랜덤 공간 분할 최적화기를 이용한 기계학습 자동화 시스템

김정택[a] and 정종헌[b] and 최승진[a]

[a] 포항공과대학교 컴퓨터공학과
37673, 경상북도 포항시 남구 청암로 77
Tel: +82-54-279-2924, Fax: +82-54-279-2299, E-mail: {jtkim, seungjin}@postech.ac.kr

[b] 엑스브레인
37673, 경상북도 포항시 남구 청암로 77
Tel: +82-70-8820-7277, E-mail: jongheonj@exbrain.io

Abstract

이 논문은 미리 정의된 기계학습 알고리즘 구성 공간에서, 랜덤 공간 분할 최적화기를 이용한 베이지안 최적화를 통해 최적의 알고리즘 구성을 찾는 방법과 이를 적용한 시스템을 제시한다. 빅데이터 분석을 위한 기계학습 적용을 수월하게 하기 위해선, 단순한 기계학습 알고리즘의 활용을 넘어서 전문가의 세밀한 조정이 필요하다. 하지만 이는 경험적인 방법이며 전문가가 어느 정도 성능을 보장해주지만, 모델이 최적의 성능에 도달하였다고 말하기 힘들다. 따라서 최근 최적의 알고리즘 구성을 찾기 위한 자동 시스템이 제시되고 있으며, 이를 기계학습 자동화 시스템이라 부른다. 이 논문에서는 사용자의 간섭 없이 순차적인 베이지안 최적화를 이용하여 최적의 알고리즘 구성을 찾는다. 특히 일반적인 시스템에서는 가우시안 프로세스 회귀법이 사용되는 것에 반해서, 이 논문에서는 랜덤 공간 분할 방법 중 하나인 몬드리안 포레스트 회귀법을 모든 형태의 변수로 확장시키고 병렬화가 가능하게 만든 몬드리안 포레스트 최적화기를 이용했다. 이 최적화기는 다른 최적화기와 비교해서 실제 성능측정값을 알지 못하는 상황에서 회귀를 했음에도 불구하고 거의 비슷한 성능을 보였다. 또한 이 논문에서 제시한 기계학습 자동화 시스템은 AutoML Challenge의 Final3와 Final4에서 4등, AutoML5에서 3등을 차지했다.

Keywords:

기계학습; 베이지안 최적화; 몬드리안 포레스트; 랜덤 공간 분할 최적화기; 기계학습 자동화

개요

최근 다양한 분야에서 다양한 형태로 방대한 데이터가 생성되고 있으며, 이들은 제각기 다른 특성을 가진다. 이러한 데이터들은 흔히 빅데이터 (big data)로 일컬어 진다. 흔히 빅데이터는 양의 초점을 맞춰서 접근되는 경우가 많지만, 이의 특성에서 각기 다른 사전지식이 필요한 다양한 분야로부터 도래한 데이터라는 점과 관성을 가지고 실시간으로 변화하는 데이터라는 사실 또한 중요하다. 이러한 특성에 비추어 봤을 때 빅데이터를 대상으로 하는 기계학습 알고리즘의 적용은 양에 대한 강건함뿐만 아니라 다양한 형태를 가지고 실시간으로 변화하는 데이터에 대한 고려가 필요하다. 하지만 이를 위해선 전문가라고 불리는 사람이 직접 데이터를 확인하여 기계학습 모델의 설정을 조정하는 작업이 필요하다. 기존의 기계학습을 통해 분류 (classification)나 군집 (clustering) 등의 문제를 해결하는 방법은 이러한 기작을 따랐으며, 데이터 분석가, 기계학습 전문가 등이 앞으로 많이 필요하다라고 이야기되는 이유였다. 하지만 이는 경험적인 방법 (heuristic method)일 뿐이며, 전문가가 성능을 가능한 범위에서 보장해주기는 하지만 그 성능이 최적의 성능이라 말하기는 힘들다. 따라서 최적의 성능을 효율적으로, 그리고 지능적으로 찾는 연구가 진행되어야 한다. 이러한 연구는 기계학습 자동화 시스템 (automated machine learning system)이라 불리며, 다양한 연구 그룹에 의해 연구 [1, 2, 3, 4]되었고 2014년부터 AutoML Challenge라는 이름의 대회가 진행 [5]되기도 했다.

우리는 기계학습 알고리즘이 작동하는 설정을 알고리즘 구성 (algorithm configuration)이라고 부른다. 또한 모든 알고리즘 구성이 존재하는 공간을 알고리즘 구성 공간 (algorithm configuration space)이라 부른다. 이는 다음과 같은 정의 1로 정의될 수 있다:

정의 1 (알고리즘 구성 공간)

$$\Theta \times \Lambda \times A.$$

Θ는 모델 매개변수의 전체 셋이며, Λ는 하이퍼파라미터의 전체 셋, A는 알고리즘의 전체 셋이다. 모델 매개변수와 하이퍼파라미터의 전체 셋은 각 알고리

즘이 가지고 있는 변수들의 전체 셋을 의미한다. 결과적으로 알고리즘 구성 공간은 세 차원의 외적 공간이다. 최적의 성능을 보이는 모델은 정의 1의 알고리즘 구성 공간에서 찾아지게 되며, 이 공간을 지능적으로 찾는 방법이 이 논문에서 주목하는 문제이다. 이 문제를 풀기 위해서 확률에 기반한 베이지안 최적화 (Bayesian optimization)가 적용되며, 지금까지 이를 위한 다양한 방법들이 제시 [4, 9]되었다. 이 논문에서는 결과적으로 새로운 최적의 알고리즘 구성을 찾는 방법인 랜덤 공간 분할 최적화기 (random space partitioning optimizer)를 이용한 기계학습 자동화 시스템을 제안한다. 이를 수식으로 정의하면 정의 2와 같다:

정의 2 (기계학습 자동화 시스템)

$$\underset{\theta_i\in\Theta,\lambda_i\in\Lambda,A_i\in\mathbf{A}}{\operatorname{argmin}} \mathcal{L}(f(\theta_i,\lambda_i,A_i;\{(\boldsymbol{x}_k,y_k)\}_{k=1}^n)).$$

정의 2에서 $\mathcal{L}$은 손실 함수, f는 예측 모델, θ_i와 λ_i는 선택된 알고리즘 A_i의 매개변수와 하이퍼파라미터 (hyperparameter)의 집합, $\boldsymbol{x}_k$과 y_k는 이미 알고 있는 알고리즘 구성과 그의 성능측정값 (performance measure)이다. 최종적으로 정의 1과 같이 기계학습 알고리즘을 작동할 수 있도록 하는 알고리즘 구성들의 공간을 정의하고, 이 알고리즘 구성을 매개변수화하여 정의 2를 풀게 된다.

기계학습 자동화 시스템 문제에서 주목하고 있는 주요 이슈는 1. 연속 변수 (continuous variable)와 분류 변수 (categorical variable) 등을 모두 포함하는 최적화의 결과가 어떠한가 2. 알고리즘 성능측정값을 빠르게 예측 가능한가 3. 성능측정값의 불확실성 (uncertainty)이 얼마나 정확한가 4. 알고리즘의 병렬화 (parallelization)가 얼마나 가능한가 등이 있다. 이러한 이슈를 해결하기 위한 대표적인 방법으로는 현재 최고의 성능을 보인다고 알려져 있는 시스템인 auto-sklearn [4]이 있다. 이 시스템은 랜덤 포레스트 (random forests) [12]를 이용한 베이지안 최적화 방법인 순차적 모델 기반의 알고리즘 구성 (sequential model-based algorithm configuration, SMAC) [9]을 이용하는데, 위의 네 가지 이슈에 대해 다른 방법들과 비교했을 때 강점을 가진다.

이 논문에서 제시하는 시스템은 auto-sklearn [4]에 기반하여 몬드리안 포레스트 최적화기 (Mondrian forests optimizer, MFO)를 통해 SMAC이 가지는 이점을 더 강화한다. MFO는 몬드리안 포레스트 회귀 (Mondrian forests regression) [7]를 숫자 변수 (numerical variable)와 분류 변수로 확장한 랜덤 공간 분할 최적화기이다. 이 최적화기는 실제 성능측정값을 알지 못하는 상태에서 측정값을 예측할 수 있으므로 병렬화 또한 가능하다. 이 최적화기는 다양한 전역 최적화 벤치마크에서 기존의 최적화기와 비교해서 거의 유사하거나 좀더 나은 성능을 보였다. 또한 전체 기계학습 자동화 시스템인 *postech.mlg_exbrain*은 AutoML Challenge의 Final3와 Final4에서 4등, AutoML5에서 3등을 차지했다.

관련 연구

기계학습 자동화

기계학습의 적용은 크게 네 과정으로 정리될 수 있다. 이는 특징 변환 (feature transformation), 모델 매개변수 추정 (model parameter estimation), 하이퍼파라미터 최적화 (hyperparameter optimization), 알고리즘 선택 (algorithm selection)이다. 이 중에 특징 변환은 선처리로 행해지는 경우가 많다. 따라서 나머지 세 과정이 순차적으로 진행된다. 일반적으로 기계학습은 알고리즘 선택과 하이퍼파라미터 최적화는 사용자가 수행하고 모델 매개변수 추정 문제만을 푼다. 이를 식으로 표현하면 다음과 같다:

$$\underset{\theta_i\in\Theta}{\operatorname{argmin}} \mathcal{L}(f(\theta_i;\ \lambda_i,A_i,\{(\boldsymbol{x}_k,y_k)\}_{k=1}^n)).$$

또한 모델 매개변수 추정과 함께 하이퍼파라미터 또한 매개변수화하여 최적화할 수 있다. 이는 다음과 같은 식으로 표현된다:

$$\underset{\theta_i\in\Theta,\lambda_i\in\Lambda}{\operatorname{argmin}} \mathcal{L}(f(\theta_i,\lambda_i;\ A_i,\{(\boldsymbol{x}_k,y_k)\}_{k=1}^n)).$$

모델 매개변수 최적화와 하이퍼파라미터 최적화를 포함하여 알고리즘 선택도 함께 매개변수화하여 최적화하면 정의 2와 같이 정의될 수 있다. 이 논문에서 다루는 문제는 정의 2이다.

순차적 모델 기반 베이지안 최적화

알고리즘 구성과 성능측정값의 함수와 같이 정확한 해가 존재하지 않고 주어진 점에서만 함수값을 알 수 있는 함수를 대상으로 최적화 문제를 풀 경우, 순차적 모델 기반 베이지안 최적화 (sequential model-based Bayesian optimization) [8, 9]를 적용한다. 실제 데이터가 순차적으로 입력되고 이를 기반으로 회귀법을 통해 예측 함수를 생성한다. 그리고 이 예측 함수의 공역에 대해 평가를 하여, 가장 전역 최적값 (global optimum)이 존재할 가능성이 큰 점에서 실제 값을 얻는다. 예측된 함수를 평가하는 함수를 습득 함수 (acquisition function)라 부른다. 이를 의사코드로 정리하면 다음과 같다.

알고리즘 1 (순차적 모델 기반 베이지안 최적화)
필요: 초기 데이터셋 $\{(\boldsymbol{x}_k,y_k)\}_{k=1}^I$
1: for $t = 1, 2, \cdots$ do
2: 목적함수라 고려되는 $f^*(\boldsymbol{x}|\{(\boldsymbol{x}_k,y_k)\}_{k=1}^{I+t-1})$를 예측.

3: 습득 함수를 최적화하여 다음 점인 $\mathbf{x}_{I+t}$를 얻음, $\mathbf{x}_{I+t} = \operatorname{argmax}_{\boldsymbol{x}} a(\boldsymbol{x})$.
4: 실제 목적함수에서 함수값을 얻음, $y_{I+t} = f(\boldsymbol{x}_{I+t}) + \epsilon_{I+t}$.
5: 전체 데이터셋을 업데이트.
6: endfor

일반적으로 예측 함수를 생성하기 위한 회귀법으로 베이지안 비모수 방법 (Bayesian nonparametric method) 중 하나인 가우시안 프로세스 회귀 (Gaussian process regression) [10]가 사용된다. 많은 경우에서 가우시안 프로세스 회귀는 정확한 결과를 만들어내지만, 알고리즘 구성을 다루는 문제에서는 숫자 변수뿐만 아니라 분류 변수도 존재하므로 랜덤 포레스트 기반의 회귀법을 사용하는 방법이 제안되기도 했다. 그러나 더 자세히 순차적 모델 기반 베이지안 최적화를 살펴보면, 데이터가 순차적으로 들어오므로 매번 회귀 모델을 생성하는 것이 비효율적이라는 사실을 알 수 있다. 결과적으로 이 문제는 온라인 순차적 문제라 생각될 수 있으며, 이 문제를 해결하기 위해 다음에 설명할 몬드리안 포레스트 기반의 회귀법을 도입하게 된다.

습득 함수는 회귀 모델의 출력값인 함수값과 함수값의 분산 (variance)을 바탕으로 전역 최적값이 존재할 가능성이 가장 높은 점을 평가한다. 전통적으로 세 종류의 습득 함수, 확률 개선 (probability improvement), 기대 개선 (expected improvement) [15], 가우시안 프로세스 상부 신뢰 경계 (Gaussian process upper bound confidence) [16]가 사용된다.

알고리즘 구성를 정의역으로 하는 순차적 모델 기반 베이지안 최적화 문제를 풀기 위해 대표적으로 가우시안 프로세스 기반의 Spearmint [11]와 랜덤 포레스트 기반의 SMAC [9]이 제시되었다.

몬드리안 포레스트

Lakshminarayanan 외 2인 [6, 7]은 확률적인 일반화된 k-d 트리 (k-d tree)인 몬드리안 트리 (Mondrian tree)의 앙상블 (ensemble)인 몬드리안 포레스트 (Mondrian forests)를 제안한다. 몬드리안 포레스트는 분류나 회귀에 사용되는 결정 트리(decision tree) 중 하나이다. 또한 몬드리안 트리는 몬드리안 프로세스 (Mondrian forests) [13]를 유한한 점으로 구성하여 만든 제한적인 구조이다. 몬드리안 트리의 중요한 특징은 함수값 정보 없이 주어진 공간에 대해서 랜덤 분할, 즉 트리를 학습한다는 사실이다. 실제 함수값은 주어진 공간에 대한 함수값을 예측할 때 사용한다. 결과적으로 m번째 몬드리안 트리, T_m의 예측 함수값 분포 (predictive label distribution)는 다음과 같이 표현된다:

$$p_{T_m}(y|\boldsymbol{x}_{test}, \{(\boldsymbol{x}_k, y_k)\}_{k=1}^{n}) = \sum_{j \in path(leaf(\boldsymbol{x}_{test}))} w_{mj} \mathcal{N}\left(y|\mu_{mj}, {\sigma_{mj}}^2\right).$$

총 M개의 트리가 주어지며 μ_{mj}과 ${\sigma_{mj}}^2$는 노드 j의 경계 분포 (marginal distribution)가 가지는 평균과 분산이다. w_{mj}는 노드 j에서 가지가 나눠질 확률로 표현되는 가중치이다. 그리고 $leaf$는 입력된 점을 가지는 잎 노드 (leaf node)를 의미하며, $path$는 입력된 잎 노드까지 도달하는 경로를 의미한다. 마지막으로 주어진 점에 대한 함수값과 그의 불확실성은 다음과 같이 표현된다:

$$\mu_{T_m} = \sum_j w_{mj} \mu_{mj}$$

$${\sigma_{T_m}}^2 = \sum_j w_{mj}(\sigma_{mj} + \mu_{mj}^2) - \mu_{T_m}^2.$$

함수값을 예측하기 위해 신뢰 전파 (belief propagation)를 이용해 각 노드의 경계 분포를 구해야 한다. 하지만 이 논문에서 다루는 문제가 온라인 상태인 특성상 이를 계산할 수 없다. 따라서 노드가 지니고 있는 학습 데이터의 샘플 분포 (sample distribution)를 경계 분포로 대체해서 계산한다.

시스템 명세

기반 시스템, auto-sklearn과 시스템 특성

auto-sklearn [4]은 최고의 성능을 가지는 기계학습 최적화 시스템 중 하나이다. 이는 scikit-learn 라이브러리 [14]를 사용하며, 네 개의 구성요소, 메타학습 초기화기 (meta-learning initializer), 베이지안 최적화기 (Bayesian optimizer), 기계학습 프레임워크 (machine learning framework), 앙상블 제조기 (ensemble builder)를 가진다. 위에서 언급한 순차적 모델 기반의 베이지안 최적화에 기반하여 순차적으로 최고의 알고리즘 구성을 찾는다. 첫 번째로 메타학습 최적화기는 다양한 데이터셋으로 초기 시작점을 결정한다. 베이지안 최적화기는 다음으로 실제 성능측정값을 얻을 알고리즘 구성을 결정하며, 기계학습 프레임워크는 실제 성능측정값을 계산한다. 마지막으로 앙상블 제조기는 순차적으로 최적화를 하면서 얻은 기계학습 모델 중 최적의 모델들로 앙상블을 구성한다.

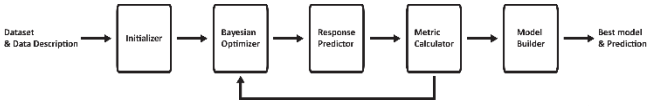

그림 1 – 제안한 시스템의 구성요소. 시스템은 총 다섯 개의 구성요소를 가짐.

이 논문에서 제안하는 시스템은 다섯 개의 구성요소

를 가진다. 각 구성요소는 초기화기 (initializer), 베이지안 최적화기 (Bayesian optimizer), 성능측정값 예측기 (response predictor), 측정기준 계산기 (metric calculator), 모델 제조기 (model builder)이다. 이 시스템의 가장 큰 강점은 가장 시간을 소비하는 과정인 최고의 성능을 보일 것이라고 예상되는 후보 알고리즘 구성을 습득하는 부분이다. 시간 제한과 하드웨어 성능 제한과 같은 한정된 자원 하에서 현명하게 알고리즘 구성을 얻기 위해 이 논문에서 개발한 MFO가 사용된다. MFO는 온라인 알고리즘일 뿐만 아니라 병렬화 가능한 알고리즘이므로 함수를 예측하고 습득할 알고리즘 구성을 결정하는 시간이 빠르다. 또한 랜덤 포레스트의 특성상 SMAC은 함수값의 불확실성을 알 수 없으므로, 경험적인 방법으로 이를 결정한다. 각 트리에서 얻은 결과의 불확실성을 함수값의 불확실성으로 여겨서 구한다. 하지만 이 값은 적은 수의 반복임에도 불구하고 0으로 빠르게 수렴한다. 아직 값을 확인하지 않은 공간에서 알고리즘 구성을 습득하기 위해서는 불확실성이 계속 존재해야 하는데, 불확실성 값이 0으로 수렴하므로 베이지안 최적화가 제대로 이뤄지지 않는다. 반면에 MFO는 각 트리 안에서 노드들의 경계 분포를 이용하여 불확실성을 계산하므로 이러한 문제를 가지지 않는다.

몬드리안 포레스트 최적화기

위에서 언급한대로 베이지안 최적화기가 시스템의 한 구성요소로 포함된다. 이 논문에서 제안하는 시스템은 몬드리안 포레스트 최적화기 (MFO)를 베이지안 최적화기로 사용한다. MFO는 몬드리안 포레스트를 기계학습 자동화에 맞추어 확장한 랜덤 공간 분할 최적화기이다. 실제 성능측정값을 알기 어렵고 순차적으로 회귀 모델이 업데이트되므로, 몬드리안 포레스트의 고유한 특성을 이용할 수 있으며 이 문제를 풀기 위해 이를 적용했다. MFO는 알고리즘 2와 같이 정리된다.

알고리즘 2 (몬드리안 포레스트 최적화기)
입력: $\mathcal{D} = \{(\boldsymbol{x}_k, y_k)\}_{k=1}^{N}$. $\boldsymbol{x}_k$는 알고리즘 구성, y_k는 실제 측정된 성능측정값. $\mathcal{T}$는 시간 제한.
출력: 알고리즘 구성, $\boldsymbol{x}_{best}$

```
MF = None
for t < 𝒯 do
    if MF == None then
        주어진 𝒟로 몬드리안 포레스트, MF를 구성.
    else
        새로 추가된 데이터로 MF를 확장.
    endif
    지역 검색을 위한 시드 알고리즘 구성을 임의로 뽑음.
    시드 알고리즘 구성 주변에서 습득 함수의 결과가 가장 큰 알고리즘 구성을 습득.
    정해진 개수의 임의로 뽑힌 알고리즘 구성과 아홉 번째 줄에서 습득한 알고리즘 구성을 합침.
    열 번째 줄의 결과 중 정해진 개수의 최고 결과를 𝒟로 업데이트.
endfor
return 𝒟에서 성능측정값이 최대인 x_best.
```

알고리즘 2를 보면 알 수 있듯이, 기본적으로 베이지안 최적화의 순서를 유지하면서 기계학습 자동화 문제를 해결하기 위한 접근으로 확장했다. 세 번째 줄부터 일곱 번째 줄에서 알 수 있듯이 몬드리안 포레스트의 특성에 맞게 온라인으로 작동한다. 또한 문제 특성상 정의역 공간이 매우 넓으므로 지역 검색을 통해 회귀를 적용하고 습득 함수로부터 전역 최적값일 가능성이 높은 알고리즘 구성을 결정한다. 그리고 전체 공간을 검색하지 못하므로 임의로 정해진 개수만큼의 알고리즘 구성을 뽑는다. 이러한 과정은 최적화기의 탐색의 특성을 강화한다. 이는 여덟 번째 줄부터 열 번째 줄까지로 구현된다.
MFO는 매개변수화 된 알고리즘 구성 공간에서 알고리즘 구성을 비교하기 위해서 숫자 변수와 분류 변수를 모두 분할할 수 있게 몬드리안 포레스트를 확장했다. 분류 변수를 분할하는 방법은 두 가지가 있다. 첫 번째는 일대 전체 분할 (one-vs-rest partitioning)이며, 두 번째는 원핫 벡터 (one hot vector)를 이용한다. 일대 전체 분할은 결정 트리의 특성을 이용한 확장으로, 한 트리 안에서 쉽게 알고리즘 구성을 비교할 수 있게 한다. 원핫 벡터를 이용한 확장은 몬드리안 프로세스를 기반으로 확장한 결과이다. 이 논문에서는 구현의 수월성을 위하여 일대 전체 분할을 이용한다.
또한 MFO를 이용하면 순차적 모델 기반 베이지안 최적화를 병렬화할 수 있다.

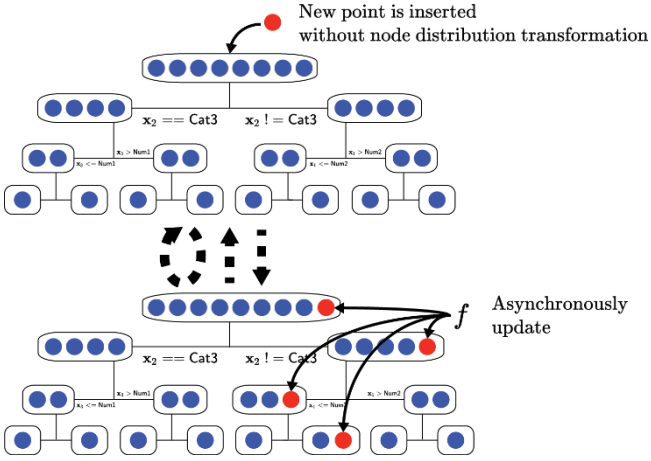

그림 2 – 새로운 알고리즘 구성을 노드의 분포에 변화없이 삽입하는 과정. 새롭게 삽입된 알고리즘 구성은 성능측정값 계산 후에 비동기적 업데이트.

그림 2와 같이 새로운 알고리즘 구성을 습득한 뒤에 실제 성능측정값을 계산하지 않고 노드의 분포에 변화없이 트리에 삽입한다. 그리고 병렬로 계산한 성능측정값을 추후에 비동기적으로 업데이트한다. 이러한 병렬화가 가능한 이유는 몬드리안 트리의 특성상 트리를 만들 때는 실제 함수값을 알 필요가 없기 때문이다.

이외의 새로운 접근법

이 논문에서 제시한 시스템을 최적화할 때, 데이터셋을 보충 (replacement)하면서 학습 데이터를 결정한다. 따라서 과적합 (overfitting)이 발생하지 않을 것이라 가정했다. 따라서 앙상블로 모델을 구성하는 대신, 최고의 성능을 보이는 하나의 모델로 최종 결과를 출력했다. 또한 이 문제에 대해서는 습득 함수로 자주 사용되는 기대 개선 함수 보다 가우시안 프로세스 상부 신뢰 경계 함수가 더 적합할 것으로 예상되어 이를 사용했다. 결정 트리의 특성상 회귀의 결과가 현재 알고 있는 최고 성능예측값에 제한될 수 있기 때문에 이 습득 함수를 적용해 시스템을 구현했다.

벤치마크 및 실제 문제 실험과 Challenge 결과

전역 최적화 벤치마크 및 실제 문제 실험

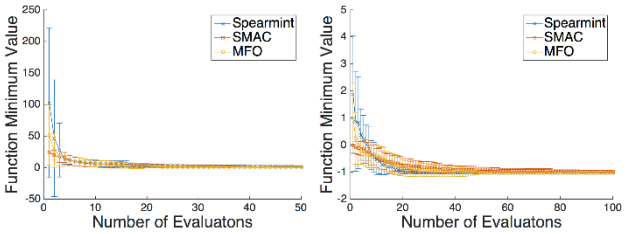

그림 3 – Spearmint, SMAC, MFO으로 최적화한 Branin 함수 (왼쪽)와 Camelback 함수 (오른쪽)의 평균과 표준편차. Spearmint는 10번 반복, SMAC과 MFO는 20번 반복.

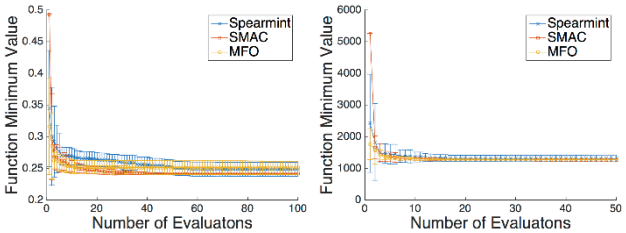

그림 4 – Spearmint, SMAC, MFO으로 최적화한 SVM (왼쪽)과 LDA (오른쪽)의 평균과 표준편차. Spearmint는 10번 반복, SMAC과 MFO는 20번 반복.

최적화기의 성능을 비교하기 위해, 전역 최적화 벤치마크 함수인, Branin 함수와 Camelback 함수를 비교했다. 또한 실제 문제인 SVM과 LDA의 하이퍼파라미터를 최적화하는 실험을 진행했다. 모든 실험의 Spearmint는 10번 반복하였으며, SMAC과 MFO는 20번 반복했다. 그리고 모든 초기 시작점은 임의로 설정했다.

AutoML Challenge 결과

표 1 – Final3, Final4, AutoML5 결과. 모든 순위는 https://competitions.codalab.org/competitions/2321에서 참조.

Final3		
팀명	순위	평균 순위
aad_freiburg	1	1.80
djajetic	2	2.00
ideal.intel.analytics	3	3.80
asml.intel.com	3	3.80
postech.mlg_exbrain	**4**	**5.40**
Final4		
팀명	순위	평균 순위
aad_freiburg	1	1.60
ideal.intel.analytics	2	3.60
Abhishek4	3	5.40
postech.mlg_exbrain	**4**	**5.80**
AutoML5		
팀명	순위	평균 순위
aad_freiburg	1	1.60
djajetic	2	2.60
postech.mlg_exbrain	**3**	**4.60**

AutoML Challenge [5]는 2014년 12월에 시작했다. Round 0를 제외하고 총 5 라운드로 구성되었다. 그리고 round 0과 round 5를 제외하고 각 라운드는 세 단계 (AutoML, Tweakathon, Final)를 가진다. 5 라운드 동안 이진 분류, 다중 클래스 분류, 다중 레이블 분류, 회귀 문제가 주어졌다. 우리의 시스템인 *postech.mlg_exbrain*은 Final3과 Final4에서 4등을, AutoML5에서 3등을 차지했다. 표 1의 평균 순위는 5개의 데이터셋에 대한 개별적인 순위의 평균이다. 이 평균 순위를 기준으로 최종 순위가 결정된다.

결론

이 논문에서는 랜덤 공간 분할 최적화기를 사용한 새로운 기계학습 자동화 시스템을 제안했다. 랜덤 공간 분할 최적화기는 MFO로 구현된다. 기계학습 자동화 문제는 온라인, 순차적인 환경이므로 MFO가 적합하게 사용될 수 있다. 이 논문에서 제안한 MFO는 전역 최적화 벤치마크와 실제 문제에서 기존 최적화기와 비교했을 때 비교할 만하거나 좋은 성능을 보였다. 그리고 최종적으로 이 전체 시스템은

AutoML Challenge의 Final3와 Final4에서 4등을, AutoML5에서 3등을 차지했다.

참고문헌

[1] A. Fukunaga (2002). “Automated discovery of composite SAT variable-selection heuristics”. *Proceedings of the AAAI National Conference on Artificial Intelligence (AAAI)*, pages 641-648.

[2] C. Ansotegui, M. Sellmann, and K. Tierney (2009). “A gender-based genetic algorithm for the automatic configuration of algorithms”. *Principles and Practice of Constraint Programming-CP 2009*, pages 142-157. Springer.

[3] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown (2013). “Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms”. *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 847-855.

[4] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter (2015). “Efficient and robust automated machine learning”. *Advances in Neural Information Processing Systems (NIPS)*, volume 28.

[5] I. Guyon, K. Bennett, G. Cawley, H. J. Escalante, S. Escalera, T. K. Ho, N. Macia, B. Ray, M. Saeed, A. Statnikov, et al. (2015). “Design of the 2015 ChaLearn AutoML challenge”. *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 1-8.

[6] B. Lakshminarayanan, D. M. Roy, and Y. W. Teh (2014). “Mondrian forests: Efficient online random forests”. *Advances in Neural Information Processing Systems (NIPS)*, volume 27.

[7] B. Lakshminarayanan, D. M. Roy, and Y. W. Teh (2016). “Mondrian forests for large-scale regression when uncertainty matters”. *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*.

[8] D. R. Jones, M. Schonlau, and W. J. Welch (1998). “Efficient global optimization of expensive black-box functions”. *Journal of Global Optimization*, 13(4):455-492.

[9] F. Hutter, H. H. Hoos, and K. Leyton-Brown (2010). “Sequential model-based optimization for general algorithm configuration (extended version)”. *Technical Report 10-TR-SMAC*, UBC.

[10] C. E. Rasmussen and C. K. I. Williams (2006). *Gaussian Processes for Machine Learning*. MIT Press.

[11] J. Snoek, H. Larochelle, and R. P. Adams (2012). “Practical Bayesian optimization of machine learning algorithms”. *Advances in Neural Information Processing Systems (NIPS)*, volume 25.

[12] L. Breiman (2001). “Random forests”. *Machine learning*, 45(1):5-32.

[13] D. M. Roy and Y. W. Teh (2008). “The Mondrian process”. *Advances in Neural Information Processing Systems (NIPS)*, volume 21.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. (2001). “Scikit-learn: Machine learning in python”. *Journal of Machine Learning Research*, 12:2825-2830.

[15] J. Mockus, V. Tiesis, and A. Zilinskas (1978). “The application of Bayesian methods for seeking the extremum”. *Towards Global Optimization*, 2:117-129.

[16] N. Srinivas, A. Krause, S. Kakade, and M. Seeger (2010). “Gaussian process optimization in the bandit setting: No regret and experimental design”. *Proceedings of the International Conference on Machine Learning (ICML)*.