한 학습률을 적용하였지만, 표 1의 ResNet과 같이 각 Epoch에서의 다른 학습률을 적용하는 것이 더 좋은 학습모델을 구현 할 수 있음을 알 수 있다.

따라서 본 연구에서는 이와 같은 네트워크 모델에 의존적인 매개변수를 자율적으로 선택하고 제공하는 방법에 대해 제시하고자 하며, Bayesian 최적화의 특성상 여러 번의 Sampling을 진행하게 되지만, 전체 학습 Epoch를 생각하면, 실제 작은 Epoch로 높은 결과를 제공한다. 또한 이러한 선택의 문제에서 더 합리적인 선택을 위하여 목적함수의 출력값을 Convex 한 형태로 제시함으로써, 성능에서 높은 결과를 제공한다.

1.3 연구의 방법 및 범위

AutoML과 관련한 연구는 일반적으로 자동화된 특징 학습(Automated Feature Learning)[15, 16], 학습 계층 구조에 대한 설계와 이를 활용한 성능향상 및 관련한 검색(Automated Neural Architecture Search)[17-19]과 Hyperparameter 최적화가 있다. Hyperparameter 최적화에는 학습 속도, Mini-Batch 크기 및 정규화 계수 최적화가 포함된다.

본 논문의 연구 방법은 네트워크 모델의 Hyperparameter를 최적화하는 것이며, 그중에서 학습률의 설정을 자율적으로 하며, 특히, 학습률을 자율적으로 설정하기 위하여 Bayesian 최적화에서 사용되는 Acquisition Function과 관련하여 Gamma 분포를 적용한다. 또한 Saddle Point에 대한 최적의 추론을 위하여 목적함수의 출력값과 관련된 특정의 함수를 제시하고 정의한다.

연구범위는 본 논문에서 사용되는 Bayesian 최적화와 관련하여 SM(Surrogate Model)은 기존연구에서 설계된 모델을 적용하며, AF(Acquisition Function)를 기초로 하지만, Gaussian 분포와 Gamma 분포로 한정한다. 또한, Gamma 분포를 사용하면서 Conjugate 사전 확률을 이용하여 함수적 형태의 일치성을 이용한다.

이와 함께, 목적함수의 정의와 관련하여 Hyperparameter 모두를 적용하지는 않으며, 가장 직접적인 관계가 있는 학습률을 적용하고, 목적함수의 결과 평가를 위해서 손실 값과 정확도 값을 적용한다.

II. 관련 연구

본 논문에서는 BO(Bayesian Optimization) 기법 및 목적함수의 정의를 기반으로 하며, BO에는 SM(Surrogate Model), AF(Acquisition Function) 및 Gaussian 분포와 관련이 있다. 본 장에서는 BO 최적화와 관련된 연구와 SM, AF에 대하여 설명한다.

2.1 Bayesian 최적화

BO의 가장 일반적인 용도는 목표가 되는 블랙박스(Black Box) 함수[14]와 관련된 전역 최적화 문제를 해결하는 것이다. 여기서 집합 A에 비선형 함수 $f(x)$를 최적화하는 문제는 식 (1)과 같이 간단히 표현할 수 있다.

$$\max_{x \in A \subset \mathbb{R}^d} f(x) \tag{1}$$

특히, Sampling 된 값에서 목적함수를 추정할 수 없거나, 미분할 수 없는 목적함수, non-convex 할 경우 유용하다[20]. 이러한 전역 최적화에 대한 다수의 접근법[21-24]이 현재까지 연구가 활발히 진행되고 있으며, 구간 최적화, 분기 및 바운드 방법과 같은 확률론적 근사법의 기계학습에서 알려지지 않은 목적 함수를 효율적으로 최적화하는 데 효과적이라는 결과[11]가 있다. 따라서 머신 러닝의 Hyperparameter 최적화는 학습 전에 설정된 값

을 참조[20]하며, BO는 이러한 값을 자동으로 설정하기 위한 최적화 방법 중에서 하나의 대안으로 사용할 수 있다.

BO는 Bayes' Theorem을 사용하기 때문에 Bayesian이라 하며, 모델의 증거 E가 주어진 모델 M의 사후 확률은 주어진 E의 주변 확률에 M의 사전 확률을 곱한 값으로 설명할 수 있으며, 식 (2)과 같다.

$$P(M|E) \propto P(E|M)P(M) \tag{2}$$

식 (2)을 기반으로 x_i를 i번째 샘플로 정의하면 i번째 목적함수의 Sampling 값을 $f(x_i)$로 생각할 수 있고, Sample 값 $D_{1:t} = \{x_{1:t}, f(x_{1:t})\}$로 정의하면, 주변 확률을 $P(D_{1:t}|f)$로 정의할 수 있다. 이를 이용하여 식 (3)을 정의 할 수 있다.

$$P(f|D_{1:t}) \propto P(D_{1:t}|f)P(f) \tag{3}$$

식 (3)과 같이 Likelihood $P(D_{1:t}|f)$와 사전확률 $P(f)$로 사후 확률 $P(f|D_{1:t})$를 계산할 수 있음을 알 수 있는데, 이러한 이론이 BO의 기본 이론이 된다.

BO 최적화와 관련하여 연속 확률 변수의 Gaussian 분포는 Parametric 분포의 예이다. Parametric 분포는 Parameter에 의해 분포가 결정되며, Gaussian 분포에서의 Parameter는 평균과 분산을 예로 들 수 있다. 이러한 모델을 밀도 추정 문제에 적용하기 위해서는 관찰된 데이터 집합을 바탕으로 Parameter를 구하는 과정이 필요하게 되며, 빈도적 관점에서는 어떤 특정 기준을 최적화하는 방식으로 Parameter를 찾을 수 있다. 최적화의 방식

으로는 가능도 함수가 있으며, Bayesian 관점에서는 사전 분포를 바탕으로 Sampling 된 데이터 집합이 주어졌을 때 사후 분포를 계산한다[25].

2.2 Conjugate 사전 확률과 Hyperparameter

Conjugate 사전 확률은 사후 확률이 사전 확률과 같은 함수적 형태를 가질 수 있도록 한다. 따라서 Bayesian 분석에서 더 단순한 분포의 함수적 형태로 대체하여 쉽게 계산할 수 있도록 할 수 있다. 일반적으로 다항 분포 매개변수의 Conjugate 사전 확률은 Dirichlet 분포이며, Gaussian 분포 평균값의 Conjugate 사전 확률은 다른 Gaussian 분포로 표현이 가능하다. 이러한 분포들은 모두 Exponential family에 속하고, 이러한 같은 Exponential family에 속하는 분포는 같은 함수적 형태를 가지게 된다.

즉, 특정 형태의 사전확률과 가능도 함수의 곱에 비례하는 사후 분포는 사전분포와 같은 함수적 형태를 가지게 된다[25].

통계학에서의 Hyperparameter는 함수적 형태를 동일하게 만들어 줌으로써 발생하게 되는 부가적인 매개변수를 말하며, 특히 이러한 매개변수가 특정분포를 조절할 수 있게 되는데 이러한 변수를 Hyperparameter라고 한다.

기계학습에서의 Hyperparameter는 목적함수가 되는 학습 함수, 일반적으로 학습 Network모델에 영향을 주는 변수들을 이야기 할 수 있다.

본 연구에서는 목적함수에 영향이 있는 학습률을 Hyperparameter로 설정한다.

2.3 Gaussian Process

일반적인 목적함수는 입력값 x를 입력받아 알 수 없는 목적 함수[26, 27] f를 사용하여 함숫값 $f(x)$를 최대화하는 최적의 결과인 x^*를 찾는 것이다. 그러나 입력값 후보에 대해 함숫값을 순차적으로 검사하고 $f(x)$를 최대화하는 최적의 결과 값 x^*를 찾으려면 두 가지가 필요하다.

첫 번째는 사전에 기대되는 특정 모델에 대한 분포를 정의한, 표준편차가 *1*이며, 평균이 *0*으로 설계된 SM이며, 두 번째는 지금까지 조사 된 입력값과 함숫값을 기반으로 목적 함수를 추정하는 AF로 구성된다.

이를 이용하여 현재까지의 확률적 추정 결과를 기반으로 최적의 결과 값 x^*를 도출하게 된다. 여기서 SM은 GPs(Gaussian Processes)[28]를 적용하여 모델을 설계하게 되며, GP는 SM에서 널리 사용되는 기초적 개념으로 생각할 수 있다. 따라서 GP는 확률 통계에서 일반적으로 사용되는 Gaussian 분포를 이용하여 정의 할 수 있는데, 이와 관련한 식은 (4)와 같다.

$$f(x) \sim gp(m(x), k(x, x')) \tag{4}$$

식 (4)에서 m은 평균, k는 공분산을 나타내며, GP는 다변량 Gaussian 분포를 확률론적 과정으로 확장하여 평균과 공분산으로 지정된 함수에 대한 분포를 의미한다. 즉, GP는 앞으로 설명하게 될 AF의 주변 확률계산을 가능하게 한다.

GP와 관련하여 일반적으로 $m(x)=0$ 으로 하는 Random 한 분포를 정의

할 수 있는데, 사전 평균에 대한 대안적인 방법[29, 30] 또한 연구되었으며, 공분산 k에 대한 정의를 식 (5), (6)과 같이 정의하였다.

$$k(x_i, x_j) = \exp(-\frac{1}{2} \| x_i - x_j \|^2) \quad (5)$$

$$k(x_i, x_j) = \exp(-\frac{1}{2\theta^2} \| x_i - x_j \|^2) \quad (6)$$

식 (5)는 함숫값이 가까울수록 *1*에 가까워지고 멀어질수록 *0*에 가까워지는 결과를 나타내는데, 따라서 가까운 두 지점 간에는 큰 차이를 나타내는 특징이 있다. 식 (6)은 식(5)와 유사하지만, GP의 너비 설정과 관련하여 θ를 이용할 수 있는 장점이 있고, 실제 ARD(Automatic Relevance Determination)[31]에 영향을 주었다. 즉, GP는 사후 확률의 평균과 표준편차의 생성과 관련한 다양한 분포를 설계하는 이론으로써, 연구가 진행되었다.

2.4 Acquisition Function

AF는 BO에서 실제적인 다음으로 예측할 수 있는 최적의 위치를 제공하기 위하여, 확률적 기법을 적용하여 계산되는 함수를 의미한다. 즉, 대부분의 목적함수의 경우 직접 Sampling 하는데 있어서 비용이 크다[26]. 특히 많은 데이터를 사용하는 기계학습의 경우 부담스러운 부분이 많다. 따라서 SM을 이용하여 일종의 가상 목적함수를 만들고 AF를 통하여 사전적 최적의 위치를 검토한 후 실제 목적함수에 적용하게 된다.

이와 관련하여 새롭게 Sampling을 시도할 위치를 추정하는 방법에는 두 가지의 개념을 생각할 수 있는데, Exploitation과 Exploration[32]이 있다.

Exploitation은 현재 위치에서 최적의 Saddle Point가 있을 확률이 높다

는 개념이다. 즉, 현재 Sampling 점이 최적의 Saddle Point라고 가정했을 때, 목적함수의 주변에 더 최적의 Saddle Point가 있을 확률이 다른 불확실한(Sampling이 이루어지지 않아 확인되지 않은) 위치보다 확률이 높다는 뜻이 된다.

Exploration은 전체 범위에서 Sampling이 되지 않은 위치에 최적의 Saddle Point가 있을 확률이 높다는 개념이다. 즉 Sampling이 이루어지지 않음으로 인한 목적함수의 불확실성이 큰 위치가 가장 최적의 Saddle Point가 있을 수 있다는 의미가 있다.

AF에 기본이 되는 PI(Probability Improvement)는 $f(x^+)$의 Sampling 점을 최대화하기 위해 처음 1964년 Kushner에 의해 제안[33]되었으며, 관련 식은 (7)과 같다.

$$\begin{aligned} PI &= P(f(x) \geq f(x^+)) \\ &= \Phi\left(\frac{\mu(x) - f(x^+)}{\sigma(x)}\right) \end{aligned} \tag{7}$$

여기서 $\Phi(\cdot)$는 CDF(Cumulative Distribution function)를 나타내며, 계산식을 *Lipschitz-continuous*로 가정하면 $\sigma(x)$로 나누어 CDF로 계산할 수 있다. 하지만, 식 (7)은 Exploitation에 너무 집중되어 Sampling 된 점 주변을 중심으로 Sampling 되는 문제점이 있었고, 이 문제점을 해결하기 위하여 Trade-off 기법을 사용하게 되는데 관련 식은 (8)과 같다.

$$\begin{aligned} PI &= P(f(x) \geq f(x^+) + \xi) \\ &= \Phi\left(\frac{\mu(x) - f(x^+) + \xi}{\sigma(x)}\right) \end{aligned} \tag{8}$$

$\xi \geq 0$ 로 사용자의 설정에 따라 Exploration을 초기에 진행 할 수 있다. 하지만 사용자의 설정은 현재에도 많은 이슈가 되고 있고, 다른 영역에서

ξ 와 관련한 실험적 영향을 연구[34-36]했다.

하지만, Exploitation에 의존적인 문제는 완전히 해결할 수 없었고, 따라서 추정 중에 과도하게 기존 Sampling 된 지점에서 Sampling을 진행하는 문제점이 있어, 불필요한 자원 소모가 많은 것[20]으로 알려져 있다.

따라서 더 안정적이고, 합리적인 방법에 대한 연구가 진행되었고, 실제 Sampling 된 위치에서 편차가 작은 점을 찾는 방법을 위와 같은 문제의 대안으로 제안되었으며, 실험적 연구는

$$
\begin{aligned}
x_{t+1} &= \arg min \mathbb{E}(\| f_{t+1}(x) - f(x^*) \| \mid D_{1:t}) \\
&= \arg min \int \| f_{t+1}(x) - f(x^*) \| P(f_{t+1} \mid D_{1:t}) df_{t+1}, \\
x_{t+1} &= \arg min \mathbb{E}(\min \mathbb{E}(\| f_{t+2}(x') - f(x^*) \| \mid D_{t+1}) \mid D_{1:t})
\end{aligned}
$$

와 같다. 실험적 연구는 편차를 최소화하면서, 추가적 및 재귀적으로 구현할 수 있는 방향으로 연구가 진행되었으나, 비용이 많이 드는 문제로 실제 사용되지는 못하였고, Mockus에 의해 더 합리적인 방법이

$$
\begin{aligned}
I(x) &= \max\{0, f_{t+1}(x) - f(x^+)\} \\
x &= \arg max_x \mathbb{E}(\max\{0, f_{t+1}(x) - f(x^+)\} \mid D_t)
\end{aligned}
$$

와 같이 제안[37]되었다. *I(x)*는 양의 값을 가지며, 추정값이 알려진 위치의 값보다 큰 경우에만 적용됨을 알 수 있다. *I(x)*를 적용하여 정규분포에 적용하고 그에 대한 기댓값을 유도하는 과정은

$$
\frac{1}{\sqrt{2\pi}\sigma(x)} exp\left(- \frac{(\mu(x) - f(x^+) - I^2)}{2\sigma^2(x)} \right)
$$

와 같이 *I(x)*를 정규분포로 대입한 후

$$\mathbb{E}(I) = \int_{I=0}^{I=\infty} I \frac{1}{\sqrt{2\pi}\,\sigma(x)} exp\left(-\frac{\left(\mu(x) - f(x^+) - I^2\right)}{2\sigma^2(x)}\right) dI$$

$$= \sigma(x)\left[\frac{\mu(x) - f(x^+)}{\sigma(x)} \Phi\left(\frac{\mu(x) - f(x^+)}{\sigma(x)}\right) + \phi\left(\frac{\mu(x) - f(x^+)}{\sigma(x)}\right)\right]$$

과 같이 정의할 수 있다. 결과적으로 1978년 Mockus[37], 1998년 Jones[14]에 의해 최종적으로 식 (9)를 유도하게 되었다.

$$EI = \begin{cases} \left(\mu(x) - f(x^+)\right)\Phi(Z) + \sigma(x)\phi(Z) & \text{if } \sigma(x) > 0 \\ 0 & \text{if } \sigma(x) = 0 \end{cases}$$
$$where\ Z = \frac{\mu(x) - f(x^+)}{\sigma(x)} \tag{9}$$

식 (9)에서 $\Phi(\cdot\ )$은 PDF를 $\phi(\cdot\ )$은 CDF를 나타낸다. 추가로 기존 문제가 되었던 편향적 Exploitation과 관련하여 Global 탐색과 Local 탐색에 대한 합리적인 Trade-off를 2008년 Lizotte[36]에 의해 추가하게 됨으로써, 현재 식 (10)을 일반적으로 사용하고 있다.

$$EI = \begin{cases} \left(\mu(x) - f(x^+) - \xi\right)\Phi(Z) + \sigma(x)\phi(Z) & \text{if } \sigma(x) > 0 \\ 0 & \text{if } \sigma(x) = 0 \end{cases}$$
$$where\ Z = \begin{cases} \frac{\mu(x) - f(x^+) - \xi}{\sigma(x)} & \text{if } \sigma(x) > 0 \\ 0 & \text{if } \sigma(x) = 0 \end{cases} \tag{10}$$

ξ는 식 (8)에 언급하였던 연구와 흡사하고, 이와 관련하여 $\xi = 0.01$일 때 높은 결과를 나타낸다는 연구[35]가 있다. 그림 3와 4[38]은 단순히 BO의 작동 방식을 보여준다. 여기서 1차원 연속 입력에 대해 예측되는 것으로 가정하며, 그림 3는 목적 함수를 나타내고 그림 4은 AF를 나타낸다.

추정해야 할 목적 함수는 최초에 두 개의 점을 설정하고 두 개점의 결과

값을 바탕으로 지속해서 x를 추가하면서 value를 계산한다.

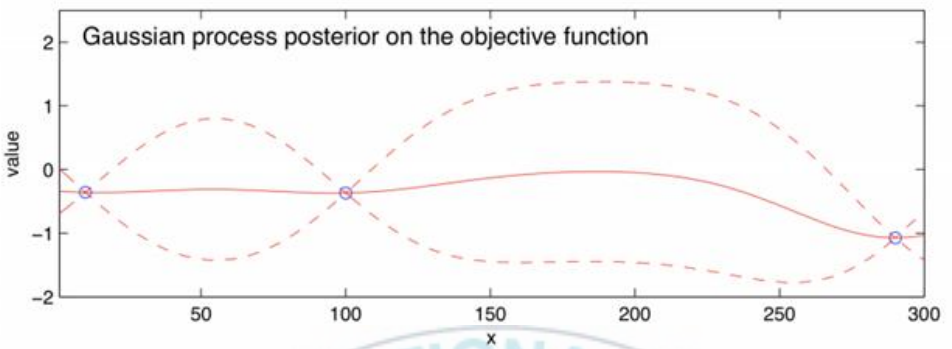

그림 3. 1차원에서의 연속입력에 대한 Gaussian Process 예

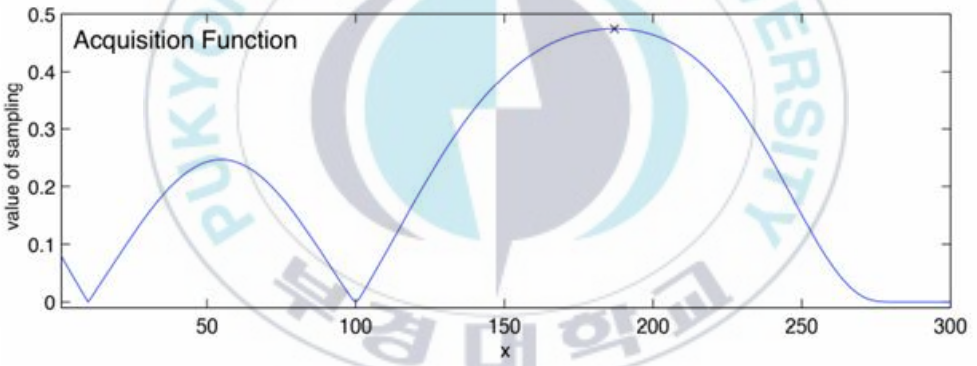

그림 4. 1차원에서의 연속입력에 대한 Acquisition Function 예

그러나 이미 설명한 바와 같이, 이 방법은 시간 및 성능 측면에서 높은 비용을 수반한다.

따라서 실제 목적함수를 통하여 추정하기 전에 AF를 이용하여 추정작업을 선행한다. AF의 Sampling 값 (실선의 높은 지점)은 그림 3의 점선영역의 분산과 평균을 이용하여 계산된다.

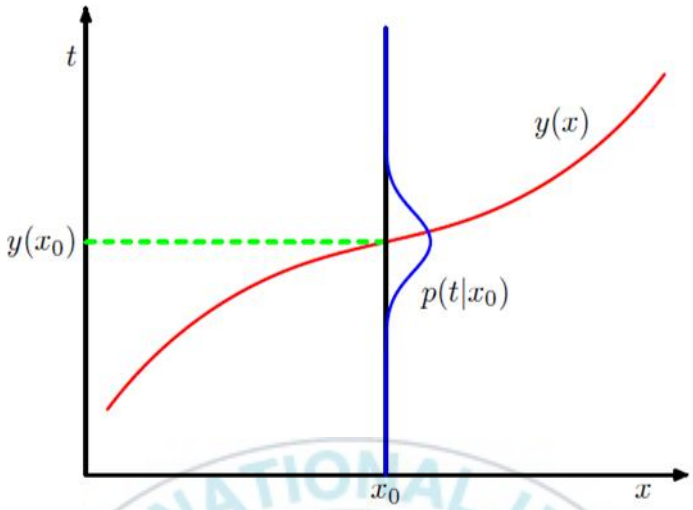

그림 5. Regression 함수의 예

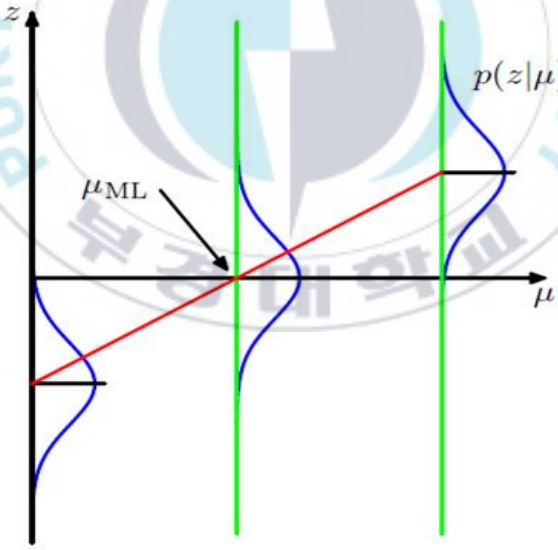

그림 6. 최대 가능도 함수를 이용한 Regression Function의 예

즉, 그림 5[25]와 같이 Regression Function을 계산할 수 있음을 알 수

있다. 따라서 분산은 평균의 중심점과 관계가 있다.

그림 6[25]는 Gaussian 분포를 이용한 최대 가능도 함수를 나타내며, 각 분산과 평균은 SM을 이용하고 최적의 지점인지 확인한다. 그림 4의 현재의 최고점이 가장 높은 점수 후보로 판단되면, 다음 후보 지점을 결정하기 위해 최종 결과가 실제 계산된다. Saddle Point가 아닌 경우 첫 번째 프로세스로 다시 돌아가며, 이 프로세스의 장점은 목적 함수를 직접 계산하지 않아도 된다는 것이며, AF만 계산하여 목적 함수의 분포 유형을 예측할 수 있음을 알 수 있다.

2.5 기계학습에서의 Optimizer

기계학습에서의 Optimizer는 기본적인 경사법 (Gradient Method) 및 확률적 경사하강법(Stochastic Gradient Descent), Momentum, Adam(Adaptive Momentum), AdaGrad(Adaptive Subgradient), RMSprop (Root Mean Square Propagation)이 대표적이다.

2.5.1 경사법 및 경사하강법

경사법은 기계학습에서 가장 기초적인 최적화 기법의 하나다. 경사법은 현재의 위치에서 기울어진 방향을 향하여 특정 거리만큼 이동하는 반복적인 작업이라 설명할 수 있다. 경사에서 최댓값 및 최솟값에 따라 경사 하강법, 경사 상승법이라 할 수 있으며, 관련 식은

$$x_0 = x_0 - \eta \frac{\partial f}{\partial x_0},$$
$$x_1 = x_1 - \eta \frac{\partial f}{\partial x_1}$$

과 같다. 식에서 η는 학습률을 나타내며 각 변수의 편미분을 통하여 갱신하게 된다.

2.5.2 확률적 경사하강법

확률적 경사하강법은 손실함수가 큰 방향으로 이동하는 방법으로 설명할 수 있으며, 식은

$$W \leftarrow W - \eta \frac{\partial L}{\partial W}$$

와 같다. 식에서 W는 갱신할 가중치의 매개변수를 말하며, η는 학습률을 의미한다. $\partial L/\partial W$는 W에 대한 손실 함수의 기울기이다. 즉 가중치를 계속해서 갱신하는 구조로 되어 있다. 학습 모델이 비등방성 함수(방향에 따라 기울기가 달라지는 함수)의 경우 방향을 인식할 수 없는 문제로 비효율적인 진행을 하게 되는 경우가 발생하게 된다.

2.5.3 모멘텀

모멘텀은 운동량을 뜻하는 단어로써, 물리와 관계가 있다. 즉 모멘텀 기법은 별도의 변수를 설정하고 기울기 방향으로 힘을 받는 형태의 기법을 적용하며 식은

$$v \leftarrow \alpha v - \eta \frac{\partial L}{\partial W}$$
$$W \leftarrow W - v$$

와 같다. 식에서 v는 Velocity를 뜻하며, 인스턴스 변수 v가 물체의 속도이고, 초기화 때는 별도의 값을 지정하지 않는다. SGD보다 x축 방향으로 빠르게 진행하는 특징이 있다.

2.5.4 AdaGrad

AdaGrad는 학습률에 직접적인 영향을 주는 Optimizer이며, 실제 Decay라는 매개변수를 설정하고 이 값에 따라 학습률을 줄여가는 방식으로 설명할 수 있으며, 식은

$$h \leftarrow h + \frac{\partial L}{\partial W} \odot \frac{\partial L}{\partial W}$$
$$W \leftarrow W - \eta \frac{1}{\sqrt{h}} \frac{\partial L}{\partial W}$$

과 같다. 식에서 h는 기존 기울기 값을 제곱해서 계속 더해주는 형태로 진행된다. ⊙값은 행렬의 원소별 곱을 의미한다. 또한 갱신할 때 $1/\sqrt{h}$을 통하여 학습률을 조정하고 있는데, 학습률 감소 값이 매개변수의 원소마다 다르다는 의미가 되고, 또한 계속해서 과거의 기울기를 더해가기 때문에 학습이 진행될수록 갱신 강도가 낮아짐을 알 수 있다. 결국 갱신 값이 0이 되어 갱신되지 않게 된다.

2.5.5 Adam

Adam은 AdaGrad와 Momentum을 같이 적용한 형태의 방식으로 이해할 수 있다. 주요 특징은 Step-size가 기울기에 영향을 주지 않는다. 따라서 어떠한 환경에서도 안정적인 하강이 가능하도록 한다.

III. Gamma를 이용한 최적화 시스템

본 절에서는 Gamma를 이용한 최적화 시스템에 대한 이해도와 시스템에 적용된 목적함수 및 Gamma 분포의 적용 방법에 대하여 설명한다. 특히, 목적 함수에 대한 정의와 Gamma 분포를 적용하기 위한 Conjugate 사전 분포에 대한 자세한 설명을 포함한다. 또한 본 절에서는 Gamma 분포를 이용하는 경우와 Gaussian 분포를 적용한 경우를 간단한 실험을 통하여 비교한다.

3.1 시스템 구조

현재 구현되는 시스템은 그림 7과 같이 구성될 수 있고, 그림 7은 본 연구를 위하여 별도로 시스템을 구성한 것이다.

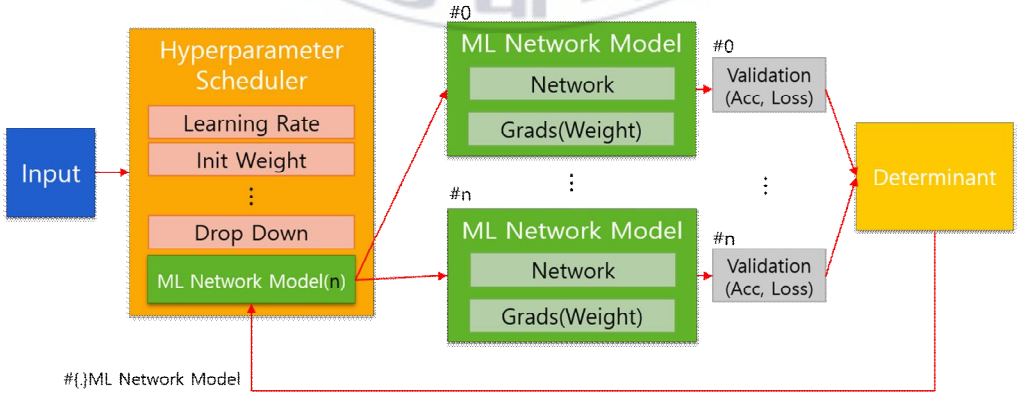

그림 7. Random 검색의 기본 구조

대부분의 현재 시스템은 학습률의 최적화와 관련하여 학습 네트워크에 포함하여 같이 운영되는 것이 일반적이다. 하지만, 본 연구의 목적은 최적의 학습률을 검색하기 위한 연구이고, 이를 위하여 구조를 외부에서 접근하는 방식으로 설계하였다.

특히, 그림 7에서는 Grid 검색 방법을 적용하기 위한 설계이다. 먼저 데이터가 입력되면 Hyperparameter Scheduler는 초기화된 학습 네트워크 모델을 생성한다. 그리고 Grid 검색 개수만큼 기계학습 네트워크 모델을 생성하게 되고, 각 모델은 학습을 진행하게 된다. 이에 대한 결과로 Acc(Accuracy)와 Loss 값을 출력한다. 이러한 결과 값을 Determinant에 의하여 최적의 결과 값 즉, Acc가 높고 Loss가 낮은 최적의 기계학습 네트워크 모델을 결정하고, 결정된 모델의 Grads(Weight) 값을 Hyperparameter Scheduler에 있는 초기 생성된 기계학습 네트워크 모델의 Grads에 Update한다. 이렇게 1 epoch가 완성되는 구조이다.

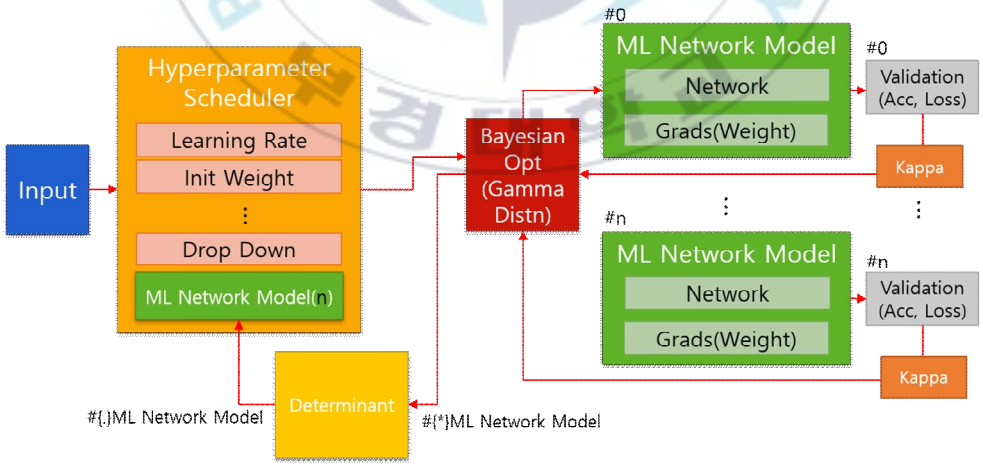

그림 8. 제안하는 시스템의 구조

하지만, 본 연구에서는 BO를 통한 최적의 학습률을 계산하고 이를 통한 기계학습 네트워크 모델로 Update해야 함으로 그림 8과 같이 구조가 바뀌게 된다.

그림 8에서 추가된 Bayesian Opt는 기존 Grid 검색 방법과 달리 순차적인 구조로 κ의 값을 바탕으로 다음의 최적 학습률이 있을 것으로 예상되는 Point를 생성하고 생성된 Point에 해당하는 학습률로 다시 변경하여 학습을 진행하게 된다. 따라서 기존 Determinant에 전달되는 검증데이터가 Bayesian Opt에 전달되고 Bayesian Opt는 이에 대한 값을 바탕으로 최적의 값을 계속 유추하여 Sampling을 진행하고, 이에 대한 결과 값을 모두 저장하여 Determinant에 전달한다.

Determinant는 전달받은 모든 기계학습 네트워크 모델 중에서 최적의 기계학습 네트워크 모델을 결정하고 이를 Hyperparameter Scheduler에 있는 기계학습 네트워크 모델의 Grads에 Update를 진행한다.

3.2 목적함수

이전 연구에서 BO를 활용한 목적함수의 출력값에 정확도 값을 설정하고, 이 값을 이용하여 최적의 학습률을 추정한 연구 외에, 추가로 손실 값을 적용하였다. 기존 연구의 문제점은 정확도 값의 민감도로 인해 학습 속도의 작은 변화에도 불구하고 값이 크게 변하는 것이며, 그래프가 전체적으로 안정적이지 않다는 것을 의미하기 때문이다. 그리고 SGD(Stochastic Gradient Decent)에서 학습률을 적용하는 방법으로 진행하였다. 학습에서 일반적으로 사용하는 모델이며, 실제 Bayesian 추론과 관계가 있기 때문이다.

하지만, 결과는 기존 정확도보다 좋지 않은 결과를 가져왔다. 그 이유는 손실 값은 낮을수록 학습 성능이 높게 평가되기 때문이며, 또한 손실이 낮다고 해서 반드시 높은 정확도를 보장하지 않기 때문이다. 손실 값 추가로만 문제를 해결할 수 없음을 알 수 있었고, 기존연구에서는 식(11)과 같이 손실 값과 추정 정확도를 고려한 방법을 제안하였다.

$$\kappa = \begin{cases} \lambda M_a(x) - \log(M_l(x)) & \text{if } M_l(x) > 0 \\ 0 & \text{if } M_l(x) \leq 0 \end{cases} \tag{11}$$

식(11)에서 M_a는 정확도, M_l는 손실, lambda는 *0*에서 *1* 사이의 조정 된 값으로 삽입되고, *0*에 가까운 변별력을 높이기 위하여 *log*는 M_l에 적용되었다.

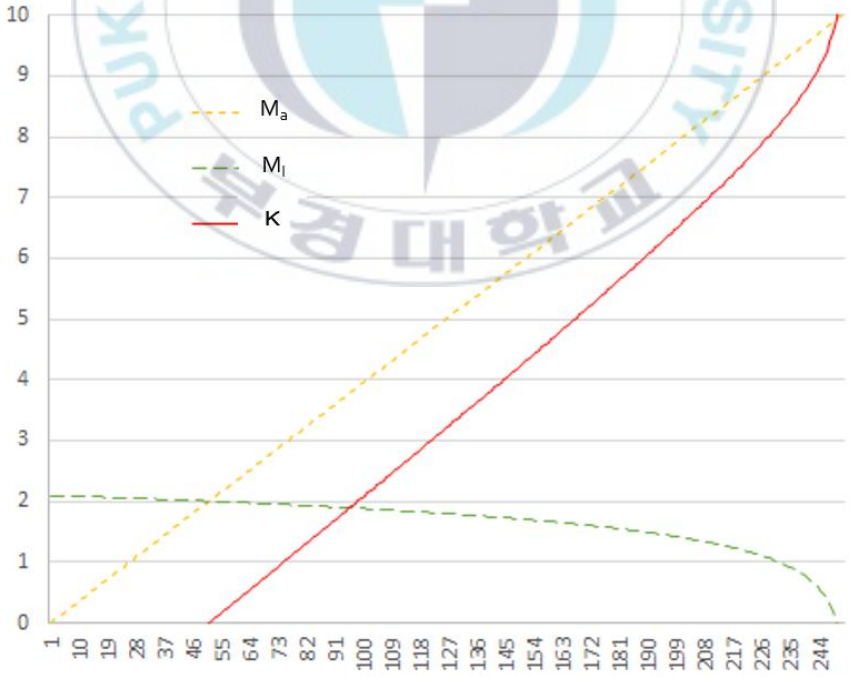

그림 9. 기존 목적함수 출력값에 대한 그래프(κ)

그리고 M_l 함수의 값이 가장 낮고 M_a 함수의 값이 가장 높은 시점이 최대가 되는 값을 최적의 값으로 추정하기 위하여, M_a에서 M_l를 뺀 값이 최댓값일 때를 학습이 잘된 것으로 판단할 수 있다. 하지만, M_l 값이 M_a 값보다 큰 경우 음의 값이 도출될 수 있고, 이것을 고려하여 추가적인 조건을 작성하였으며, 그림 9과 같다.

그림 9에서 짧은 파선은 M_a를 긴 파선은 M_l을 나타내며, 실선은 κ값을 나타낸다. κ의 결과 값은 M_a에 매우 의존적이다. 따라서 M_l의 값의 변동에 따른 값의 변화폭이 작았다. 이와 같은 문제를 해결하기 위하여, 그림 9 및 식 (12)와 같이 개선하였다.

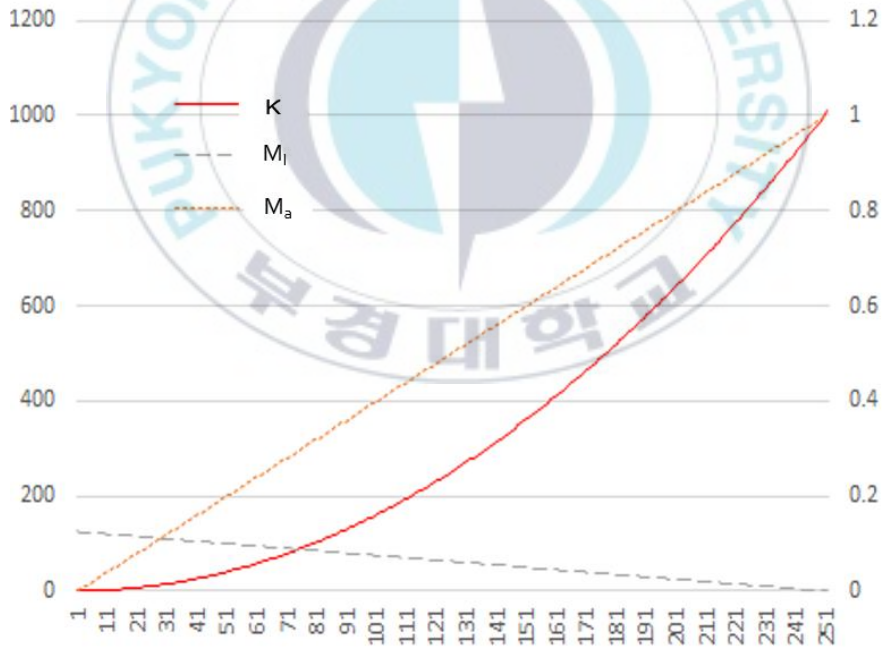

그림 10. 개선된 목적함수 출력값에 대한 그래프(κ)

그림 9의 개선된 목적함수의 경우 M_l에 대한 추가적인 변별력이 높아졌

음을 알 수 있고, 또한 κ의 값도 Convex 한 형태로 변화되었다.

$$\kappa = \begin{cases} \lambda M_a(x)^2 - \dfrac{\log(M_l(x)}{\lambda}) & \text{if } M_l(x) > 0 \\ 0 & \text{if } M_l(x) \leq 0 \end{cases} \tag{12}$$

개선된 식에서는 M_a에 자승 값을 주고, M_l에 λ를 나누는 형태로 변경하였고, 본 연구에 이와 같은 개선식을 적용한다.

3.3 Gamma 분포를 활용한 Bayesian 최적화

본 연구는 BO의 대리 모델이 아니라 AF과 관련이 있다.

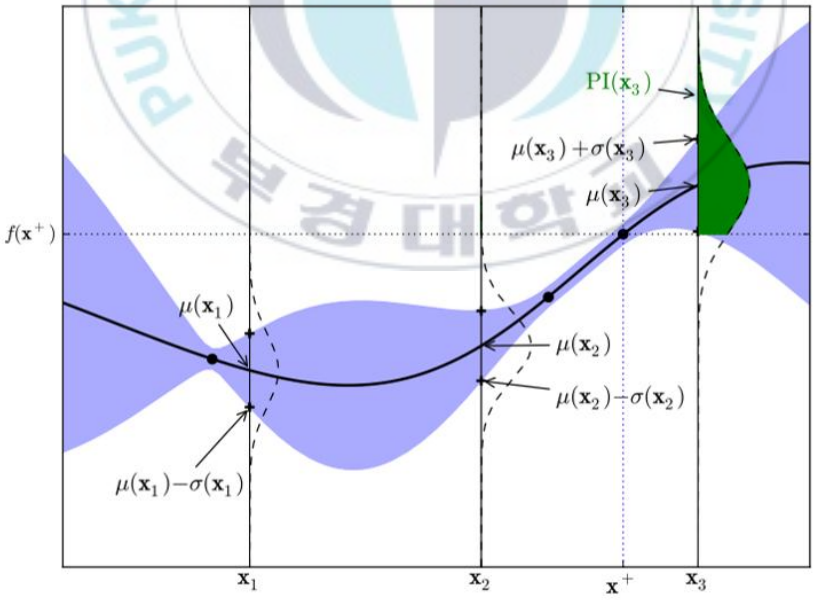

그림 11. Probable Improvement의 계산 영역[26, 38, 40]

기존 PI에서는 그림 11과 같이 SM의 분포를 참조하고 다음으로 높은 값을 가질 것으로 예상되는 영역을 검색한다. 그림 11의 PI는 CDF에 대한 결과 값을 나타낸다. 최대 Sampling 된 위치는 $x+$이며, 이미 실제 Sampling 된 x_3에서 CDF가 가장 높은 값을 계산하게 되면 $x+$로 결론 내릴 수 있다. 관련 연구에서 설명한 것처럼 이러한 경우 Exploitation에 의존적으로 실제 지점이 Local Point라고 가정했을 때, 다른 포인트를 검토하지 못하고 계속해서 x_3 지점 근처에서만 목표지점을 찾는 문제점이 있다. 따라서 이를 개선한 방법이 EI이며, EI는 2장 3절 식 (10)과 같이 기댓값을 계산하는 방식이며, 정규분포를 기저로 가정하고 있다.

그림 12은 이와 같은 Gaussian 분포를 사용하는 GP를 이용하여 함수 식 $f(x) = \sin(3x) - x^2 + 0.7x$에 대한 실험을 보여주며, 관련 실험은 Martin Krasser의 Blog를 참조하였다.

실험은 식 (10)의 AF를 이용하여 최적의 위치를 찾고 그 위치에서 실제 목적함수의 값을 산출하게 된다. 즉, 첫 번째 회차에서는 기본적으로 기준이 될 수 있는 두 개의 위치를 선정하고 AF를 이용하여 목적함수를 근사하게 된다. 근사 된 식을 바탕으로 현재에서 가장 높은 값이 있을 것으로 예상되는 점을 추정하고 추정된 위치에서 실제 목적함수에 입력하여 결과를 도출하게 된다. 그림 12과 같이 최종 10번의 추정을 통하여 현재 식에서의 최적의 점을 찾을 수 있음을 알 수 있다.

하지만, 2장의 설명과 같이 현재와 같이 추정하게 되면 적어도 기본적인 추정 2번을 포함하여 총 8번에서 9번의 실제 목적함수에 대입하여야 하는 문제점이 발생한다.

즉 현재 예시와 같은 단순한 목적함수의 경우 그 비용이 적을 수 있다. 하지만 대부분의 기계학습 모델에서의 네트워크 모델은 각 1 epoch에서 많은 시간적 비용이나, 자원의 비용이 크다. 따라서 더 Exploration에 적합

한 모델을 검토하게 되었으며, 이와 같은 문제점을 해결하기 위하여 기존 MSE(Mean Square estimation)을 사용하고 있는 BO에선 MLE(Maximum likelihood Estimation)으로 교체하는 방법을 적용하였다. 이와 관련하여 Gaussian Conjugate prior distribution을 적용하였고, 식 (13)과 같다.

$$
\begin{aligned}
p(\mu) &= N(\mu_0, \sigma_0^2) \\
&= \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{(\mu-\mu_0)^2}{2\sigma_0^2}\right) \\
p(x_1,...,x_N|\mu) &= \prod_{i=1}^{N} N(x_i|\mu) \\
&= \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{(x_i-\mu)^2}{2\sigma^2}\right) \\
p(\mu|x_1,...,x_N) &\propto p(x_1,...,x_N|\mu)p(\mu) \\
&\propto \exp\left(-\frac{(x_i-\mu'_0)^2}{2\sigma'^2_0}\right)
\end{aligned}
\tag{13}
$$

식 (13)에서 $p(\mu)$는 사전확률이며, $p(x_1,...,x_n|\mu)$는 가능도 함수이고, $p(\mu|x_1,...,x_n)$는 평균에 대한 사후 확률을 의미한다. 단 모두 독립적인 정규분포를 가정한다. Gaussian Conjugate Prior Distribution을 유도하면 식은

$$
\begin{aligned}
\mu'_0 &= \frac{\sigma^2}{N\sigma_0^2+\sigma^2}\mu_0 + \frac{N\sigma_0^2}{N\sigma_0^2+\sigma^2}\frac{\Sigma x_i}{N} \\
\frac{1}{\sigma'^2_0} &= \frac{1}{\sigma_0^2} + \frac{N}{\sigma'^2}
\end{aligned}
$$

와 같다.

μ'_0 와 $1/\sigma^2_0$으로 갱신된 Hyperparameter를 가지는 정규분포가 됨을 알 수 있다. 하지만, 사전 연구된 식 (13)과 관련하여 SM과의 의존 문제 때문

에 기대하였던 결과를 얻지 못하였고, 이러한 의존 문제를 해결하기 위하여 기존 SM에서 사용하던 분산 대신 평균을 사용하는 방법을 적용하였으며, 식 (14)와 같다.

$$
\begin{aligned}
p(\mu|D) &\propto \left[p(\mu)\prod_{n=1}^{N-1}p(X_N|\mu)\right]p(X_N|\mu) \\
p(X|\lambda) &= \prod_{n=1}^{N}N(x_n|\mu,\lambda^{-t}) \propto \lambda^{N/2}\exp\left\{-\frac{\lambda}{2}\sum_{n=1}^{N}(x_n-\mu)^2\right\}
\end{aligned}
\tag{14}
$$

식 (14)에서 정확도(Precision)는 $\lambda \equiv 1/\sigma^2$을 의미하며, 정확도에 대한 사후 확률 분포는 Conjugate prior distribution으로 Gamma 분포를 사용할 수 있는데, 이와 관련한 식은

$$
Gam(\lambda|a,b) = \frac{1}{\Gamma(a)}b^a\lambda^{a-1}\exp(-b\lambda)
$$

$$
\mathbb{E}|\lambda| = \frac{a}{b}, var|\lambda| = \frac{a}{b^2}
$$

$$
\Gamma(a) = \int_0^\infty u^{x-1}e^{-u}du
$$

와 같으며, 식 (14)를 *Gamma(a_0, b_0)*로 다시 정리하면 식 (15)와 같다.

$$
\begin{aligned}
p(X|\lambda) &\propto \lambda^{a_0-1}\lambda^{N/2}\exp\left\{-b_0\lambda-\frac{\lambda}{2}\sum_{n=1}^{N}(x_n-\mu)^2\right\} \\
a_N &= a_0+\frac{N}{2} \\
b_N &= b_0+\frac{1}{2}\sum_{n=1}^{N}(x_n-\mu)^2 = b_0+\frac{N}{2}\sigma_{ml}^2
\end{aligned}
\tag{15}
$$

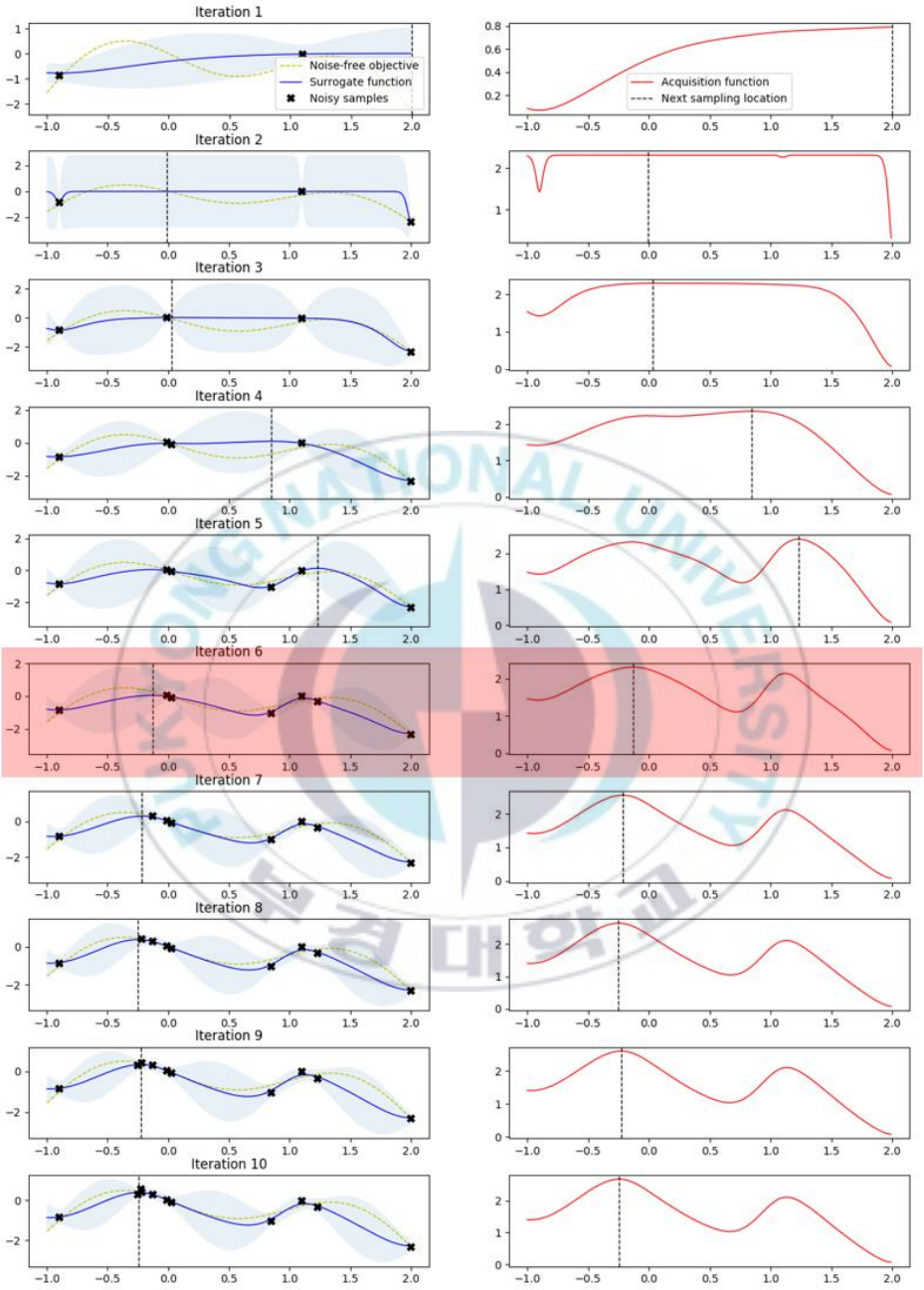

그림 12. Gaussian 분포를 이용한 Gaussian Process 예

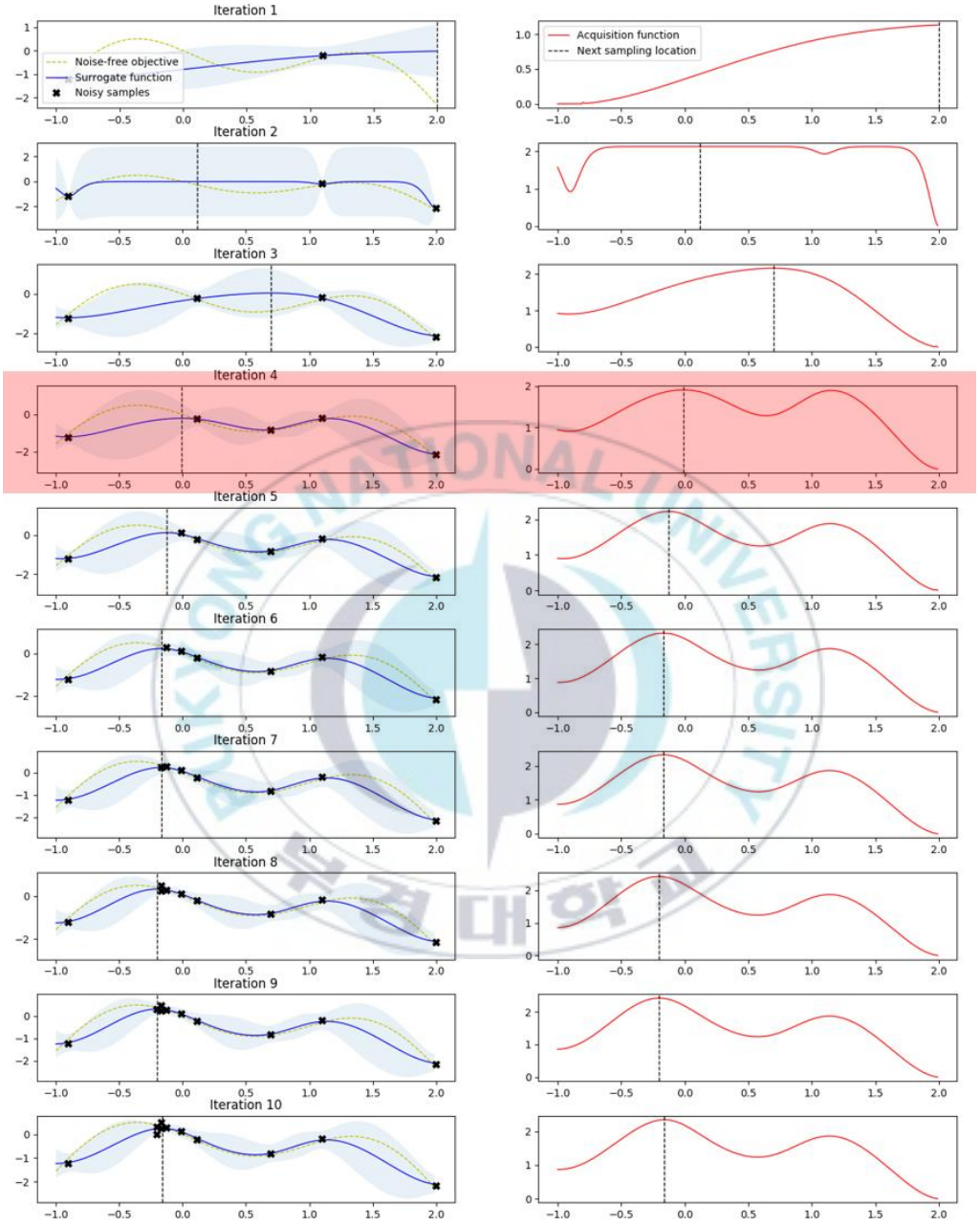

그림 13. Gamma 분포를 이용한 Gaussian Process 예

이를 이용하여 분산을 추정하고, 추정된 분산을 이용해서 다시 평균을

추정하게 된다. 이렇게 추정된 평균을 바탕으로 10회 걸쳐 최적의 Saddle Point를 유추하게 된다. 이러한 방식은 Gaussian의 Bayesian 추론[41]에서 정확도에 대한 사후 확률 분포에 사용되는 것이며, 이를 활용하였다.

그림 13는 Gamma 분포를 활용한 GP를 나타내며, 추정에서 최소 5~6번으로 결과 값을 도출한다. 빠른 결과 값의 도출은 시간적, 자원 비용에 이점이 있다.

IV. 최적화 시스템 실험 및 평가

실험은 MNIST 데이터를 이용한 비교검토와 CIFAR-10 데이터를 이용한 실험을 진행하며, 이와 관련된 평가 방법과 평가에 대하여 설명한다.

4.1 Gamma 분포와 Gaussian 분포 적용의 비교 실험

이전 연구와 같이 Grid 검색, Random 검색 및 Bayesian 최적화를 MNIST(Modified National Institute of Standards and Technology) 데이터에 적용하고 Gaussian Process of Bayesian 최적화를 Gaussian 및 Gamma 분포와 비교했다.

표 2. MNIST학습과 관련한 CNN 네트워크 사양

Circumstance of Test	Contents
Data Set	500 images(28 *28)
Verification Data	20% of Total data set
Hidden Layer	6
Loss Fucntion	Cross Entropy Error
Optimizer	SGD
Activative Function	ReLu
Initial Value of Weight	Kaiming He
InputSize/OutputSize	784/10
Mini-Bach Size	50
Weight_decay	0.001
Dropout Ratio	0.5

또한 이전 연구에서 손실 값과 정확도 값을 함께 적용하는 방법으로 MNIST 모델 목적 함수를 제안하였고, 본 논문에서는 이러한 목적함수의 출력값에 대하여 추가로 개선한 함수를 제안하였다. 속도에 대한 테스트를 위하여 훈련 데이터의 개수를 500개로 제한하고, 검증 데이터는 20%로 설정하였다. 일부 프로그램의 그래프 부분은 Martin Krasser의 블로그를 참조하였고, GPR(Gaussian Process Regression)은 scikit-learn 패키지와 PyOpt를 활용하였다.

MNIST를 학습하기 위한 일반적인 CNN의 학습에 대한 사항은 표 2와 같으며, 시스템 사양은 표 3과 같다.

표 3. 실험 시스템 사양

Category	Contents
OS	Ubuntu 18.04 LTS
Processor	Intel 4.1GHz Core i7-8750H(6 Core)
Graphics Coprocessor	NVIDIA Geforce GTX 1050 Ti
RAM	32GB
Model	Dell XPS 15 9570

그림 14은 0.0~0.5 범위에서 100개의 학습률을 사용하는 Grid 방식의 MNIST 학습 모듈의 그래프를 나타낸다. 그림 14의 상단은 각 학습률에 대한 정확도를 나타내며, 하단의 각 학습률에 대한 손실 값을 그래프로 나타낸 것이다.

가중치 초깃값 설정과 관련하여 정규분포를 사용하거나, Xavier 및 Kaiming He 등의 방법이 있고 이와 관련하여 Random 값을 적용함에 따라, 그래프는 일부 차이를 나타낼 수 있다. 하지만, 대체로 그림 14과 같은 결과를 나타낸다.

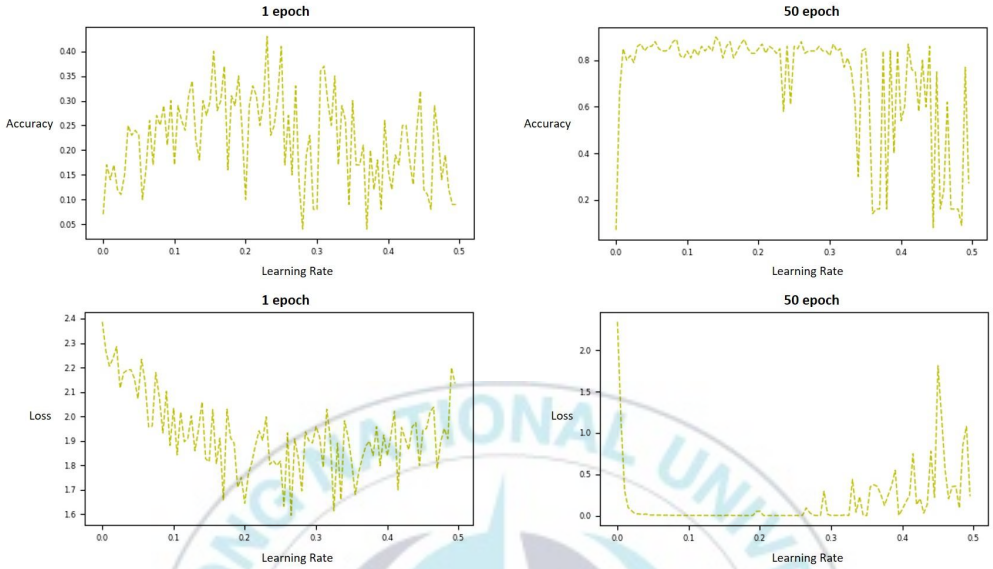

그림 14. 1 Epoch 및 50 Epoch에 해당하는 학습률에 대한 100개의 Grid로 표현된 정확도 및 손실 값의 예

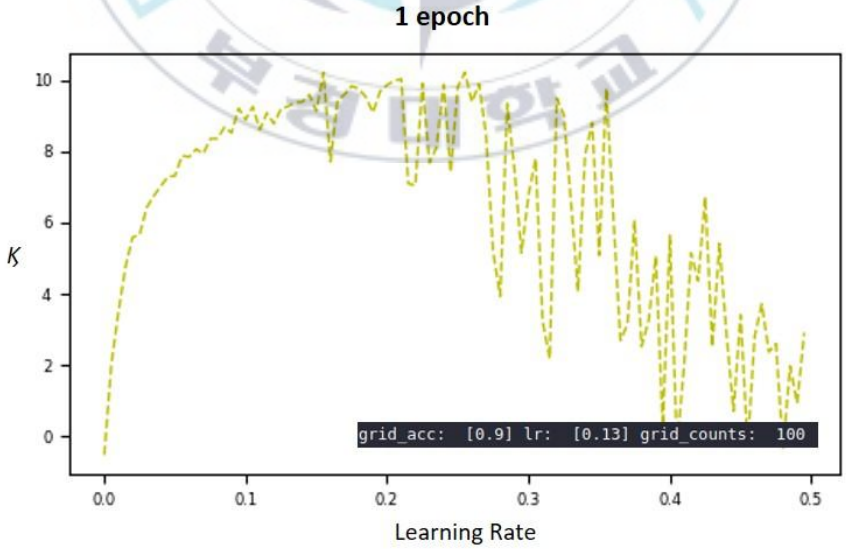

그림 15. MNIST에서 κ를 고려한 출력 값의 예

그림 14에서 알 수 있듯이 잘 훈련된 모델은 가장 높은 정확도를 나타내며, 낮은 손실 값을 나타낸다고 볼 수 있다. 중요한 것은 그래프에 Local Point가 많았으며, 이러한 Local Point를 극복하고 Saddle Point를 찾는 것이 중요하다. 일반적으로 기계 학습 모듈의 평가는 손실 값을 사용한다. 그러나 본 연구에서는 그림 15와 같이 정확도 값과 손실 값을 함께 고려하였다.

그림 15에서 최고 정확도는 90이고 해당 학습률은 0.13임을 확인을 할 수 있다. 즉 보통의 경우 0.01로 학습을 진행한다. 기존 손실 값만 사용하거나 정확도 값만 사용하는 경우와 비교하면, 학습률이 0.2 이후에 κ의 높은 값이 없음을 확인 할 수 있다. 두 개의 값을 고려하면 목적 함수의 Local 최댓값을 최소화 할 수 있다. 이를 통해 목적 함수를 더 쉽게 추정할 수 있는데, 실제 비선형 식에서 Convex 한 형태를 이룬다고 가정할 경우 최댓값 및 최솟값을 쉽게 구할 수 있는 예와 같다.

따라서 본 연구에서 제시된 방법은 매우 중요하다고 볼 수 있다. 또한 LR 0.2 이후의 그래프는 문제가 있는 것으로 보이지만, 학습률이 0.13에서 가장 높은 정확도를 나타내면 제외될 수 있다. 하지만, 추가적인 연구가 필요한 부분이다.

그림 16는 다음 절에서 Gaussian 분포와 Gamma 분포의 비교와 관련이 있으며, MNIST를 목적 함수로 사용하여 1 epoch에서의 결과를 보여준다. 왼쪽은 Gaussian 검색의 사용한 결과를 보여주고 있으며, 오른쪽은 Gamma 검색의 사용 결과를 보여준다. 제안된 방법의 정확도는 47%였으며 검색 반복 횟수는 정확도 대비 가장 적었습니다.

Gaussian 검색을 사용한 방법은 39%이고 검색 반복 횟수는 9이며, 마지막으로 Grid 검색 방법의 경우 정확도는 46%이고 반복 횟수는 100이다. 즉, 제안하는 Gamma 검색에서 정확도의 차이가 7%이고 반복 횟수의 차

이가 1이므로 유사한 정확도 수준에 필요한 반복 횟수는 Gaussian 검색보다 낮았다.

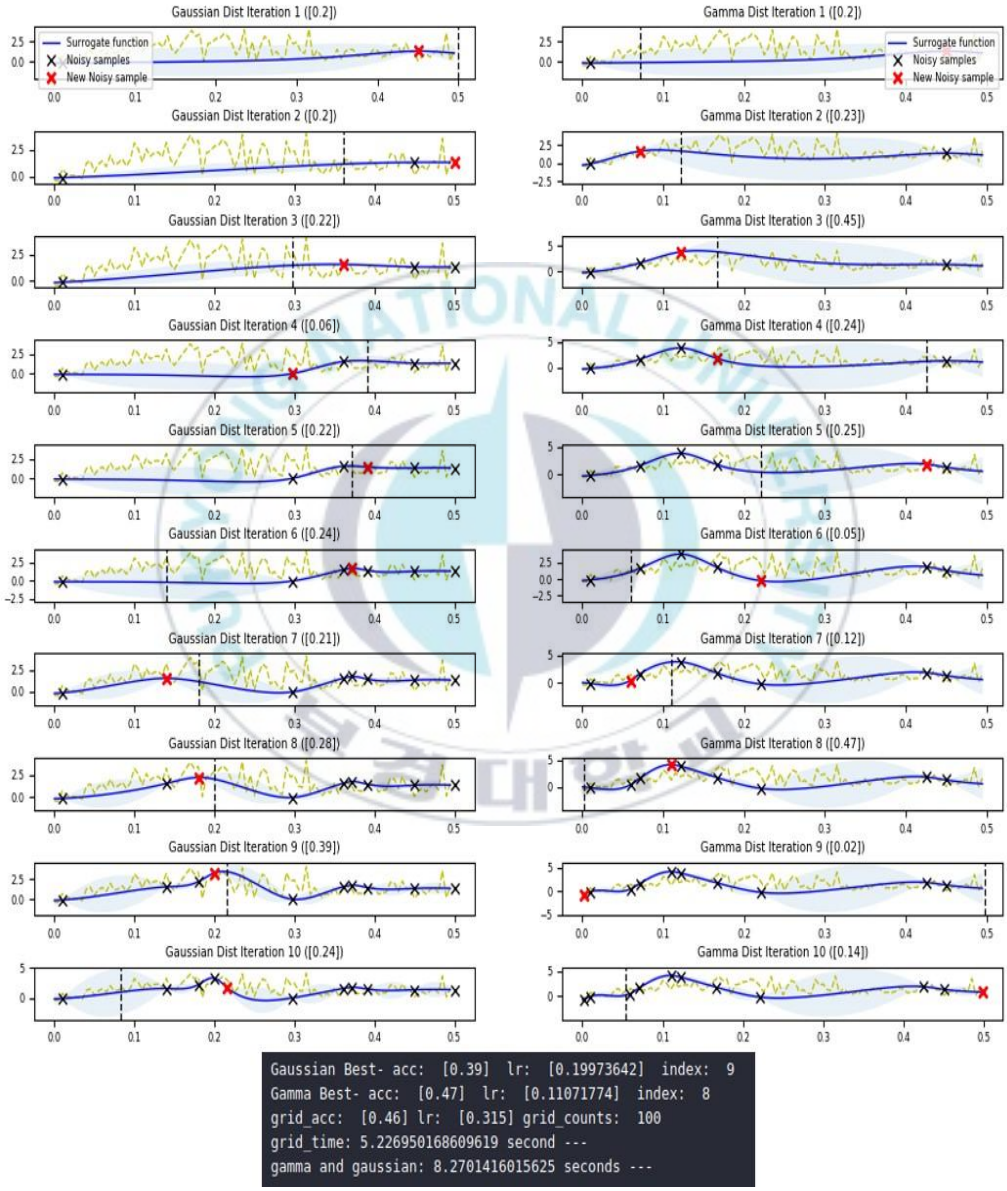

그림 16. 1 epoch에서의 Gamma 검색과 Gaussian 비교

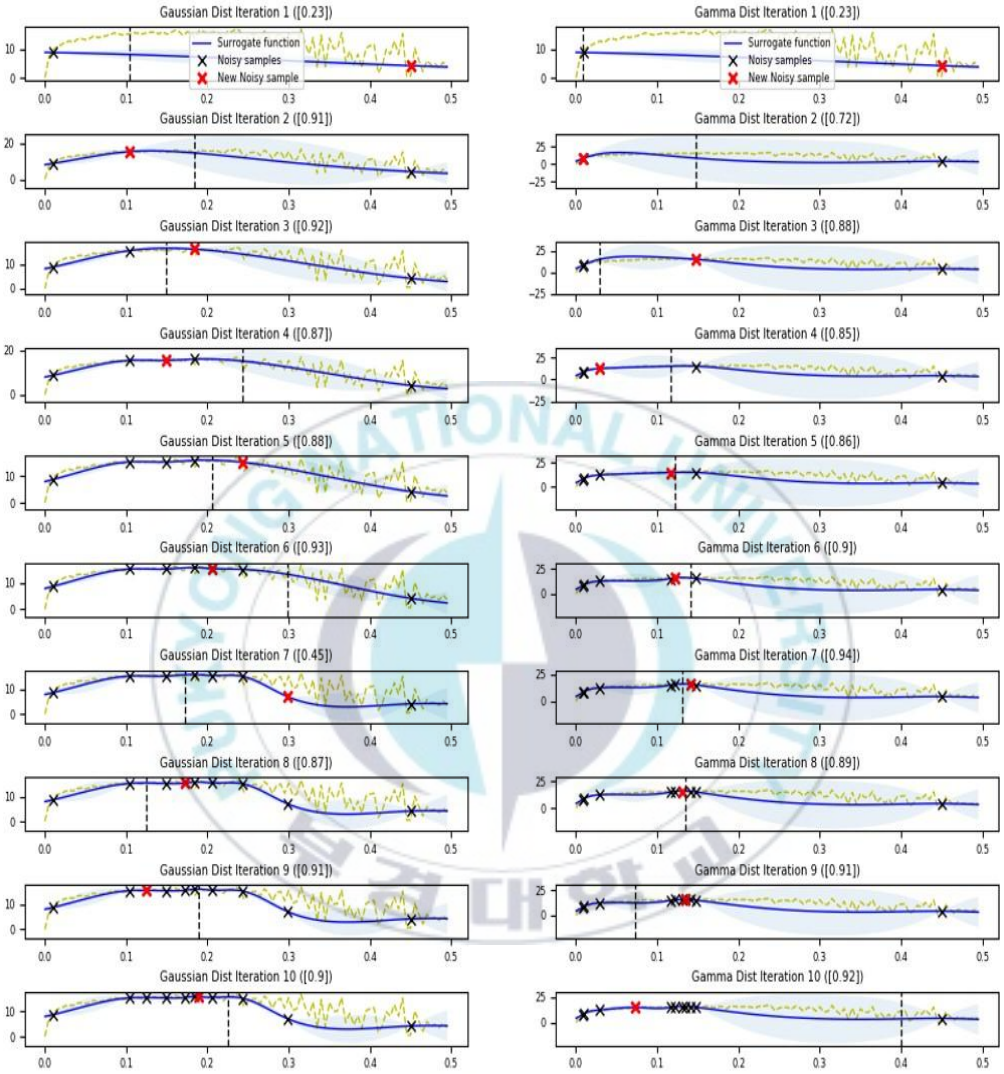

그림 17. 50 epoch에서의 Gamma 검색과 Gaussian 비교

또한, 제안된 방법의 학습률은 약 0.11이었으며, Grid 검색은 약 0.32의

결과를 보여 주었다.

이 경우, 학습률의 최적 위치가 0.11인지 또는 약 0.32인지 확실하지 않다. 하지만, 특정 학습률 값 이후에 로컬 최댓값이 다시 자주 발생하는 것을 확인한 결과, 정확한 Saddle Point는 0.11로 추정 할 수 있다.

본 논문의 목적 중에서 학습속도와 관련하여, 최소 개수의 추정값을 사용하여 MNIST 데이터의 목적 함수에서 Saddle Point를 찾는 것을 확인할 수 있는데, 2장 2절의 Sin 함수를 이용한 비교와 비슷한 결과를 보여준다. 또한, 그림 17은 50 epoch에서의 결과를 나타내는데 그림 17과 같이 Gamma 분포를 사용한 결과가 빠른 추정이 가능하고, 또한 정확도도 높음을 알 수 있다. 무엇보다 Grid 검색보다 높은 정확도를 보여 주고 있는데 이러한 이유는 Grid의 경우 일정한 간격으로 학습률을 결정하고 학습을 진행하게 되는데, 앞서 그림 16의 경우와 같이 Local Point가 많은 지역에서의 Grid 검색은 실제 Saddle Point에 해당하는 위치를 검토하지 못할 수도 있기 때문이다. 그림 17에서 제안된 기법의 정확도는 94%이고 학습률은 약 0.12이고, Gaussian 검색의 경우 정확도는 93%이고 학습률은 약 0.2이다. Grid 검색의 경우 정확도는 93%이고 학습률은 0.16이다. 결과적으로 Grid 검색의 경우 약 100회의 반복이 필요했지만, 제안된 방법은 약 7회의 반복으로도 정확도가 더 높은 학습 결과를 나타냈다.

4.2 시스템 실험 및 정량적 평가

성능 평가 측면에서, 모델의 예측 성능을 측정하는 데 사용되는 평가 지표에는 분류성능평가지표의 민감도, 특이도, 정확성 및 Mathew Correlation coefficient(MCC)가 포함되며 식 (16, 17, 18, 19)과 같다.

$$Sensitivity = \frac{TP}{TP + FN} \tag{16}$$

$$Specificity = \frac{TN}{TN + FP} \tag{17}$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{18}$$

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{19}$$

본 연구는 다음 식 외에도 학습에 사용된 시간을 측정하기 위하여 정확도가 일치하는 수준에서 측정하는 방법으로 평가를 진행하였으며, Grid 검색, Gamma 검색 및 Gaussian 검색 방법은 기존 방법과 비교하였다. 하지만, Random 검색은 방법의 불확실성이 크고, 실제 비교하기에 어려움이 있어 제외하였다. 실제 Random 검색 방법은 학습 속도 측면에서 우수할 수 있지만, 많은 데이터를 학습할 때 epoch 또는 단계 수가 무한대로 증가할 수 있다. 즉, 한계 값에 대한 정의를 할 수 없는 상태에서 검토 횟수를 한정한다는 것은 무리가 있다.

그리고 평가는 속도에 대한 평가와 정확도에 대한 평가로 나누어 진행하였다.

4.2.1 정확도 평가

표 4, 5와 그림 18, 19은 기존 연구, 기존 κ 및 개선된 κ 값의 유무에

따라 평가 방법을 적용하여 비교한 것을 보여준다.

표 4는 기존 시스템과 제안한 시스템의 κ 개선 전 시스템의 비교 평가를 보여준다. 개선 전의 경우 이전 연구에서 개선 전 κ를 적용하여 목적 함수에서 더 많은 볼록한 출력값을 도출함으로써 성능을 향상할 수 있었다. 하지만, 추가적인 학습을 진행하지는 않았다. 평가된 데이터가 작은 문제점도 있지만, 더 이상의 정확도 및 평가 값에 영향을 미치지 않는 수준에서 계속해서 진행됨을 확인하였기 때문이다.

표 4. 기존시스템과 κ 개선 전 시스템의 비교

Epoch	구분	기존 시스템				제안한 시스템(κ 개선 전)			
		Sens	Spec	Acc	MCC	Sens	Spec	Acc	MCC
1	Grid	43.69	93.94	47.0	35.61	46.98	94.61	52.0	38.20
	Gaussian	42.78	93.42	42.0	35.48	43.05	94.40	50.0	41.23
	Gamma	44.02	94.00	46.0	38.90	47.46	94.42	50.0	39.89
2	Grid	90.04	98.67	88.0	87.61	91.76	99.02	91.0	89.50
	Gaussian	89.37	98.55	87.0	86.86	88.70	98.70	88.0	85.92
	Gamma	89.38	98.56	87.0	86.83	90.13	98.80	89.0	87.36

표 5. κ 개선 전 및 κ 개선 후 시스템의 비교

Epoch	구분	κ 개선 전				κ 개선 후			
		Sens	Spec	Acc	MCC	Sens	Spec	Acc	MCC
1	Grid	46.98	94.61	52.0	38.20	46.88	94.20	52.0	37.03
	Gaussian	43.05	94.40	50.0	41.23	42.02	94.47	51.0	40.12
	Gamma	47.46	94.42	50.0	39.89	46.92	94.01	51.0	39.81
2	Grid	91.76	99.02	91.0	89.50	90.56	98.72	90.0	89.50
	Gaussian	88.70	98.70	88.0	85.92	88.70	99.21	88.0	85.92
	Gamma	90.13	98.80	89.0	87.36	90.13	98.90	91.0	87.36

그러나 κ를 적용함으로써 기존 시스템과 비교할 때 정확성, 감도 또는 특이성에서 높은 값을 확인할 수 있습니다. 2 Epoch에서 Grid 검색이 0.3% 정도 높은 값으로 학습됨을 확인 할 수 있었다.

표 5는 κ를 개선 전 및 후에 대한 비교를 나타낸다. 전체적으로 평가가 개선됨을 확인할 수 있다. 특히 Gamma 검색에서 기존 Grid 검색보다 0.2% 높은 정확도가 나왔음을 확인할 수 있었다.

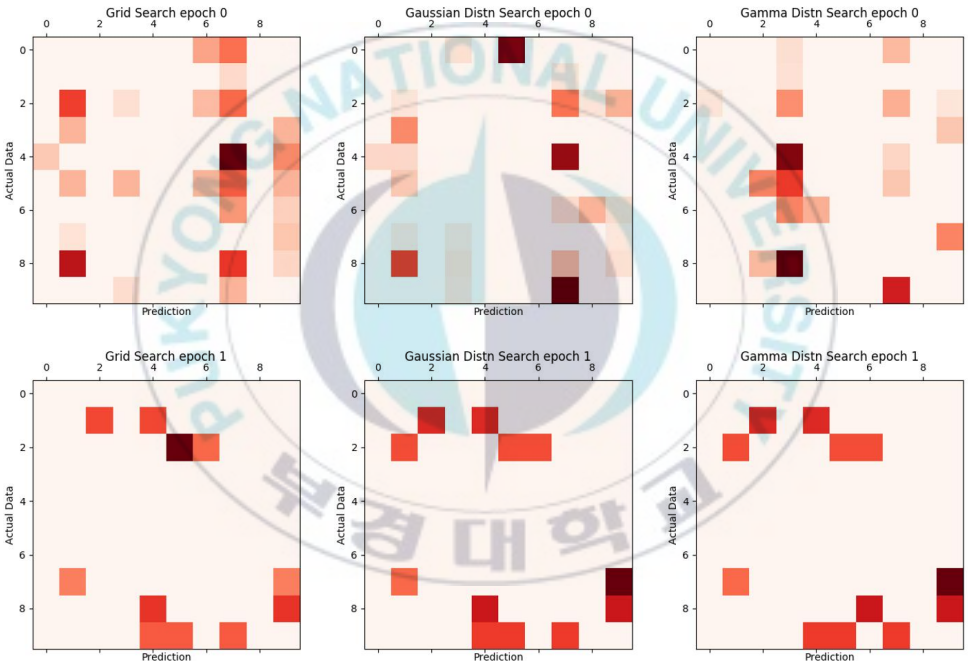

그림 18. 기존 Grid, Gaussian, Gamma 검색의 분석도

그림 18과 19은 표 4에 대한 결과를 시각적으로 나타낸 것이며, 세로축은 실제 레이블을 나타내고 가로축은 학습에서 추론된 결과 값을 나타낸다. 색이 진할수록 일치하지 않음을 나타낸다. 즉, Grid 검색 0 epoch에서 가로 4와 세로 7번의 색이 가장 진한 빨간색으로 볼 수 있는데, 이러한 의

미는 실제 숫자 4에 대한 그림을 추정하였을 때 7로 오 분류됨을 나타낸다. 따라서 1 epoch에서의 Grid 검색의 경우 숫자 2를 5로 오 분류한 경우가 많으며, Gaussian 및 Gamma의 경우는 실제 7의 숫자를 9로 오 분류한 경우가 많음을 알 수 있다. 시각화에서는 특정 숫자에 대한 추론이 정확하게 되어있는지 확인하는 도구로 활용이 가능하다. 즉 대부분의 숫자에 대한 추론은 정확하고 일부분은 서로 다른 점을 보이지만, 전체 정확도에서는 Gamma 분포를 사용한 경우가 높음을 알 수 있었다.

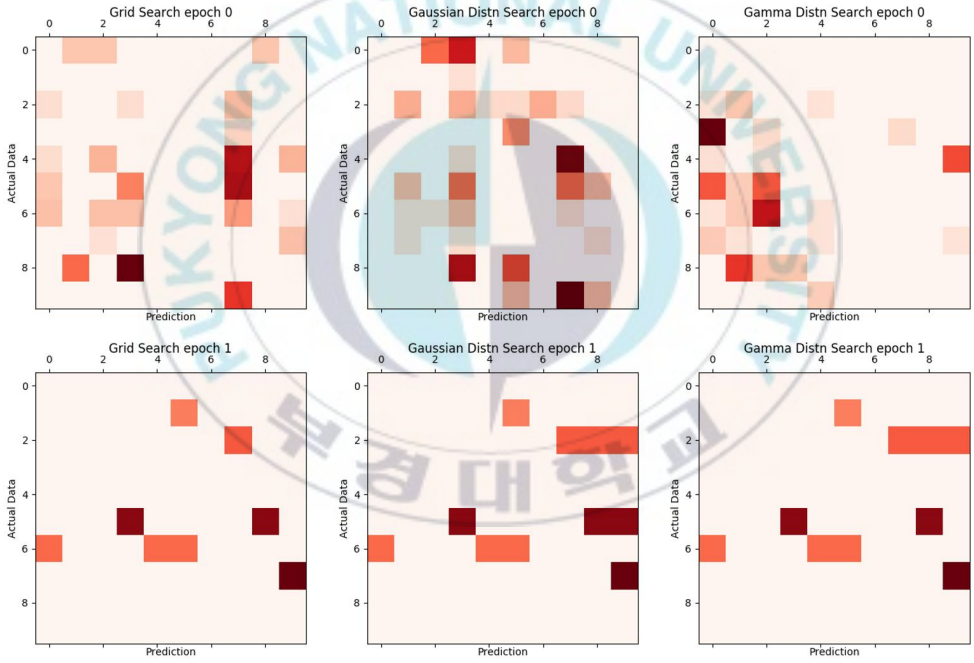

그림 19. κ를 적용한 추정에 대한 비교 시각화

그림 19 역시 그림 18과 같이 잘못 분류된 이미지에 대한 시각화를 나타내며, 1 epoch에서 κ를 적용하기 전과 다르게 Grid 및 Gamma 검색의 잘못 분류된 결과가 거의 비슷한 것을 확인 할 수 있고, Gamma의 경우 실

제 숫자 5를 8로 잘못 분류한 경우가 더 많음을 알 수 있다. 이러한 결과는 표 5에서와 같이 정확도가 떨어짐에 따른 결과로 볼 수 있다.

표 6은 60,000건의 훈련 데이터와 10,000건의 검증 데이터를 사용한 학습 결과를 epoch 단위로 보여준다. 학습 결과로부터 κ 개선 전의 경우 Gamma 검색 방법의 2 Epoch에서 최대 98.36%를 보여주고 있다. 하지만, κ 개선 후 더 높은 결과를 확인할 수 있다. 표 6에는 이전 연구와 비교되지 않았지만, 결과는 κ가 없는 경우 더 낮은 결과 값을 나타내었다. 일반적인 CNN 모델을 사용하여 이와 같은 높은 결과를 나타낸다는 의미에서 더 높은 연구 결과가 있는 CNN 모델을 적용할 경우 더 좋은 결과를 기대할 수 있다.

표 6. 60,000개의 학습 데이터 및 10,000개의 검증 데이터 적용 시 κ 개선 전 및 κ 개선 후 시스템의 비교

Epoch	구분	평가 방법			
		Sens	Spec	Acc	MCC
1	Grid	95.95	99.51	95.62	95.10
	Gaussian	95.73	99.53	95.75	95.25
	Gamma	95.61	99.52	95.64	95.12
	κ개선Gamma	96.21	99.49	96.24	96.01
2	Grid	98.15	99.80	98.18	97.97
	Gaussian	98.11	99.79	98.13	97.94
	Gamma	98.35	99.82	98.36	98.17
	κ개선Gamma	98.61	99.90	98.62	98.56
3	Grid	97.81	99.76	97.83	97.58
	Gaussian	98.19	98.21	98.21	98.00
	Gamma	98.21	99.80	98.23	98.03
	κ개선Gamma	98.77	99.81	98.79	98.51

표 6에서의 1 epoch는 실제 Gaussian 및 Gamma에서는 12 epoch를 생각할 수 있다. 왜냐하면, Gaussian 및 Gamma의 경우 Sampling을 진행하게 되는데, 현재의 설정된 Iteration은 10로 되어 있다. 따라서 기본적인 2개의 Sampling을 포함하면 1 epoch에 12번의 Sampling이 진행됨을 알 수 있다.

따라서 표 6에 대한 결과는 총 24 epoch의 학습에 대한 정확도를 나타낸다고 볼 수 있다. 일반 학습의 경우 50 epoch를 생각할 때 많은 시간 또한 줄일 수 있음을 알 수 있다. 그림 20는 표 6에 관한 결과를 보여준다.

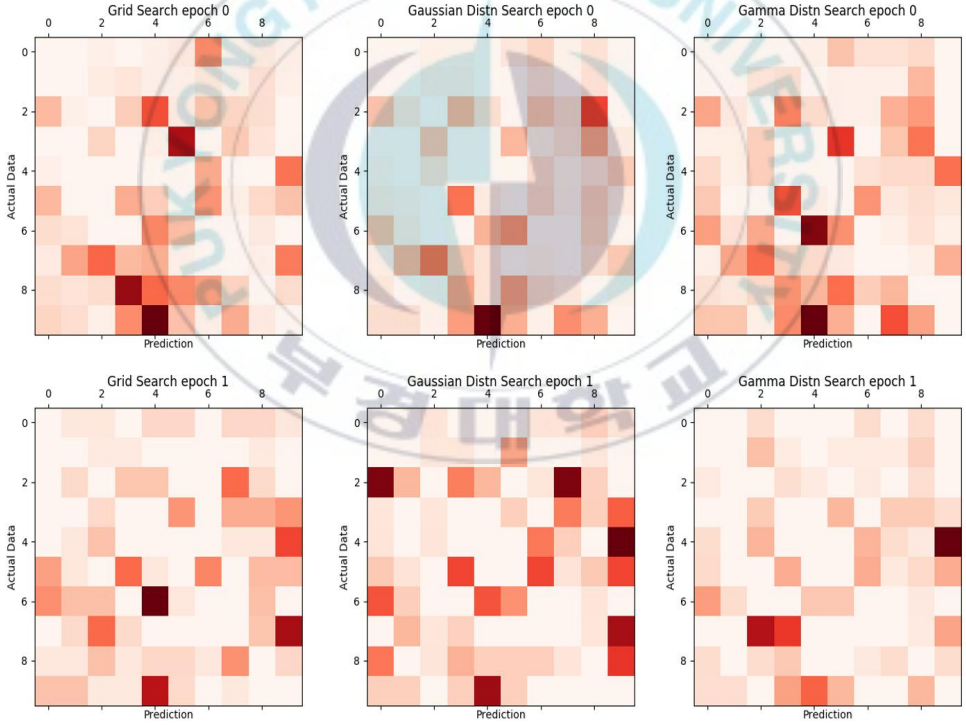

그림 20. 60,000개의 학습 데이터 및 10,000개의 검증 데이터 적용 시 κ 개선 전 및 κ 개선 후 시스템의 비교 시각화

표 7. CIFAR-10 데이터를 이용한 κ 개선 전 및 κ 개선 후 시스템의 비교

Epoch	구분	평가 방법				
		Sens	Spec	Acc	MCC	Loss
1	Keras CNN Grid	48.76	94.41	49.09	42.69	1.4160
	Keras CNN Gaussian	32.37	92.37	29.95	21.83	1.8700
	Keras CNN Gamma	46.69	94.03	45.79	39.25	1.4728
	Keras CNN-κ개선Gamma	45.64	92.41	44.74	40.21	1.3701
10	Keras CNN Grid	73.36	96.80	70.72	68.19	0.8619
	Keras CNN Gaussian	56.64	95.20	56.30	51.14	1.2100
	Keras CNN Gamma	74.35	97.06	73.07	95.12	0.7942
	Keras CNN-κ개선Gamma	76.41	96.54	75.42	79.23	0.7421
20	Keras CNN Grid	76.32	97.29	75.30	72.73	0.7133
	Keras CNN Gaussian	65.43	96.18	65.30	95.25	0.9930
	Keras CNN Gamma	76.71	97.36	76.06	73.48	0.6998
	Keras CNN-κ개선Gamma	78.05	97.92	78.21	77.64	0.6214
50	Keras CNN Grid	76.76	97.31	75.55	73.15	0.7362
	Keras CNN Gaussian	63.84	95.98	63.45	59.19	1.0278
	Keras CNN Gamma	83.22	98.10	82.78	80.92	0.5169
	Keras CNN-κ개선Gamma	84.01	95.25	83.04	82.20	0.4951

표 7은 CIFAR-10 데이터를 사용하여 각 epoch에서의 평가를 진행한 결과를 나타낸다. 적용된 모델은 Keras에서 제공하는 모델을 적용했으며 최대 50 epoch에서 79%의 학습 정확도를 보장한다. CIFAR-10 데이터와 모델을 비교 값으로 설정한 이유는 Keras에서 제공하는 같은 모델을 이용하여 더 높은 정확도를 산출하게 된다면, 제안하는 모델이 우수함을 보일 수 있기 때문이다. 결과적으로 κ 개선 전에서 82.78%의 정확도를 얻을 수 있

었고, κ 개선 후에서는 83.04%의 정확도를 확인 할 수 있었다.

본 연구에서 제시된 방법은 모델 유형과 관계없이 정확도가 증가함을 나타내며 민감도와 특이도 또한 높은 결과를 얻을 수 있음을 확인 할 수 있었다.

그림 21은 각 epoch에 대한 평가 결과를 시각화 한 것이며, 10개의 이미지를 대상으로 학습을 진행하는 결과를 나타낸다. 정확도와 비슷하게 Grid 검색의 경우와 Gamma 검색에서 흰색에 가까운 값이 많이 나타남을 알 수 있는데, κ 개선 전과 후를 비교하였을 때, 다른 이미지를 추가적인 추론한 결과를 나타내지는 못하는 것을 확인 할 수 있었다. 단지 기존 추정에 실패하였던 이미지 중 일부를 정확하게 추정함으로써, 시각화에서의 차이는 κ 개선 전과 후에서 작은 차이를 확인할 수 있었다.

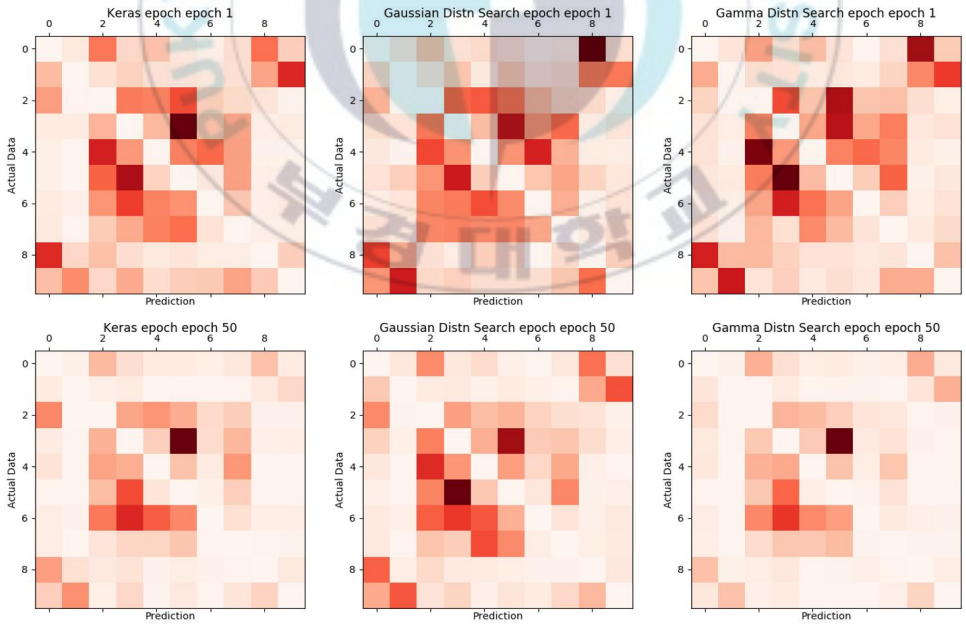

그림 21. CIFAR-10 데이터를 이용한 학습 결과의 비교 시각화

4.2.2 학습속도 평가

표 8은 10번의 최적화 횟수 동안 1 epoch에서 Grid 검색, Gaussian 검색 및 Gamma 검색 방법을 비교한 것이며, 정확도와 관련한 평가는 4장 2절 1항에서 평가를 진행하였다. 본 항에서는 학습 시간에 대한 테스트로써, 정확성의 비교보다는 학습 시간 비교에 중점을 두고자 한다. Grid의 값이 클수록 Grid 검색에서 정확도가 더 높아짐을 알 수 있지만, 이에 따라 학습 시간이 증가함을 알 수 있다. 이러한 의미는 Gaussian 및 Gamma에서 Iteration을 증가시키는 의미와 같다. 하지만 비슷한 정확도 대비 학습 시간의 차이는 크게 차이가 있음을 알 수 있다. 즉, 표 8의 평균을 비교해 보면 Gaussian 및 제안하는 Gamma 검색에 필요한 시간은 거의 변하지 않고, 비슷했지만 정확도는 7% 이상 개선되었다. 이는 이전 연구에서 제안한 것과 일치함을 알 수 있다.

표 8. 1 epoch에서의 기존 시스템과 제안 시스템의 비교

Grid No	기존				제안	
	Grid 검색		Gaussian 검색		Gamma 검색	
	*Acc(%)	**LT(Sec)	Acc(%)	LT(Sec)	Acc(%)	LT(Sec)
100	47	5.3	42	4.4	52	4.3
200	49	10.4	42	4.1	48	4.2
300	50	16.6	42	4.7	47	4.7
400	52	20.1	37	5.1	41	4.9
500	49	25.6	39	4.6	51	4.3
Avg	49.4	15.6	40.4	4.58	47.8	4.48

* Acc: Accuracy, ** LT: Learning Time

표 9는 10개의 최적화와 50개의 epoch를 적용하는 경우의 Grid 검색, Gaussian 검색 및 Gamma 검색을 비교한 것이다. 특히 Grid 수가 증가하

더라도 정확도는 크게 영향을 받지 않았다. Grid 크기의 평균을 100~500에서 비교하면 제안 검색과 Grid 검색 간에 0.7% 차이가 있었지만, Grid 검색 시간은 729.9초였으며 제안된 검색 시간은 29.38로 약 24.8배 빠름을 알 수 있다.

표 9. 50 epoch에서의 기존 시스템과 제안 시스템의 비교

Grid No	기존				제안	
	Grid 검색		Gaussian 검색		Gamma 검색	
	*Acc(%)	**LT(Sec)	Acc(%)	LT(Sec)	Acc(%)	LT(Sec)
100	87	226.2	87	27.0	90	27.1
200	90	443.5	89	27.2	88	26.9
300	91	679.2	87	26.8	89	27.0
400	89	881.3	86	25.9	89	26.1
500	87	1419.3	81	39.2	84	39.8
Avg	88.8	729.9	86	29.22	88	29.38

* Acc: Accuracy, ** LT: Learning Time

표 10. 20 Iteration 및 1 epoch에서의 기존 시스템과 제안 시스템의 비교

Grid No	기존				제안		
	Grid 검색		Gaussian 검색		Gamma 검색		
	Acc(%)	LT(Sec)	Acc(%)	LT(Sec)	Acc(%)	LT(Sec)	LR*
100	93	215.9	92	26.1	93	27.7	0.265
200	92	459.1	88	28.55	91	27.8	0.113
300	94	647.58	91	27	92	26.7	0.178
400	94	899.84	93	28.1	92	27.9	0.077
500	94	1124.46	93	26.58	94	26.4	0.230
Avg	93.4	699.38	91.4	27.266	92.4	27.3	

* LR: Learning Rate

또한 Gaussian 검색과 비교하여 제안된 방법의 정확도가 2% 증가한 것으로 확인 할 수 있다. 손실 값과 정확도 추정값을 비교할 때 개선을 확인하기 위해 20번의 최적화로 실험을 다시 수행하였으며, 그에 대한 결과는 표 10과 같다.

표 10은 20 Iteration에 1 epoch일 때 학습 속도를 비교한 것이다. 실제 20 Iteration을 설정함에 따라 정확도가 올라감을 알 수 있다. 전체적인 학습 시간도 증가하였음을 또한 발견할 수 있었다. 따라서 Iteration을 계속 증가하게 되면 Grid 검색보다 높은 정확도가 될 수 있고, 또한 학습 속도도 증가하겠지만, Grid 검색 보다 우위에 있음을 예측할 수 있다. 또한 실제 최적화 포인트가 Saddle Point인지 확인하는 것이 중요한데, Grid 검색을 통해 얻은 포인트의 경우 이를 Saddle Point의 근사치로 생각할 수 있다. 즉, 정확도가 높고 학습률이 Saddle Point와 유사하기 때문에 최적화 횟수를 늘리면 더 정확한 포인트를 찾을 수 있다고 해석할 수 있다.

마지막으로 표 10의 평균값을 비교하면 κ를 사용하여 이전 연구보다 학습 속도의 전반적인 향상을 확인할 수 있으며, Grid 검색과 비교할 때 학습 시간 차이는 정확도 차이보다 훨씬 높음을 알 수 있었다. 하지만 500 Graid No에서 제안하는 방법이 정확도는 비슷하지만 속도는 더 높았다.

표 11. 표 10의 결과에 대한 학습률 비교표

Grid No	기존		제안
	Grid 검색	Gaussian 검색	Gamma 검색
100	0.085	0.307	0.265
200	0.070	0.075	0.113
300	0.156	0.117	0.178
400	0.096	0.138	0.077
500	0.078	0.186	0.230

또한 제안된 방법의 결과는 Gaussian 검색보다 정확도가 높았으며, 표 11과 같이 학습률의 경우 제안된 방법이 Gaussian 검색보다 Grid 검색의 학습률에 가까운 값을 가지고 있음을 확인 할 수 있다. 즉, 제안된 검색 방법은 정확성과 비용 측면에서 기존 시스템보다 우수함을 나타낸다.

V. 결론 및 향후 연구

현재 기계 학습은 사용자 편의를 위한 방향으로 관심이 높아가고 있다. 그에 따라 AutoML과 같은 분야로 새로 정립하고 이를 실현하기 위한 연구에 대하여 활발히 논의 중에 있다.

최근 문제를 고려할 때 AutoML은 향후 머신 러닝에서 핵심적인 역할을 할 것으로 예상할 수 있다. 특히 이와 관련하여 데이터 전처리, 특징값의 추출, 모델 선택의 적절성, 매개변수 최적화로 분류하여 상세화하고 있다. 그중에서 Hyperparameter와 관련하여 학습 속도, Mini-Batch 크기 및 정규화 계수를 고려할 수 있으며, 이러한 변수는 항상 기계학습을 진행할 때 직접적으로 학습 성능에 영향을 준다.

하지만, 학습 네트워크 모델의 대부분은 학습에 사용된 데이터에 의존적인 Hyperparameter를 정의하고, 개발되고 있다. 따라서 데이터가 변경되게 되면 기존에 설계된 Hyperparameter를 다시 수정 및 보완하여야 한다.

이러한 문제는 실제 AutoML을 진행하는데 많은 어려운 요소로 작용할 수 있다. 따라서 본 논문에서는 기계학습에 사용되는 Hyperparamter에 대하여 자율적으로 검색하고 적절한 값을 반영할 수 있는 시스템을 제안하고자 하였다.

이러한 자율적 기능을 반영하기 위하여 Bayesian 최적화를 활용하였으며, 기존의 Surrogate Model을 참고하여 목적함수에 직접적인 접근을 통한 Sampling 전에 특정 함수 Acquisition Function을 사용하였다.

Acquisition Function은 Gaussian 분포를 이용하여 목적함수를 근사하는 역할을 한다. 본 연구에서는 이와 관련된 분포를 Gamma 분포를 적용하고

자 하였으며, 추가로 목적함수의 출력값과 관련하여, 손실 값과 정확도의 값을 활용하여 Convex 한 목적함수로 제공하는 기법에 대하여 설명하였고, 이러한 기법을 통하여, MNIST, CIFAR-10의 데이터를 대상으로 실험을 진행하였다.

결과에서 Gamma 분포를 사용한 개선된 목적함수의 경우가 학습 시간 및 정확도에서 향상할 수 있음을 확인하였으며, 통계적 기법을 통하여 성능향상에 대한 평가가 신뢰가 있음을 보였다.

추가 연구에서는 목적 함수에서 출력되는 값에 대한 연구 중 Convex 한 모델의 구성과 Sampling 과정에서 편차가 0에 가까워지는 문제점과 Bayesian 최적화의 시작 시점에 설정되는 범위 값에 대한 연구를 추가로 진행하고자 한다.

참고문헌

[1] AutoML, Machine Learning for Automated Algorithm Design(2018), http://www.l4aad.org/automl/.

[2] J. Bergstra, and Y. Bengio, “Random search for hyper-parameter optimization,” *Journal of Machine Learning Research,* Feb. 2012, pp. 281-305.

[3] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, “Algorithms for hyper-parameter optimization,” *Proc. Advances in Neural Information Processing Systems 24, Neural Information Processing Systems*(NIPS), Granada Congress and Exhibition Centre, Granada, Spain, Dec. 2011, pp. 2546-2554.

[4] B. Guo, J. Hu, W. Wu, Q. Peng, and F. Wu, “The Tabu Genetic Algorithm A Novel Method for Hyper-Parameter Optimization of Learning Algorithms,” Electronics, Vol. 8, No. 5, 2019, p. 579.

[5] B. Settles, “Active learning literature survey. Computer Science Technical Report,” University of Wisconsin-Madison, Jan. 2010.

[6] A. Kapoor, K. Grauman, R. Urtasun, and T. Sarrell. “Active learning with Gaussian processes for object categorization,” *Proc. 2007 IEEE 11th Int’l Conf. Computer Vision*(ICCV), Rio de Janeiro, Brazil, Oct. 2007. pp. 1-8.

[7] M. Osborne, R. Garnett, and S. Roberts, “Active data selection for sensor networks with faults and changepoints,” *Proc. 2010 IEEE 24th Int’l Conf. Advanced Information Networking and Applications*, Perth, WA, Australia,

Apr. 2010, pp. 533-540.

[8] V. Dalibard, M. Schaarschmidt, and E. Yoneki, "BOAT: Building auto-tuners with structured Bayesian optimization," *Proc. the 26th Int'l Conf. World Wide Web, International World Wide Web Conferences Steering Committee*, Perth, Australia, Apr. 2017, pp. 479-488.

[9] X. Lu, J. Gonzalez, Z. Dai, and N. Lawrence, "Structured Variationally Auto-encoded Optimization," *Proc. 35th Int'l Conf. Machine Learning, Proceedings of Machine Learning Research*(PMLR), Vol. 80, Jul. 2018, pp. 3267-3275.

[10] Z. Dai, A. Damianou, J. González, and N. Lawrence, N. "Variational auto-encoded deep Gaussian processes," *arXiv preprint*, Nov. 2015, arXiv:1511.06455.

[11] R. Calandra, J. Peters, C. E. Rasmussen, and M. P. Deisenroth, "Manifold Gaussian processes for regression," *Proc. the 2016 Int'l Joint Conf. Neural Networks*, Vancouver, BC, Canada, Jul. 2016, pp. 3338-3345.

[12] J. Bergstra, D. Yamins, and D. D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," *Proc. the 30th Int'l Conf. Machine Learning*, Atlanta, Georgia, USA, Jun. 2013, pp. 115-123.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Proc. Advances in Neural Information Processing Systems 25, Neural Information Processing Systems*(NIPS), Lake Tahoe, Nevada, USA, Dec. 2012, pp. 1097-1105.

[14] D. R. Jones, M. Schonlau, and W. J. Welch, “Efficient global optimization of expensive black-box functions,” *Journal of Global Optimization,* Vol. 13, No. 4, Dec. 1998, pp. 455-492.

[15] C. Hung, J. Underwood, J. Nieto and S. Sukkarieh, “A feature learning based approach for automated fruit yield estimation,” *In Field and service robotics*, Springer, 2018, pp. 485-498.

[16] Z. Zhang, T. Jiang, S. Li and Y. Yang, “Automated feature learning for nonlinear process monitoring - An approach using stacked denoising autoencoder and k-nearest neighbor rule,” *Journal of Process Control*, MDPI, 2018, Vol. 64, pp. 49-61.

[17] B. Zoph and Q. V. Le, “Neural architecture search with reinforcement learning,” *arXiv preprint*, 2016, arXiv:1611.01578.

[18] T. Saikia, Y. Marrakchi, A. Zela, F. Hutter and T. Brox, “AutoDispNet: Improving Disparity Estimation with AutoML,” *arXiv preprint*, 2019, arXiv:1905.07443.

[19] C. Wong, N. Houlsby, Y. Lu and A. Gesmundo, “Transfer learning with neural automl,” *Proc. Advances in Neural Information Processing Systems 32, Neural Information Processing Systems*(NIPS), Montreal, Canada, Dec. 2018, pp. 8356-8365.

[20] J. Snoek, H. Larochelle, and R. P. Adams, “Practical bayesian optimization of machine learning algorithms,” Proc. Advances in Neural Information Processing Systems 25, Neural Information Processing Systems(NIPS), Lake Tahoe, Nevada, USA, Dec. 2012, pp. 2951-2959.

[21] A. Törn, and A. Žilinskas, *Global Optimization*, Springer-Verlag: Berlin, Germany, 1989.

[22] M. Mongeau, H. Karsenty, V. Rouzé, and J. B. Hiriart-Urruty, "Comparison of public-domain software for black-box global optimization," Optim. Methods Software, Vol. 13, No. 3, 1998, pp. 203-226.

[23] L. Liberti, and N. Maculan, *Global Optimization: From Theory to Implementation*, Springer Optimization and Its Applications; Springer: Berlin, Germany, 2006.

[24] A. Zhigljavsky, and A. Žilinskas, *Stochastic Global Optimization*, Springer Optimization and Its Applications; Springer: Berlin, Germany, 2007.

[25] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2009.

[26] E. Brochu, V. M. Cora, and N. De Freitas, "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *arXiv preprint*, 2010, arXiv:1012.2599.

[27] J. Mockus, "Application of Bayesian approach to numerical methods of global and stochastic optimization," *Journal of Global Optimization*, Vol. 4, No. 4, 1994, pp. 347-365.

[28] P. J. Frazier, "A tutorial on bayesian optimization," *arXiv preprint*, 2018, arXiv:1807.02811.

[29] R. Martinez-Cantin, N. de Freitas, E. Brochu, J. Castellanos, and A. Doucet, "A Bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot," *Autonomous Robots*,

Vol. 27, No. 2, 2009, pp. 93-103.

[30] E. Brochu, T. Brochu, and N. de Freitas. "A Bayesian interactive optimization approach to procedural animation design," *Proc. 2010 ACM ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Jul. 2010, pp. 103-112.

[31] M. M. Khajah, B. D. Roads, R. V. Lindsey, Y. E. Liu, and M. C. Mozer, "Designing engaging games using bayesian optimization," *Proc. ACM the 2016 chi Conf. human factors in computing systems*, 2016, pp. 5571-5582.

[32] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth, "Bayesian optimization for learning gaits under uncertainty," *Annals of Mathematics and Artificial Intelligence*, Vol. 76, No. 1-2, 2016, pp. 5-23.

[33] H. J. Kushner, A new method of locating the maximum of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, Vol. 86, No. 1, 1964, pp. 97-106.

[34] A. Torn and A. Zilinskas. *Global Optimization,* Springer-Verlag, 1989.

[35] D. R. Jones, "A taxonomy of global optimization methods based on response surfaces," *Journal of Global Optimization*, Vol. 21, No. 4, 2001, pp. 345-383.

[36] D. J. Lizotte, "Practical Bayesian Optimization," Doctoral Dissertation, Univ. Alberta, Edmonton, Alberta, Canada, 2008.

[37] J. Mockus, V. Tiesis, and A. Zilinskas, *Toward Global Optimization, volume 2, chapter The Application of Bayesian Methods for Seeking the Extremum*, Elsevier, 1978, pp. 117-128.

[38] Y. Kim, and M. Chung, “An Approach to Hyperparameter Optimization for the Objective Function in Machine Learning,” *Joural of Electronics*, Vol. 8, No. 11, 1267, Nov. 2019.

[39] M. H. DeGroot, and M. J. Schervish, *Probability and Statistics*,” Pearson Education Limited:London, UK, 2014, pp. 302 - 325.

[40] Y. Kim, and M. Chung, “An Approch of Hyperparameter Optimization using Gamma Distribution,” *Proc. the 15th Int’l Conf. Multimedia Information Technology and Applications*, University of Economics and Law, Ho chi minh, Vietnam, Vol. 1, Jul. 2019, pp. 75 - 78.

[41] N. Q. K. Le, T. T. Huynh, E. K. Y. Yapp, and H. Y. Yeh, “Identification of clathrin proteins by incorporating hyperparameter optimization in deep learning and PSSM profiles,” *Journal of Compute Methods and Programs in Biomedicine*, Vol. 177, Aug. 2019, pp. 81 - 88.

감사의 글

석사과정을 마치고 박사과정을 시작할 수 있는 용기를 낼 수 있게 많은 분의 도움을 받았습니다. 이 자리를 빌려 감사의 말씀을 드립니다.

연구 설계와 결과를 만들기 위해 노력하였던 지난날의 깊었던 밤이 문득 떠오릅니다. 지도교수님께서 말씀하신 “이제 연구 하려하니 끝이 났다.”는 말씀을 이제 이해할 것 같아 아쉬운 마음이 끝이 없지만, 이제 시작이라는 마음으로 최선을 다하고자 합니다. 그리고 박사과정 중에 수많은 어려움이 있었고, 이러한 부분이 있을 때 현명하게 지도하여 주신 정목동 지도 교수님께 감사의 말씀을 전하고자 합니다. 석사과정부터 현재까지 변함없이 지켜봐 주신 교수님께 어떻게 감사의 말씀을 드려야 할지 어려울 정도입니다. 교수님, 감사합니다.

학기 중에 많은 시간과 연구를 허락하신 주식회사 썬컴의 고태훈 대표님과 직원분들께도 깊은 감사를 드립니다. Lab. 식구들에게도 감사를 표하고 싶습니다. 바쁘다는 핑계로 도와주지 못한 동분서주하던 현기와 항상 도움 주신 김태우 샘께 감사의 말을 전합니다. 자주 만나지는 못하지만, 항상 격려와 웃음으로 용기를 주었던 친구들 채균, 홍주, 상우, 현재, 윤철, 영철에게도 감사의 마음을 전합니다.

마지막으로 모든 대소사에 적극적으로 참여하지 못한 저를 이해해 주시고, 묵묵히 응원하여 주시고, 항상 걱정해 주셨던 어머니, 형, 누나, 자형, 형수께도 무한한 감사들 드리고자 합니다.

“Learn as if you would live forever, live as if you would die tomorrow”의 Mahatma Gandhi 말처럼 현명하게 노력하고 최선을 다하겠습니다.

2020년 1월 14일

김용훈 드림.