공학석사 학위논문

유전 알고리즘 탐색을 이용한 랜덤 탐색 대비 효율적인 기계학습 시스템 구성을 위한 연구

A study for constructing an efficient machine learning system compared to random search using genetic algorithm search

울 산 대 학 교 대 학 원

전기전자컴퓨터공학과
컴퓨터공학전공

이 경 태

유전 알고리즘 탐색을 이용한 랜덤 탐색 대비 효율적인 기계학습 시스템 구성을 위한 연구

지 도 교 수 권 영 근

2020년 12월

울 산 대 학 교 대 학 원

전기전자컴퓨터공학과
컴퓨터공학전공

이 경 태

이 경 태의 공학석사 학위논문을 인준함

심사위원 김 종 면 (인)

심사위원 윤 석 훈 (인)

심사위원 권 영 근 (인)

울산대학교 대학원

2020년 12월

[감사의 글]

먼저 학부생 1학년 시절부터 지금까지 저를 지도해주셨던 권영근 교수님께 진심으로 감사드립니다. 저의 부족한 모습에도 항상 격려와 조언을 아낌없이 주셨기에 지금의 제가 될 수 있었습니다. 그리고 바쁘신 와중에 시간을 내어 심사해주신 김종면 교수님과 윤석훈 교수님께 감사드립니다. 굉장히 긴장된 상태로 시작했던 논문 심사였지만 교수님들의 배려가 있었기에 제가 발표를 통해 전달하고자 했던 내용을 충분히 설명할 수 있었습니다. 그리고 심사 중에 해주셨던 값진 조언들은 가슴 깊이 새겨 앞으로의 연구를 수행하는데 있어서 항상 생각하고 잊지 않도록 하겠습니다. 또한 항상 저를 응원해주신 부모님께 감사드립니다. 그동안 타지 생활을 하느라 자주 연락도 못하고 만나지도 못했음에도 아낌없는 지원과 믿음을 보내주셔서 제가 무사히 졸업을 할 수 있었습니다. 마지막으로 제가 석사 과정을 마치기까지 도움주신 모든 분께 감사드립니다.

[국문요약]

유전 알고리즘 탐색을 이용한 랜덤 탐색 대비 효율적인 기계학습 시스템 구성을 위한 연구

기계학습 모델을 활용하는 분류 및 회귀 문제 해결을 위한 시스템은 크게 기계학습 모델에 적용하기 위해 데이터를 가공하는 전처리 과정과 데이터를 적용하여 학습 및 예측을 진행하는 기계학습 모델로 구성된다. 여기서 전처리 과정은 일반적으로 데이터 크기 변환, 특징 구축, 특징 선택, 차원 축소 등을 포함하며, 분류 문제의 경우 학습 데이터의 클래스 비율을 조절하는 데이터 재조정을 추가로 고려할 수 있다. 이처럼 기계학습 시스템은 각 단계별로 여러 종류의 알고리즘과 파라미터를 고려할 수 있기 때문에 다양한 형태로 구성을 할 수 있다. 하지만 시스템의 합리적인 성능을 위해서는 주어진 데이터와 잘 부합하는 시스템 구성의 해를 찾는 과정이 필수적으로 요구된다. 본 논문에서는 유전 알고리즘의 진화 과정 중 각 기계학습 모델 그룹의 평균 제곱 오차 정보를 기반으로 특정 기계학습 모델 그룹에 대한 집중 탐색을 유도하는 방법을 제안하였으며, 랜덤 탐색과의 비교를 통해 여러 데이터에 대한 결과로부터 제안한 유전 알고리즘의 평균 성능과 안정성을 확인한다. 마지막으로 결론에서는 집중 탐색을 유도하는 방법의 한계점을 확인하여 본 논문에서 제안하는 탐색 방법의 개선점을 확인한다.

주요어 : 기계학습, 전처리, 유전 알고리즘

목 차

그 림 목 차

표 목 차

1. 서론

Machine Learning(ML) 모델을 활용하는 분류 및 회귀 문제 해결을 위한 시스템은 크게 ML 모델에 적용하기 위해 데이터를 가공하는 전처리 과정과 데이터를 적용하여 학습 및 예측을 진행하는 ML 모델로 구성된다. 여기서 전처리 과정은 일반적으로 Data Scaler(DS), Feature Construction(FC), Feature Selection(FS), Dimension Reduction(DimR) 등을 포함하며, 분류 문제의 경우 학습 데이터의 클래스 비율을 조절하는 Data Rebalancing(DR)을 추가로 고려할 수 있다. 각 단계별로 다양한 알고리즘과 파라미터를 고려할 수 있기 때문에 시스템은 다양한 형태로 구성을 할 수 있게 된다. 하지만 시스템의 합리적인 성능을 위해서는 주어진 데이터와 잘 부합하는 시스템 구성의 해를 찾는 과정이 필수적으로 요구된다.

일반적으로 사전 정보가 주어지지 않은 미지의 공간에 대한 탐색 과정은 합리적인 성능을 만족할 때까지 반복적인 실험을 통해 여러 해를 비교해야만 한다. 이때 전역적인 영역에 대한 탐색 방법으로는 Grid Search(GS), Random Search(RS), Genetic Algorithm(GA) 등의 방법이 있다. 여기서 GS는 탐색 영역에 대한 모든 경우의 수를 고려하는 방법으로 최적의 성능을 보장한다는 장점이 있지만 탐색 영역의 크기에 따라 탐색 종료까지 요구되는 시간적 비용의 크기가 매우 커질 수 있는 단점이 있다. 시스템 구성의 해는 각 과정의 알고리즘 종류 및 파라미터를 고려하기 때문에 GS의 활용은 현실적으로 어려움이 존재한다. 반면에 RS는 임의의 영역에 대해 탐색하는 방법으로 GS와는 다르게 최적의 성능을 보장하지 않지만 시간적 비용을 제한할 수 있는 장점이 있다. RS는 시스템 구성의 해를 탐색하는 문제에서 반복적인 임의의 구성을 통해 가장 성능이 좋았던 해를 취하는 방식으로 적용된다. 마지막으로 GA는 RS와 마찬가지로 모든 영역에 대한 탐색을 하지 않는 방법이다. 하지만 RS와는 다르게 탐색 과정에서 GA의 구현 내용에 따라 특정 영역으로 탐색을 집중하도록 할 수 있으며, 이를 통해 탐색의 수렴 속도를 높이거나 더 나은 성능의 해를 찾도록 유도할 수 있다. 반대로 잘못된 영역으로 탐색을 집중할 수도 있으며, 이와 같은 경우에는 임의의 영역을 탐색하는 RS보다 좋지 못한 결과가 나타날 수 있다.

본 논문에서는 GA의 진화 과정 중 각 ML 모델 그룹의 Mean Squared Error(MSE) 정보를 기반으로 특정 영역에 대한 집중 탐색을 유도하는 방법을 제안하였다. 실험은 Python 환경에서 진행되었으며, 시스템 구성에 사용된 모든 알고리즘은 Scikit-Learn[1]과 Imbalanced-Learn[2] 라이브러리를 활용하였다. 제안된 방법의 성능 검증을 위해 문제의 특성에 대한 사전 정보가 없는 상황에서 최적의 해를 찾기 위한 두 가지 방법 GA와 RS를 각각 구현하여 비교 실험을 진행 하였으며, 이때 사용된 데이터는 모두 UCI Machine Learning Repository[3]에서 제공하는 데이터를 활

용하였다. 마지막으로 여러 데이터에 대한 결과로부터 기존 RS 방법 대비 평균적으로 더 나은 성능을 확인할 수 있었다.

본 논문은 다음과 같이 구성된다. 2장에서는 본 논문에서 고려한 ML 시스템의 구성에 대해 설명하고, 3장에서는 본 논문에서 제안하는 GA에 대해 소개한다. 그리고 4장에서는 제안된 방법에 대한 결과 비교 및 분석을 진행하며, 마지막으로 5장에서는 결론과 함께 향후 개선점을 제시하며 마무리한다.

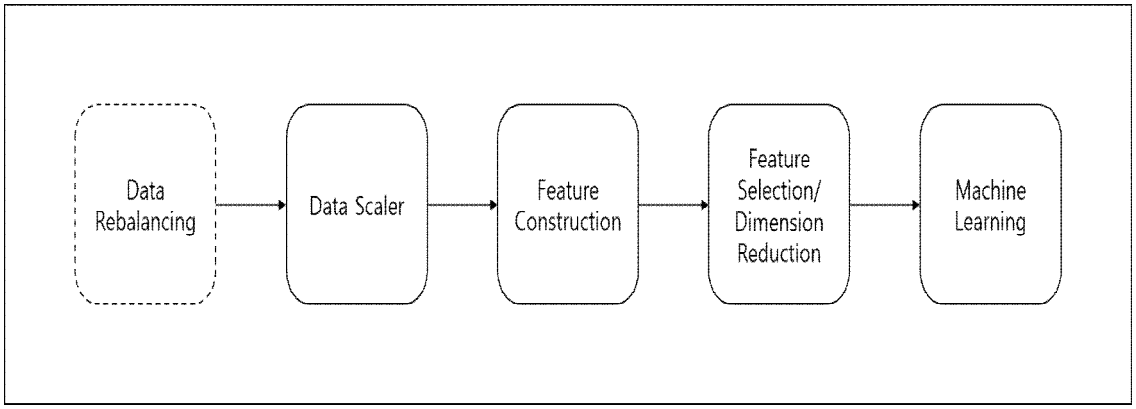

그림 2 시스템 구성

2. 기계학습 시스템

본 논문에서 고려한 ML 시스템을 구성하는 각 과정을 그림 1에 나타내었다.

2.1 Data Rebalancing

DR은 분류 문제에 대해서만 고려한다. 분류 문제에서는 학습 데이터가 특정 클래스에 치우쳐져 있을 경우, 클래스 불균형으로 인해 편향된 학습 결과가 발생할 수 있다. 이와 같은 현상을 방지하기 위해 DR으로부터 사전에 학습 데이터의 소수 클래스 대비 다수 클래스 비율을 균형하게 조정한다. DR의 모든 알고리즘에서는 소수 클래스 대비 다수 클래스 비율을 [0.75,1.00]의 값으로 고려하였으며, 비율과 관련된 파라미터는 sampling_strategy로 표현된다. 비율과 관련된 파라미터 이외의 파라미터는 기본 값을 사용하였다.

2.1.1 Over-Sampling

Over-Sampling은 소수 클래스의 데이터를 복제, 또는 가공하여 다수 클래스와의 비율을 줄이는 것을 목적으로 하며, 오히려 추가된 데이터로 인해 과대적합이 발생할 수 있다. 본 논문에서 고려한 Over-Sampling과 관련된 알고리즘 및 파라미터를 표 1에 나타내었다.

표 1 Over-Sampling 알고리즘 및 파라미터

Algorithm	Parameter	
	Name	Value
Adaptive Synthetic[4]	sampling_strategy	[0.75, 1.00]
	n_neighbors	5
SVM SMOTE	sampling_strategy	[0.75, 1.00]
	n_neighbors	5
	m_neighbors	10
Random	sampling_strategy	[0.75, 1.00]
SMOTE[5]	sampling_strategy	[0.75, 1.00]
	k_neighbors	5
K-Means SMOTE[6]	sampling_strategy	[0.75, 1.00]
	k_neighbors	2
Borderline SMOTE[7]	sampling_strategy	[0.75, 1.00]
	k_neighbors	5
	m_neighbors	10
	kind	borderline-1

2.1.2 Under-Sampling

Under-Sampling은 다수 클래스에서 데이터를 선별하여 소수 클래스와의 비율을 줄이는 것을 목적으로 하며, 오히려 제외된 데이터로 인해 정보의 손실이 발생할 수 있다. 본 논문에서 고려한 Under-Sampling과 관련된 알고리즘 및 파라미터를 표 2에 나타내었다.

표 2 Under-Sampling 알고리즘 및 파라미터

Algorithm	Parameter	
	Name	Value
Condensed Nearest Neighbor[8]	sampling_strategy	[0.75, 1.00]
	n_neighbors	1
	n_seeds_S	1
Edited Nearest Neighbor[9]	sampling_strategy	[0.75, 1.00]
	n_neighbors	3
	kind_sel	all
All K-Nearest Neighbor[10]	sampling_strategy	[0.75, 1.00]
	n_neighbors	3
	kind_sel	all
Neighbourhood Cleaning Rule[11]	sampling_strategy	[0.75, 1.00]
	n_neighbors	3
	threshold_cleaning	0.5
NearMiss[12]	sampling_strategy	[0.75, 1.00]
	n_neighbors	3
Random	sampling_strategy	[0.75, 1.00]
Tomek's links[13]	sampling_strategy	[0.75, 1.00]
Instance Hardness Threshold[14]	sampling_strategy	[0.75, 1.00]
	cv	5
Repeated Edited Neartest Neighbour[10]	sampling_strategy	[0.75, 1.00]
	n_neighbors	3
	max_iter	100
	kind_sel	all

2.2 Data Scaler

ML 모델은 학습 과정에서 입력 데이터의 분포에 따라 수렴속도 또는 성능에 차이가 발생할 수 있다. DS에서는 입력 데이터 값의 크기 또는 값 사이의 간격을 조절하여 전체적인 ML 시스템의 성능 향상을 기대할 수 있다.

2.2.1 Standard Scaler

Standard Scaler는 데이터의 각 특징별로 전체 샘플에 대해서 적용되며, 각 특징의 평균을 0, 분산을 1로 변환한다.

2.2.2 Normalizer

Normalizer는 데이터의 각 샘플별로 전체 특징에 대해서 적용되며, 본 논문에서는 각 샘플을 L2-Norm으로 나눈다.

2.2.3 Min Max Scaler

Min Max Scaler는 데이터의 각 특징별로 전체 샘플에 대해서 적용되며, 본 논문에서는 각 특징의 최댓값을 1, 최솟값을 0으로 변환한다.

2.3 Feature Construction

FC는 DS를 통해 변환된 데이터를 기반으로 서로 연관성을 갖는 특징을 조합하여 새로운 특징을 생성한다. 따라서 FC는 여러 개의 알고리즘이 선택될 수 있다. 생성할 특징의 개수를 클러스터링 알고리즘에서는 n_clusters로 표현하고 최소 2개의 특징을 생성하도록 하였으며, 이외의 알고리즘에서는 n_components로 표현하고 최소 1개의 특징을 생성하도록 하였다. 그리고 특징의 최대 생성 개수는 입력 특징 수(n_features)에서 10을 나눈 값으로 설정하였다. 만약 n_features/10의 값이 최소 생성 개수 이하일 경우 최소 생성 개수에서 1을 더한 값으로 설정하였다. 마지막으로 생성할 특징과 관련된 파라미터 이외의 파라미터는 기본 값을 사용하였다. 본 논문에서 고려한 FC와 관련된 알고리즘 및 파라미터를 표 3에 나타내었다.

2.4 Feature Selection / Dimension Reduction

FS/DimR은 DS를 통해 변환된 데이터와 앞의 FC에서 생성된 데이터를 활용하여 ML 모델에 적용하기 위해 특징 수를 줄이는 마지막 가공 단계이다. 본 논문에서 고려한 FS/DimR과 관련된 알고리즘 및 파라미터를 표 4에 나타내었다.

2.5 Machine Learning

ML은 앞에서 가공된 데이터를 적용하여 학습 및 예측을 진행한다. ML 모델의 종류와 파라미터는 본 논문에서 구성하는 시스템의 성능을 결정짓는 가장 중요한 요소이다. 본 논문에서는 대표적으로 활용되는 기계학습 알고리즘인 Multi Layer Perceptron(MLP), K-Nearest Neighbors(KNN), Support Vector Machine(SVM), Decision Tree(DT), Random Forest(RF) 다섯 가지의 알고리즘을 고려하였다. Scikit-Learn 라이브러리에서는 MLP 알고리즘에서 은닉 층과 관련된 정보를 hidden_layer_sizes로 표현하며 배열의 크기를 은닉 층의 개수로, 배열의 요소 값을 은닉 층의 노드 수로 사용한다. 본 논문에서는 편의를 위해 은닉 층의 개수를 num_layers, 은닉 층의 노드 수를 hidden_size로 표현하였다. 또한 트리 기반 알고리즘에서 분류 문제와 회귀 문제에 따라 파라미터의 구성에 차이가 있기 때문에 본 논문에서 고려한 ML과 관련된 알고리즘 및 파라미터를 표 5와 표 6으로 나누어 나타내었다.

표 3 FC 알고리즘 및 파라미터

Algorithm	Parameter	
	Name	Value
Feature Agglomeration	n_clusters	[2, n_features/10]
	affinity	euclidean
	linkage	ward
	pooling_func	mean
Truncated SVD[15]	n_components	[1, n_features/10]
	algorithm	randomized
	n_iter	5
PCA[16]	n_components	[1, n_features/10]
	svd_solver	auto
	iterated_power	auto
Gaussian Random Projection	n_components	[1, n_features/10]
	eps	0.1
Sparse Random Projection	n_components	[1, n_features/10]
	density	auto
	eps	0.1

표 4 FS/DimR 알고리즘 및 파라미터

Algorithm	Parameter	
	Name	Value
Variance Threshold	threshold	0
PCA	n_components	[0.9, 1.0]
	svd_solver	auto
	iterated_power	auto

표 5 ML 분류 알고리즘 및 파라미터

Algorithm	Parameter	
	Name	Value
MLP	num_layers	[1, 50]
	hidden_size	[5, 100]
	activation	[identity, logistic, tanh, relu]
	solver	[adam, lbfgs, sgd]
	max_iter	[100, 1000]
	alpha	[0.0001, 0.001]
	learning_rate	[constant, invscaling, adaptive]
	learning_rate_init	[0.001, 0.01]
KNN	n_neighbors	[2, 20]
	weights	[uniform, distance]
	algorithm	[auto, ball_tree, kd_tree, brute]
	leaf_size	[10, 50]
SVM	C	[1, 100]
	kernel	[linear, poly, rbf, sigmoid]
	max_iter	[100, 1000]
	gamma	scale
DT	max_depth	[20, 100]
	criterion	[gini, entropy]
	splitter	[best. random]
RF	n_estimators	[2, 100]
	criterion	[gini, entropy]
	max_depth	[20, 100]

표 6 ML 회귀 알고리즘 및 파라미터

Algorithm	Parameter	
	Name	Value
MLP	num_layers	[1, 50]
	hidden_size	[5, 100]
	activation	[identity, logistic, tanh, relu]
	solver	[adam, lbfgs, sgd]
	max_iter	[100, 1000]
	alpha	[0.0001, 0.001]
	learning_rate	[constant, invscaling, adaptive]
	learning_rate_init	[0.001, 0.01]
KNN	n_neighbors	[2, 20]
	weights	[uniform, distance]
	algorithm	[auto, ball_tree, kd_tree, brute]
	leaf_size	[10, 50]
SVM	C	[1, 100]
	kernel	[linear, poly, rbf, sigmoid]
	max_iter	[100, 1000]
	gamma	scale
DT	max_depth	[20, 100]
	criterion	[mse, mae]
	splitter	[best. random]
RF	n_estimators	[2, 100]
	criterion	[mse, mae]
	max_depth	[20, 100]

```
Generate p initial chromosomes;
for i ← 1 to g
    calculate convergence rate of each model groups;
    select one model group m;

    // parent1.mse < parent2.mse
    select two chromosomes parent1, parent2 in m;

    offspring ← Crossover(parent1, parent2);

    offspring ← Mutation(offspring);

    // worst : worst case in m
    if worst.mse < offspring.mse
        continue;
    else if parent2.mse < offspring.mse
        worst ← offspring;
    else
        parent2 ← offspring;

// best : best case in chromosomes
return best
```

그림 3 GA 의사코드

3. 유전 알고리즘

다양한 ML 모델을 고려하는 시스템에서 GA를 활용하여 합리적인 성능을 보이는 구성을 찾기 위해서는 데이터와 잘 부합하는 ML 모델 영역으로 탐색을 유도하는 것이 중요하다. 일반적으로 GA의 탐색은 각 세대마다 진화를 통한 자손(offspring)의 생성으로 이루어지는데, 이때 기존의 GA는 자손의 생성을 위한 부모의 선택 과정에서 현재 세대를 구성하는 전체 염색체(chromosome)의 성능을 기준으로 부모를 선택하게 된다. 하지만 각 ML 모델은 신경망 기반, 트리 기반 등 방법의 차이에 따라 최종적으로는 데이터에 가장 잘 부합하는 ML 모델이지만 상대적으로 평균이 낮거나 편차가 크기 때문에 선택 과정에서 불리하게 작용하여 탐색이 충분히 진행되지 못하는 경우가 발생한다.

따라서 본 논문에서는 각 ML 모델 그룹에 대한 편차의 크기를 잠재력으로 해석하였으며, 이를 반영하기 위해 각 ML 모델 그룹 내에서 가장 낮은 MSE에서 가장 높은 MSE를 나눈 값을 수렴률(convergence rate)로 정의하였다. 그리고 일반적인 GA의 선택 과정 앞에 수렴률을 기준으로 하는 ML 모델 그룹에 대한 선택 과정을 추가하여 상대적으로 편차가 큰 ML 모델에 대한 탐색을 유도하는 방법을 제안하였다. 본 논문에서 제안한 GA의 의사코드를 그림 2에 나타내었으며, GA와 관련된 설정 파라미터는 표 7에 표기하였다.

3.1 초기화

본 논문에서 사용되는 GA의 염색체는 각각의 ML 시스템을 의미한다. GA의 초기에는 생성된 부모 염색체가 존재하지 않기 때문에 자손을 생성하는 과정을 진행할 수가 없다. 따라서 임의의 영역에 대한 탐색을 진행하는 RS 방법으로 p개의 인구수(population)만큼 초기 염색체를 생성한다. 이때 각 ML 모델 그룹의 크기를 동일하게 생성한다.

3.2 선택

부모 염색체를 선택하기에 앞서, 룰렛 휠을 통해 구해진 각 모델 그룹에 대한 확률 분포에서 하나의 모델 그룹을 선택한다. 그리고 선택된 모델 그룹에 속하는 모든 염색체에 대해서 다시 한 번 룰렛 휠을 통해 각 염색체에 대한 확률 분포를 구한 후, 두 개의 부모 염색체를 선택하게 된다. 이때 선택 된 부모 염색체 중 MSE 값이 작은 염색체를 *parent1*, 큰 염색체를 *parent2*로 사용한다.

3.2.1 룰렛 휠

GA의 선택 과정에서 일반적으로 활용되는 방법으로 모든 대상의 뽑힐 확률을 합한 값이 1이 되도록 하는 방법이다. 본 논문에서는 모델 그룹 선택 과정에서는 각 모델 그룹의 수렴률을 기반으로 확률분포를 구성하고, 부모 염색체의 선택 과정에서는 선택된 모델 그룹 내의 각 염색체들의 MSE를 기준으로 확률분포를 구성한다. 그리고 기준으로 활용하는 정보간의 크기 차이를 고려하여 조절(adjust) 변수를 적용하여 가장 큰 확률이 가장 작은 확률의 *adjust*배가 되도록 하였다.

3.3 교차

앞에서 선택된 두 부모 염색체를 섞는 과정을 통해 하나의 자손을 생성한다. *parent1*을 기준으로 진행하며, [0, 1] 사이의 난수를 각 전처리 과정별로 생성한 후, 교차 확률(crossover probability)을 만족하는 각 과정에 대해서 *parent2*의 알고리즘 및 파라미터와 교차를 적용한다. 본 논문에서는 ML의 교차는 고려하지 않는다.

3.4 변이

앞에서 생성된 자손 내에서 알고리즘 혹은 파라미터를 변화시킨다. 두 부모 염색체로부터 하나의 자손을 생성하는 과정이다. 본 논문에서는 두 가지 방식의 변이를 구현하였는데, 하나는 알고리즘 및 파라미터 변이이고, 다른 하나는 파라미터 변이이다. 알고리즘 변이는 임의의 알고리즘으로 새롭게 설정하고 알고리즘의 종류에 맞는 파라미터 또한 새롭게 설정하게 된다. 그리고 파라미터 변이는 현재 파라미터 값을 기준

으로 고려하는 파라미터 범위에서 10%의 크기로 가우시안 분포만큼 변화시킨다. 만약 변이 대상이 실수가 아닌 문자열의 경우에는 변이를 적용하지 않는다. 본 논문에서는 염색체의 다양성을 위해 변이 확률(mutation probability)을 만족하는 경우에는 알고리즘 및 파라미터 변이를 적용하고, 변이 확률을 만족하지 않는 경우에는 파라미터 변이를 적용 하였다. 교차 과정과 마찬가지로 ML 알고리즘의 변이는 진행하지 않으며, 대신 ML 알고리즘의 파라미터 변이를 항상 적용하였다.

3.5 대치

앞에서 변이된 자손(*offspring*)의 MSE와 기존 염색체의 MSE를 비교하여 대치 및 생략을 진행한다. 비교는 동일한 ML 모델 그룹 내에서 가장 성능이 좋지 않은 염색체(*worst*), 그리고 두 부모 염색체 중 성능이 좋지 않은 염색체(*parent2*)와 진행된다. 만약, *offspring*의 MSE 값이 *worst*보다 높은 경우에는 대치 과정을 생략하며, *worst*보다 낮지만 *parent2*보다 높으면 *worst*와 대치한다. 마지막으로 *parent2*보다 높은 경우에는 *parent2*와 대치한다.

표 7 GA 설정 파라미터

Name	Value
인구수 (population)	150
진화 횟수 (generation)	850
조절 강도 (adjust)	4
교차 확률 (crossover probability)	0.5
변이 확률 (mutation probability)	0.3

4. 결과 비교 및 분석

4.1 실험 데이터

본 논문에서는 제안하는 방법의 범용성을 검증하기 위해 서로 다른 성격의 여러 데이터를 사용하였으며, 분류 문제 4가지와 회귀 문제 4가지로 총 8가지의 데이터로 구성하였다. 실험에 사용된 각 데이터의 이름, 입력 크기, 출력 크기 그리고 문제 구분을 표 8에 나타내었다. 그리고 각 데이터는 전체 샘플을 3:1:1의 비율로 학습 데이터(train data), 검증 데이터(validation data), 테스트 데이터(test data)를 구성하였다.

4.2 실험 방법

본 논문에서는 탐색 횟수를 1000회로 제한하였으며, GA는 초기화 과정에서 150회, 진화 과정에서 850회를 진행한다. 그리고 결과 비교를 위해 GA와 동일한 ML을 고려하는 RS 방법과 하나의 ML 모델만 고려하는 RS 탐색을 진행하였다. 보다 안정된 결과 비교를 위해 앞의 과정은 21회씩 진행하였으며, 성능 지표는 MSE를 활용 하였다.

표 8 데이터 상세

Name	Input	Output	Type
Cardiotocography(FHR Pattern) (CTGP)	21	10	Classification
Cardiotocography(Fatal State) (CTGS)	21	3	Classification
Wireless Indoor Localization (WIL)	7	4	Classification
Steel Plates Faults (SPF)	27	7	Classification
Combined Cycle Power Plant (CCPP)	4	1	Regression
Airfoil Self Noise (ASN)	5	1	Regression
QSAR Aquatic Toxicity (QAT)	8	1	Regression
QSAR Fish Toxicity (QFT)	6	1	Regression

4.3 실험 결과

GA 대비 각 방법의 성능 비교를 위해 각 방법의 테스트 데이터에 대한 평균 MSE에서 GA의 테스트 데이터에 대한 평균 MSE를 나눈 값을 MSE 비율로 정의하였다. 이때의 평균은 21회 과정의 평균을 의미한다. 그림 3은 각 방법의 전체 데이터에 대한 평균 MSE 비율을 나타내었으며, 표 9는 각 데이터에 대한 MSE 결과의 TOP 3를 나타낸 것이다. 또한, 각 방법의 성능 편차 비교를 위해 각 방법의 테스트 데이터에 대한 MSE의 표준편차에서 GA의 테스트 데이터에 대한 MSE의 표준편차를 나눈 값을 표준편차 비율로 정의 하였다. 그림 4는 각 방법의 전체 데이터에 대한 평균 표준편차 비율을 나타내었으며, 표 10은 각 데이터에 대한 표준편차 결과의 TOP 3를 나타낸 것이다.

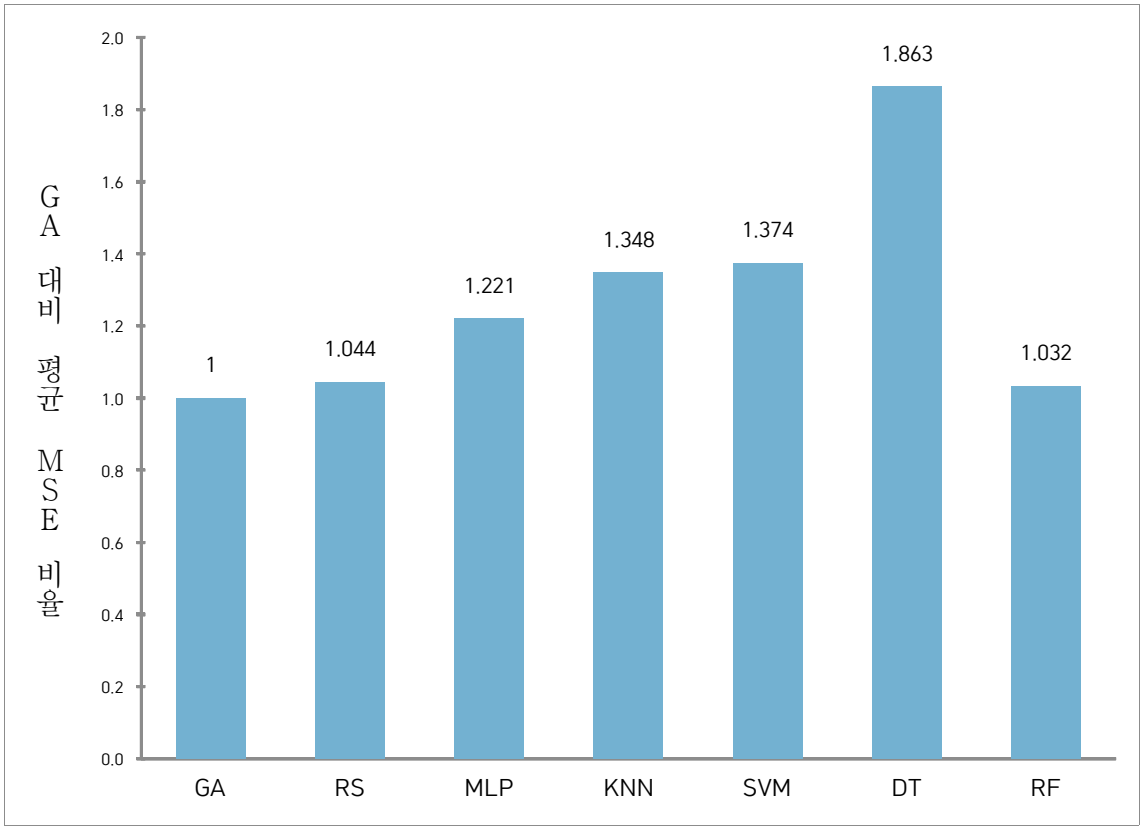

그림 4 각 방법의 GA 대비 평균 MSE 비율

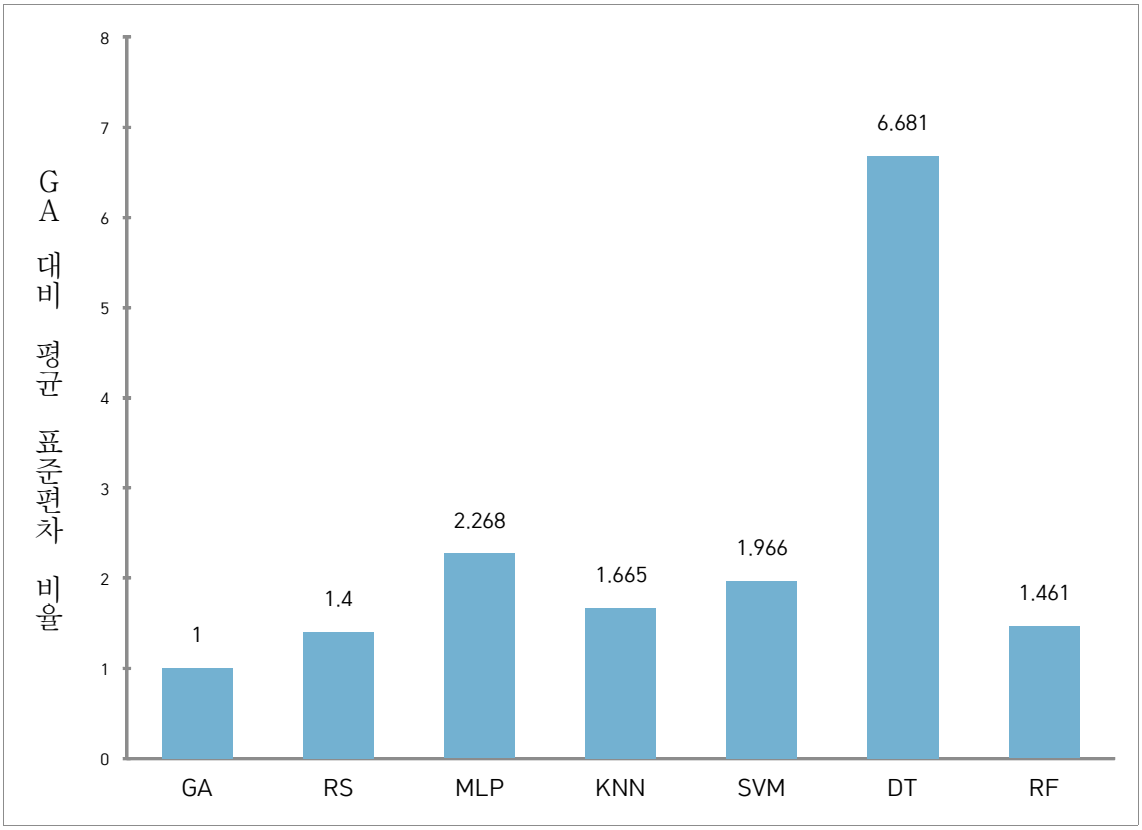

그림 5 각 방법의 GA 대비 평균 표준편차 비율

표 9 각 데이터에 대한 성능 TOP 3

	TOP 1	TOP 2	TOP 3
CTGP	GA	MLP	RS
CGTS	RF	RS	GA
WIL	GA	RF	RS
SPF	RF	RS	GA
CCPP	RF	GA	RS
ASN	GA	RS	KN
QAT	GA	RF	RS
QFT	KNN	GA	RS

그림 3에서 평균 MSE 비율에서 가장 성능이 좋은 GA는 비교 대상인 RS에 비해 약 4%의 더 나은 성능을 보였으며, 표 9를 확인하였을 때 CGTS, SPF, CCPP, QFT 데이터를 제외한 나머지 경우에서 TOP1의 결과가 나타났다. 하지만 CGTS, SPF 등의 데이터에서는 오히려 RS보다 못한 결과가 나타났다. 그리고 RS는 평균 MSE 비율에서 단일 모델인 RF보다 GA 대비 약 1% 부족한 성능이 나타났으며 표 9에서도 TOP1에서는 발견되지 못하는 한계점이 있었다. 또한 그림 4의 결과를 통해 제안된 GA가 다른 탐색 방법에 비해 보다 안정적인 성능을 보이는 것으로 나타났다.

5. 결론

본 논문에서는 GA의 진화 과정 중 각 ML 모델 그룹의 MSE 정보를 기반으로 특정 영역에 대한 집중 탐색을 유도하는 방법을 제안하였으며, 여러 데이터 및 탐색 방법에 따른 MSE 결과의 평균과 표준편차를 기반으로 성능 비교 및 검증을 진행 하였다. 그리고 분석의 편의를 위해 사용한 평균 MSE 비율과 평균 표준편차 비율을 통해 제안한 GA의 평균 성능과 안정성이 더 우수함을 확인하였으며, 주된 비교 대상이었던 RS에 비해 CGTS와 SPF 두 데이터에서 부족한 성능을 통해 개선의 여지가 있음을 확인할 수 있었다.

참 고 문 헌

[1] Scikit-Learn. https://scikit-learn.org/

[2] Imbalanced-Learn. https://imbalanced-learn.readthedocs.io/en/stable/index.html

[3] Machine Learning Repository, https://archive.ics.uci.edu/ml/index.php

[4] He, Haibo, Yang Bai, Edwardo A. Garcia, and Shutao Li. "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," In IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), pp. 1322-1328, 2008.

[5] N. V. Chawla, K. W. Bowyer, L. O.Hall, W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," Journal of artificial intelligence research, 321-357, 2002.

[6] Felix Last, Georgios Douzas, Fernando Bacao, "Oversampling for Imbalanced Learning Based on K-Means and SMOTE" Information Sciences 465 (2018) 1-20.

[7] H. Han, W. Wen-Yuan, M. Bing-Huan, "Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning," Advances in intelligent computing, 878-887, 2005.

[8] P. Hart, "The condensed nearest neighbor rule," In Information Theory, IEEE Transactions on, vol. 14(3), pp. 515-516, 1968.

[9] D. Wilson, Asymptotic" Properties of Nearest Neighbor Rules Using Edited Data," In IEEE Transactions on Systems, Man, and Cybernetrics, vol. 2 (3), pp. 408-421, 1972.

[10] I. Tomek, "An Experiment with the Edited Nearest-Neighbor Rule," IEEE Transactions on Systems, Man, and Cybernetics, vol. 6(6), pp. 448-452, June 1976.

[11] J. Laurikkala, "Improving identification of difficult small classes by balancing class distribution," Springer Berlin Heidelberg, 2001.

[12] I. Mani, I. Zhang. "kNN approach to unbalanced data distributions: a case study involving information extraction," In Proceedings of workshop on learning from imbalanced datasets, 2003.

[13] I. Tomek, "Two modifications of CNN," In Systems, Man, and Cybernetics, IEEE Transactions on, vol. 6, pp 769-772, 2010.

[14] D. Smith, Michael R., Tony Martinez, and Christophe Giraud-Carrier. "An instance level analysis of data complexity." Machine learning 95.2 (2014): 225-256.

[15] Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions Halko, et al., 2009.
[16] Tipping, M. E., and Bishop, C. M. (1999). “Probabilistic principal component analysis”. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 61(3), 611-622.

ABSTRACT

A Study on Efficient Machine Learning System Construction Using Genetic Algorithm Search

A system for solving classification and regression problems using a machine learning model is composed of a machine learning model that performs learning and prediction by applying data and pre-processing processes that process data to apply to the machine learning model. The pre-processing generally includes data scaler, feature construction, feature selection, dimensionality reduction, etc. In the case of classification problems, data rebalancing that adjusts the class ratio of the training data may be further considered. Since various algorithms and parameters can be considered for each step, the system can be configured in various forms. However, for a reasonable performance of the system, the process of finding a solution of the system configuration that matches the given data is essential. In this paper, we proposed a method to induce the intensive search for a specific area based on the mean squared error information of each group of machine learning models during the evolution of the genetic algorithm. And it was confirmed from the results of various data that the proposed genetic algorithm has excellent average performance and stability. It was also confirmed that there was room for improvement. Through the case of showing insufficient performance compared to the random search which was the main comparison target.

Key words : Machine Learning, Preprocessing, Genetic Algorithm