랜덤 탐색과 유전 알고리즘 탐색을 이용한 효율적 기계학습 방법 연구

이경태, 권영근
울산대학교 컴퓨터공학과
lmuffin@naver.com, kwonyk@ulsan.ac.kr

A Study on Efficient Machine Learning Method Using Random Search and Genetic Algorithm Search

Kyung-Tae Lee, Young-Keun Kwon
Dept. of Computer Engineering, Ulsan University

요 약

기계학습 모델을 이용한 분류 및 회귀 문제해결에는 다양한 전처리 알고리즘 및 기계학습 모델이 활용된다. 하지만 합리적인 성능을 위해서는 주어진 데이터에 따라 적절한 알고리즘 조합에 대한 탐색 및 최적화 과정이 필수적이다. 본 논문에서는 최적의 알고리즘 조합을 탐색하는 방법 중 랜덤 탐색과 유전 알고리즘 탐색 방법을 구현하고 8가지 데이터에 대한 성능 비교를 통해 여러 기계학습 모델을 고려하는 탐색 방법의 필요성을 보인다.

1. 서론

기계학습 모델을 이용한 분류 및 회귀 문제해결에는 다양한 전처리 알고리즘 및 기계학습 모델이 활용된다. 하지만 합리적인 성능을 위해서는 주어진 데이터에 따라 적절한 알고리즘 조합에 대한 탐색 및 최적화 과정이 필수적이다. 일반적으로 문제 해결을 위한 데이터의 처리과정은 Data Scaler(DS), Feature Construction(FC), Feature Selection(FS), Machine Learning(ML) 등으로 구성되며, Classification 문제의 경우 타겟 클래스의 비율을 조정하는 Data Rebalancing(DR) 과정이 추가로 고려될 수 있다.

최적의 알고리즘 조합을 찾기 위해서는 일반적으로 Grid Search, Random Search(RS)[1], Genetic Algorithm(GA)[2] 등의 방법을 고려 할 수 있다. 여기서 Grid Search는 탐색 가능한 모든 영역을 탐색하는 방법으로, 최적의 성능을 보장하지만 시간적 비용이 크다. 반면에 RS는 임의의 영역을 탐색하는 방법으로, 최적의 성능을 보장하진 않지만 시간적 비용을 제한 할 수 있기 때문에 상대적으로 시간적 비용에서 이점이 있다. 마지막으로 GA을 통한 탐색 방법은 RS와 마찬가지로 모든 영역을 탐색하지 않는 방법으로, 최적의 성능을 보장하진 않지만 마찬가지로 시간적 비용을 제한 할 수 있기 때문에 시간적 비용에서 이점이 있다. 또한, RS와는 달리 GA은 탐색 방식을 어떻게 구성하느냐에 따라 특정 영역으로의 탐색을 집중하도록 할 수 있으며, 이에 따라 수렴 속도를 높이거나 최적의 성능으로 유도 할 수 있게 된다.

본 논문에서는 최적의 알고리즘 조합을 찾기 위한 두 가지 방법 GA와 RS를 각각 구현하여 문제특성에 대한 사전 정보가 없는 상황에서 단일 기계학습 모델을 이용한 탐색 대비 GA와 RS 기반 탐색 방법의 성능을 비교하는 실험을 진행하였으며, 데이터의 종류에 따라 최적의 기계학습 모델의 종류가 다를 수 있음을 확인하였다. 또한 이를 위해 여러 종류의 기계학습 모델을 고려하여 알고리즘 조합을 탐색하는 방법의 필요성을 제시한다.

2. 문제 정의

<표 1>은 본 논문에서 고려한 각 데이터 처리과정의 알고리즘 종류를 나타낸다.

Rebalancing 과정은 Classification 문제에 대해서만 고려되며, 가장 적은 종류의 데이터와 가장 많은 종류의 데이터의 비율이 최소 75%를 만족하도

<표 1> 데이터 처리 과정 별 알고리즘 종류

Processing	Algorithm	Remarks
DR	Adaptive Synthetic	다수 클래스 대비 소수 클래스 비율 0.75-1.00 조정
	SVM SMOTE	
	Random	
	KMeans SMOTE	
	Borderline SMOTE	
	Condensed Nearest Neighbour	
	Edited Nearest Neighbour	
	NearMiss	
DS	StandardScaler	
	MinMaxScaler	
	Normalizer	
FC	PCA	기존 입력 데이터 크기의 10% 혹은 2-3개 생성
	Truncated SVD	
	Feature Agglomeration	
	Gaussian Random Projection	
	Sparse Random Projection	
FS	Variance Threshold	
ML	MLP	
	KNN	
	SVM	
	DT	
	RF	

록 파라미터를 조정하였다. ML 과정은 Multi Layer Perceptron(MLP), K-Nearest Neighbor(KNN), Support Vector Machine(SVM), Decision Tree(DT), Random Forest(RF) 모델을 사용하였다.

3. 탐색 방법

3.1. RS

각 기계학습 모델(MLP, KNN, SVM, DT, RF)별로 탐색시도만큼 DR, DS, FC, FS 각 단계에 적용할 알고리즘을 임의로 선택한 조합을 생성한다.

3.2. GA

<그림 1>은 구현한 GA의 의사 코드를 나타낸다. 본 논문에서는 기존의 일반적인 GA와 다르게 유력한 기계학습 모델의 집중적인 탐색을 유도하기 위해 하위 성능의 두 그룹사이의 T-Test 결과를 바탕으로 최하위 ML 그룹을 진화 대상에서 제외하는 방법을 적용하였다.

초기화 과정에서는 RS로 임의의 알고리즘 조합을 생성하며, 진화과정 중 자손을 생성하기 전에 각 ML 그룹의 평균 mean squared error(mse)를 기준으로 값이 큰 두 그룹에 대해 T-Test를 진행하며, 이때 p-value 값이 0.05 이하의 값을 만족하면 가장 값이 큰 ML 그룹은 진화 대상에서 제외하게 된다. 자손을 생성하는 과정에서는 두 개의 부모 해를 선택하며, 이때 두 해 중에서 mse 값이 더 작은 해를 s1, 다른 하나를 s2로 사용한다.

4. 실험 결과

```
Generate p initial chromosomes;
for i ← 1 to g
  Remove low performance model group by t-test result;
  Select two chromosomes s1, s2;
  offspring ← Crossover(s1, s2);
  offspring ← Mutation(offspring);
  if worst.mse < offspring.mse
    continue;
  else if s2.mse < offspring.mse
    worst ← offspring;
  else
    s2 ← offspring;
return best chromosome in population;
```

<그림 1> 구현하는 GA 의사코드

4.1. 실험 데이터

실험에서 사용된 데이터는 모두 UCI Machine Learning Repository에서 제공하는 데이터를 활용하였다. <표 2>은 실험에 사용된 각 Dataset의 이름, Input의 개수, Output의 개수, 문제유형을 보인다. 각 데이터는 전체 샘플을 3:1:1의 비율로 Train Set, Validation Set, Test Set을 구성하였다.

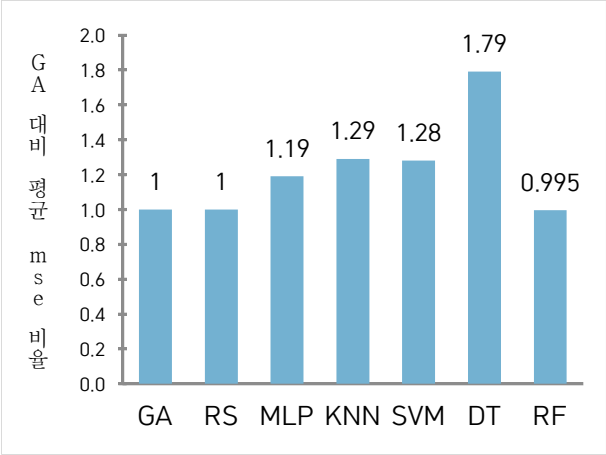
4.2. 분석 결과

실험에 사용 된 GA는 초기화 과정에서 <표 1>에 제시된 5가지의 ML방법에 대해 50개의 초기 해를 생성하며, 950세대 진화를 한다. RS는 5가지의 알고리즘에 대해 1000개의 알고리즘 조합을 생성한다. 마지막으로 앞의 두 방법과의 성능 비교를 위해 제시된 5가지의 각 알고리즘별로 RS를 진행하여 각각 1000개의 조합을 생성한다. 평가지표는 mse를 사용하였다. 앞의 과정은 각 데이터에 대해 21회씩 진행하였다.

GA 대비 각 방법의 성능 비교를 위해 Test Set 결과(mse)에서 GA의 결과를 나눈 값을 성능 비율로 사용하였다. <그림 2>는 각 방법의 전체 데이터에 대한 평균 mse 비율을 나타낸 것이며, <표 3>은 각 데이터에 대한 결과의 Top 3를 나타낸 것이다.

<표 2> 데이터의 이름, Input의 개수, Output의 개수 및 문제 구분

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



<그림 2> 각 방법의 GA 대비 평균 mse 비율

평균 mse 비율에서 가장 낮은 값을 나타내는 RF는 WIL, ASN, QFT 데이터를 제외한 모든 경우에서 가장 mse 값이 나타났고, QFT 데이터에서만 TOP 3 내에 포함되지 못하였다. 최적의 조합을 찾기 위해 다양한 종류의 ML을 고려했던 RS와 GA는 평균 mse 비율에서 두 번째로 낮은 mse를 보였으며, 마찬가지로 전체 데이터에서 TOP 3 내에 6번씩 포함되었고, 특히 QFT 데이터에서는 GA가 가장 낮은 mse를 보였다. 평균 mse 비율에서 상대적으로 좋지 못한 결과가 나왔던 SVM(GA대비 평균 28% 높은 mse)과 MLP(GA대비 평균 19% 높은 mse)는 WIL과 ASN에 대해서는 각각 TOP 1의 결과가 나타났다.

<표 4>는 GA의 진행 과정 중, 각 기계학습 모델이 선택된 횟수를 나타낸다. CTGP, CTGS, SPF, CCPP, QFT 데이터의 경우 GA가 적응적으로 탐색 빈도를 조절하여 가장 적합했던 기계학습 모델과 탐색 횟수가 가장 많았던 모델이 일치했다. 반면에 이외의 데이터에서는 가장 적합한 기계학습 모델일지라도 초기 그룹의 성능이 낮아, 일찍 탐색 고려대상에서 제외되어 결국 탐색 횟수가 저조하게 나타났다.

<표 3> 데이터에 따른 성능 TOP 3

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

<표 4> GA 과정 중 각 기계학습 모델 선택 횟수

	MLP	KNN	SVM	DT	RF
CTGP	34	69	54	52	**791**
CTGS	41	68	59	42	**790**
WIL	27	**382**	171	26	**394**
SPF	98	107	131	31	**633**
CCPP	36	29	40	37	**858**
AFN	53	36	55	64	**792**
QAT	137	275	294	27	267
QFT	47	42	**813**	40	58

5. 결론

본 논문은 문제특성에 대한 사전정보가 없는 상황에서 단일 기계학습 모델 대비 GA와 RS 기반 탐색 방법의 성능을 비교하는 실험을 진행하였다. RF는 대부분의 데이터에 대해서 안정적인 성능을 보였지만 QFT 데이터에 대한 결과와 같이 성능이 좋지 못한 경우도 있기 때문에, 최적의 알고리즘 조합을 찾기 위해서는 여러 알고리즘을 고려해야한다. 또한, ASN 및 WIL 데이터와 같이 문제에 적합한 기계학습 모델에 대한 탐색을 강화하도록 유도하는 GA에 대한 개선이 필요하다.

참고문헌

[1] James Bergstra, Yoshua Bengio, “Random Search for Hyper-Parameter Optimization”, Journal of Machine Learning Research, 13, 281-305, 2012

[2] Chih-Hung Wu, Gwo-Hshiung Tzeng, Yeong-Jia Goo, Wen-Chang Fang, “A real-valued genetic algorithm to optimize the parameters of support vector machine for predicting bankruptcy”, Expert Systems with Applications, 32, 397-408, 2007

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