https://doi.org/10.5392/JKCA.2019.19.11.567

머신러닝을 위한 불균형 데이터 처리 방법 : 샘플링을 위주로

Handling Method of Imbalance Data for Machine Learning : Focused on Sampling

이규남*, 임종태**, 복경수***, 유재수**
충북대학교 빅데이터학과*, 충북대학교 정보통신공학과**, 원광대학교 SW융합학과***

Kyunam Lee(rbska56455@gmail.com)*, Jongtae Lim(jtlim@chungbuk.ac.kr)**,
Kyoungsoo Bok(ksbok@wku.ac.kr)***, Jaesoo Yoo(yjs@chungbuk.ac.kr)**

요약

최근 학계, 산업계 등에서 접하는 기존의 문제를 머신러닝을 통해 해결하려는 시도가 증가하고 있다. 이에 따라 이탈, 사기탐지, 장애탐지 등 일반적이지 않은 상황을 머신러닝으로 해결하기 위한 다양한 연구가 이어지고 있다. 대부분의 일반적이지 않은 환경에서는 데이터가 불균형하게 분포하며, 이러한 불균형한 데이터는 머신러닝의 수행과정에서 오류를 야기하므로 이를 해결하기 위한 불균형 데이터 처리 기법이 필요하다. 본 논문에서는 머신러닝을 위한 불균형 데이터 처리 방법을 제안한다. 제안하는 방법은 샘플링 방법을 중심으로 다수 클래스(Major Class)의 모집단 분포를 효율적으로 추출하도록 검증하여 머신 러닝을 위한 불균형 데이터 문제를 해결한다. 본 논문에서는 성능평가를 통해 제안하는 기법이 기존 기법에 비해 성능이 우수함을 보인다.

■ 중심어 : | 불균형데이터 | 머신러닝 | 언더샘플링 | 오버샘플링 | 이상탐지 |

Abstract

Recently, more and more attempts have been made to solve the problems faced by academia and industry through machine learning. Accordingly, various attempts are being made to solve non-general situations through machine learning, such as deviance, fraud detection and disability detection. A variety of attempts have been made to resolve the non-normal situation in which data is distributed disproportionately, generally resulting in errors. In this paper, we propose handling method of imbalance data for machine learning. The proposed method to such problem of an imbalance in data by verifying that the population distribution of major class is well extracted. Performance Evaluations have proven the proposed method to be better than the existing methods.

■ keyword : | Imbalance Data | Machine Learning | Under Sampling | Over Sampling | Anomaly Detection |

I. 서론

최근 데이터 저장 기법의 비약적인 발달과 처리 비용의 감소에 따라 학계, 산업계 등에서 기존의 문제를 머신러닝을 통해 해결하는 시도가 다양해지고 있고 경영에 도움이 되는 사례가 점점 더 많아지고 있다. 이에 따

* 이 논문은 2017년도 정부(과학기술정보통신부)의 재원으로 한국연구재단-차세대정보·컴퓨팅기술개발사업(No. NRF-2017M3C4A7069432), 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원(No. 2019R1A2C2084257), 그리고 2019년도 정부(과학기술정보통신부)의 재원(No.B0101-15-0266, 실시간 대규모 영상 데이터 이해·예측을 위한 고성능 비주얼 디스커버리 플랫폼 개발)의 지원을 받아 수행된 연구임

접수일자 : 2019년 08월 23일 심사완료일 : 2019년 10월 11일
수정일자 : 2019년 10월 11일 교신저자 : 유재수, e-mail : yjs@chungbuk.ac.kr

라 머신러닝은 점점 더 중요해지고 있다. 특히, 머신러닝에 기반을 두어서 분류 예측하는 문제는 산업계에서 널리 사용되고 있다. 분류 예측 문제에서 가장 흔하게 만나는 문제는 데이터 불균형 문제이다. 데이터 불균형 문제는 사기 탐지, 허가되지 않은 네트워크 침입 탐지, 장애 탐지, 의료 진단 등 다양한 분야에서 데이터가 불균형하게 분포하여 발생하는 문제를 의미한다. 또한 클래스란 분류의 기준이 되는 대상 데이터를 의미한다. 예를 들어, 클래스는 사기 탐지에서는 사기 유무, 네트워크 침입 탐지에서는 침입 유무, 장애 탐지에서는 장애 유무 등을 구분할 수 있는 데이터이다. 또한 데이터의 수가 많은 클래스를 다수 클래스(Major Class)라 하고 데이터의 수가 적은 클래스를 소수 클래스(Minor Class)라 정의한다.

데이터 불균형 문제의 가장 큰 문제는 데이터 분류 예측 시 기계 학습에 부정적인 영향을 준다는 것이다. 의사 결정모형이나 신경망 모형은 훈련 데이터 셋이 등급 간 균일하게 분포한다고 가정한다[1]. 그러나 위에서 서술한대로 사기 탐지, 허가되지 않은 네트워크 침입 탐지, 장애 탐지, 의료 진단 등 실제 세상에서 발생하는 많은 분류 예측 문제들은 등급 간 데이터가 균일하게 분포하지 않으며 비율 또한 매우 낮은 경우가 대다수이다. 이러한 데이터 불균형 문제는 오류를 야기하게 된다[1]. 이러한 데이터 불균형 문제를 극복하기 위해 다양한 기법들이 연구되고 있다[1-11]. 주로 데이터를 균형 있게 조작하는 데이터 수준 방법(Data-level approach)과 보다 클래스 불균형에 민감하게 반응하는 알고리즘 수준 방법이 존재한다.

데이터 불균형이 초래하는 오류를 극복하기 위해 시도된 방법 중 데이터 레벨 접근 방법은 데이터 불균형으로 인해 발생되는 원인을 제거하여 문제를 해결하는 방법이다. 즉, 데이터 레벨 접근 방법은 데이터를 균형 있게 맞추어 학습을 시키는 방법이다. 데이터 레벨 접근방법은 크게 두 가지 접근 방향을 가진다. 첫째는 샘플링 방법이다. 샘플링은 소수 클래스의 데이터를 다수 클래스만큼 증폭시키는 오버 샘플링과 다수 클래스를 소수 클래스만큼 감소시키는 언더 샘플링으로 구성되며 일반적으로 대표적인 오버 샘플링에는 랜덤 오버 샘플링(ROS), 소수 클래스 오버 샘플링(SMOTE)이 존재하며, 언더 샘플링에는 랜덤 언더 샘플링(RUS)이 존재한다.

두 번째는 피쳐 셀렉션(Feature selection) 방법이다. 데이터의 고차원성과 관련성 없는 피쳐들은 불균형 데이터셋에서 분류기의 성능을 저하시키고 잘못 분류되는 비율을 증가시키기 때문에 피쳐 셀렉션이 필요하다[1]. Zheng et al[6]에서는 피쳐 셀렉션 기법들인 information gain, chi square, correlation oefficient, odds ratio을 명시적으로 결합한 피쳐 셀렉션을 사용했다.

본 논문에서는 머신러닝을 위한 불균형 데이터 처리 방법을 제안한다. 제안하는 기법은 데이터 레벨 접근 방법을 개선한 실무에 적합한 샘플링 방법을 제안한다. 제안하는 방법은 샘플링 방법을 중심으로 다수 클래스(Major Class)의 모집단 분포를 효율적으로 추출하도록 검증하여 머신 러닝을 위한 불균형 데이터 문제를 해결한다.

본 논문의 구성은 다음과 같다. 2절에서는 관련연구를 조사 분석한다. 3절에서는 제안하는 샘플링 기법을 기술한다. 4절에서는 성능평가를 통해 제안하는 기법의 타당성을 보인다. 5절에서는 결론 및 향후 연구를 설명한다.

II. 관련 연구

언더 샘플링이란 다수 클래스의 데이터를 임의로 선택하여 소수 클래스의 수와 맞도록 데이터 비중을 조절하여 모델링에 활용하는 방법이다. 언더 샘플링의 방법에는 크게 11가지가 존재한다. 첫 번째로, ENN(Editetd Nearest Neighbours)방법은 KNN을 사용해 다수 클래스 데이터를 축소. 이웃한 데이터 중 자신과 같은 클래스보다 다른 클래스의 데이터가 많을 경우 해당 데이터는 제외하는 방법이다[12]. 두 번째 방법은 RENN[13]으로 리샘플링된 데이터 집합의 변화가 없을때까지 ENN을 반복적으로 수행하는 방법이다. 세 번째 방법은 ALLKNN 방법[13]으로 ENN을 변형한 방식으로 k값을 설정하여 1<= i <= k 범위의 모든 i-NN을 수행한다. 각 수행 중 한번이라도 자신의 클래스와 판정이 다

른 경우 해당 데이터는 제외한다. 네 번째 방법은 TomekLinks 방법[14]으로 거리를 기반으로 데이터 쌍의 집합을 정의하고 이 집합에 속한 데이터들을 노이즈로 간주하여 제외하는 방법이다. i 클래스에 속하는 점을 xi, j 클래스에 속하는 점을 xj라 할 때 xi와 xj의 거리는 d(xi, xj)라고 정의한다. 이때 d(xi, xk)<d(xi,xj) 혹은 d(xj, xk)<d(xi, xj)가 되는 샘플 xk가 없다면 이를 토맥 링크로 정의하고 제거한다. 다섯 번째 방법은 CNN(Condensed Nearest Neighbour) 으로 다수 클래스 데이터 포인트와 가장 가까운 데이터가 소수 클래스인 다수 클래스 데이터 외 에는 모두 삭제하는 방법이다. 여섯 번째 방법은 OneSideSelection 방법[15]으로 토맥링크를 수행한 뒤 추가로 CNN(Condensed Nearest Neighbour)를 수행하는 방법이다. 일곱 번째 방법은 NCR(Condensed Nearest Neighbour + Edited Nearest Neighbours) 방법[16]으로 데이터 포인트와 가까운 데이터들이 자신과 다른 클래스일 경우 노이즈로 판단, 해당 데이터를 제외시키는 방법이다.. 여덟 번째 방법은 NearMiss-1 방법[17]으로 다수 클래스 데이터 포인트와 가장 가까운 소수 클래스 데이터 3개와의 평균 거리를 계산하고 평균 거리가 가장 작은 데이터를 비율에 맞게 남기고 그 외에는 삭제하는 방법이다. 아홉 번째 방법은 NearMiss-2 방법[17]으로 NearMiaa-1과는 반대로 가장 먼 데이터를 3개 뽑도록 변형한 방식이다. 열 번째 방법은 NearMiss-3 방법[17]으로 NearMiss-1 방법과 동일하게 작동하지만 소수 클래스 데이터 N개를 사용자가 지정해주는 방법이다. 열한 번째 방법은 랜덤 언더 샘플링 방법으로 다수 클래스 데이터를 소수 클래스 수 만큼 샘플링하는 방법이다.

오버 샘플링은 언더 샘플링과는 반대로 소수 클래스의 데이터를 다수 클래스에 맞추어 증폭하는 방법으로 크게 세 가지 방법이 존재한다. 첫 번째는 랜덤 오버 샘플링으로 소수 클래스를 랜덤하게 선택하여 복제하는 방법이다. 이 방법은 소수 클래스를 증폭시킬 때 우연성에 의지하기 때문에 결과가 좋지 않을 가능성이 크다. 두 번째 방법은 SMOTE 방법[18]으로 소수 클래스에 속하는 데이터를 합성하여 데이터 수를 증폭시킨다. 합성된 데이터는 벡터 공간에서 기존의 소수 클래스 데이터들 사이에 위치하여 데이터 간 사이를 보간 하는 지능형 오버 샘플링 방법이다. 세 번째 방법은 ADASYN 방법[19]으로 데이터의 분포를 고려하여 클래스 간의 경계면에 가까울수록 많은 소수 클래스의 데이터를 합성하는 방법이다.

언더 샘플링과 오버 샘플링을 함께 결합한 앙상블 방법도 존재한다. 주로 쓰이는 방법으로는 크게 두 가지이다. 첫 번째 방법은 SMOTE와 TomekLinks를 결합하여 함께 사용하는 방법[20]이다. 두 번째 방법은 SMOTE와 Editetd Nearest Neighbours를 함께 사용하는 방법[21]이다.

Ⅲ. 제안하는 샘플링 방법

1. 데이터 샘플링 및 검증

제안하는 기법은 언더 샘플링 접근 방법으로 훈련 데이터 셋의 다수 클래스를 학습을 통해 제거하는 방법이다. [그림 1]은 학습에 사용할 데이터 셋이다. 데이터 셋은 모델 학습에 사용할 Train data set과 성능 평가에 사용할 Test data set으로 구성되어 있다. 그리고 각 데이터 셋은 다수 클래스와 소수 클래스로 분할되어 있다. 이 때, 그림과 같이 다수 클래스는 소수 클래스보다 크기가 큰 불균형 데이터이다. 제안하는 훈련 검증 방법은 다수 클래스를 소수 클래스 와 비중을 같게 언더 샘플링 하는 방법이다.

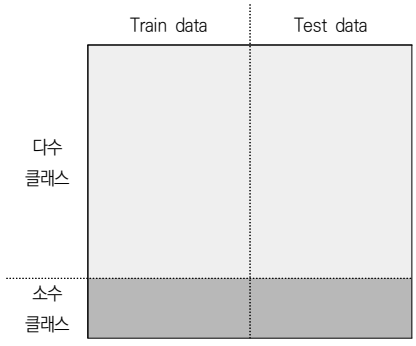

그림 1. 학습에 사용할 데이터 셋

[그림 2]는 제안하는 언더 샘플링 접근 방법으로 훈련 데이터 셋의 다수 클래스를 학습을 통해 제거하는

훈련 검증방법의 구체적인 학습 데이터 구성도이다. 먼저, 학습에 사용할 Train data의 소수 클래스만큼 있는 ① 데이터 크기만큼 다수 클래스에서 ② 데이터를 추출한다. 그 뒤 ③ 데이터로 모델 학습을 진행한다. 학습된 모델로 ②과 ④을 분간할 수 있는지 테스트 해보고 정확도가 사용자가 정한 임계값 부근일 경우 ② 데이터는 ②+③+④ 데이터 전체를 대표할 수 있다고 주장할 수 있다. 즉, 샘플링한 임계값 부근인 이유는 정확도가 너무 낮아도, 높아도 모델이 ②과 ④ 데이터를 분간 할 수 있을 정도로 분포가 차이가 나기 때문이다.

보다 자세히 표현하면, ③ 데이터로 진행하는 학습은 기존의 라벨링 된 정보가 아니라 랜덤하게 새롭게 클래스의 레이블을 부여하여 진행한다. 예를 들어, 비이탈과 이탈로 분할된 클래스의 레이블이 있을 때 이탈이 비중이 너무 적어서 비이탈이 다수 클래스인 경우를 가정해 보자. 비이탈으로만 라벨링 된 ③ 데이터의 라벨링을 랜덤하게 비이탈과 이탈로 임의로 다시 부여한다. 이는 원래 같은 클래스의 라벨링된 데이터로 학습을 하여 만들어진 모델은 오버피팅되어 새로운 ③ 데이터와 유사한 테스트 셋이 들어오면 성능이 좋지 않게 나올 것임을 고려한 것이다. 즉, 이 모델로 ②과 ④ 데이터를 분간할 수 있는지 테스트하여 두 데이터를 분류하는데 성능이 좋게 나오면 ②와 ④ 데이터는 차이가 있다고 간주하여 샘플링이 잘못된 것이라 주장할 수 있다. 즉, 다수 클래스의 데이터를 분할하는 모델이므로 너무 성능이 좋으면 두 데이터는 상이하다고 판단할 수 있으며 성능이 너무 안 좋으면 두 데이터가 차이가 없어서 성능이 안 좋은 것인지, 모델이 너무 심하게 오버 피팅이 되어있어서 안 좋게 나온 것인지에 대한 판단을 다시 수행해야하므로 상식선에서, 같은 성질의 데이터를 랜덤하게 분류 시 정확도가 (1/ 클래스 속성 수)의 비중이 나올 것이라 간주하여 두 데이터를 랜덤하게 분류할 경우 나올 수 있는 정확도 부근에서 종료한다. 본 논문에서의 임계값은 0.48 〈 threshold 〈 0.52로 설정하여 테스트를 진행하였다.

기존 기법들은 샘플링 후 훈련, validation data set을 통해 정확도 상승을 확인할 수밖에 없다. 제안하는 기법을 사용하면 샘플링 후 훈련 셋으로만 검증을 통해 주장할 수 있으며 특히, 소수 클래스 데이터를 사용하지 않고 다수 클래스만을 이용해 타 기법들 대비 빠르게 훈련 검증을 통해 주장할 수 있다.

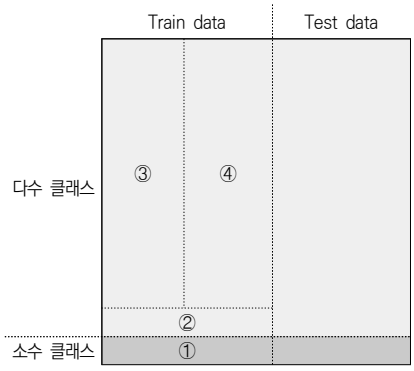

그림 2. 훈련 검증 학습 데이터 구성도

2. 제안하는 데이터 샘플링 기법의 특징

[그림 3]은 일반적인 샘플링 기법을 적용한 모델링 플로우의 구조도와 제안하는 기법의 모델링 플로우 구조도를 보여준다. 좌측의 파란부분이 일반적인 샘플링 기법이다. 일반적인 샘플링 기법은 어떤 샘플링 기법이 성능을 올릴 수 있는지 모든 샘플링 방법에 대한 조합을 검토해야한다. 우측의 붉은 부분의 제안하는 기법의 모델링은 샘플링 된 데이터가 전체 모집단을 대표한다고 주장할 수 있으므로 모든 조합을 검토해보는 시간이 감소하여 빠르게 모델링이 가능하다.

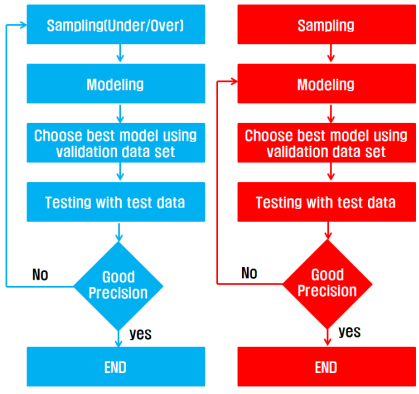

그림 3. 샘플링 모델링 구조도

보다 상세한 내용은 그림 4를 통해 설명을 할 수 있

다. [그림 4]는 일반적인 언더 샘플링 기법과 제안하는 기법의 차이를 보여준다. 일반적인 언더 샘플링 기법은 다수 클래스의 데이터를 감소시키는 방법이다. 그런데, 그 과정에서 우리는 우리가 가진 데이터의 분포를 제대로 잘 축약해서 감소시키지 못하는 경우가 생길 수 있다. 어떤 데이터를 삭제할지 선택하는 과정에서 오히려 잘못된 샘플링으로 인해 잘못된 분포를 가지게 될 수 있다. 이는 잘못된 데이터를 활용하게 되었을 때, Class를 분할하기 위해 경계선을 찾는 과정에서 오류를 야기할 수 있다. 제안하는 기법은 샘플링을 통해 축약한 분포가 이전의 데이터 분포를 잘 축약하는지 검증을 수행하여 잘못된 샘플링을 할 가능성을 감소시킨다. 샘플링한 것이 데이터 분포를 잘 축약하였는지 검증을 통해 잘 축약하지 못하였다고 판단될 경우 해당 샘플링을 배제한다.

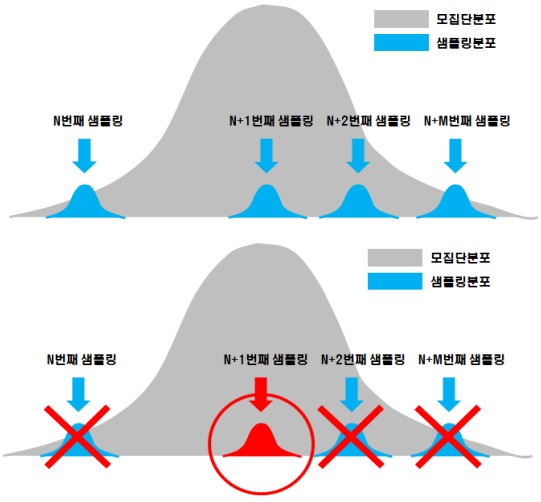

그림 4. 일반적인 언더 샘플링과 제안하는 기법 차이

산업계에서 분석을 진행하다보면 시간이나 인력이 제약된 제한된 상황에서 기계학습이 수행되어야하는 경우가 존재한다. 불균형 데이터 셋을 분석하는 경우 언더 샘플링 혹은 오버 샘플링, 코스트에 민감한 알고리즘 방법 중 하나를 선택해서 진행해야한다. 문제는 시간과 인력이 제한된 상황에는 빠르게 분석을 수행해야 하기 때문에 발생한다. 데이터마다 적합한 샘플링 방법이 다르기 때문에 실험적으로 매번 적합성을 확인하기에는 시간적 여유가 부족하다. 따라서 분석가의 경험에 비추어 방법을 선택하거나 가장 논리적인 방법을 선택해야한다. 제안하는 기법은 훈련 검증을 통해 다수 클래스의 데이터를 샘플링한 것이 다수 클래스를 대표한다고 가정하여 자원이 제약적인 상황에서 빠르게 분석해야만 하는 상황에 활용할 수 있다. 그리고 다수 클래스만을 이용해서 훈련 검증을 진행하면서 분석가가 휴리스틱하게 정확도를 조정하면서 수행 시간을 보다 단축할 수 있다.

Ⅳ. 성능 평가

성능평가는 Windows 10 Home 64bit Intel(R) Core(TM) i5-7200U CPU 8GB RAM에서 수행됐으며 Python 3.7.1로 구현됐다. 제안하는 기법의 성능평가를 위해 [9]에서 사용된 데이터 셋[22] 중 k-fold과 유사한 데이터 등을 제외하고 성능평가에 사용하였다. [표 1]은 성능평가에 사용된 데이터의 정보이다.

표 1. 성능평가에 사용된 데이터

Data_name	Row Length	Number of variable	imbalance ratio
pages-blocks0	5,472	10	0.102
segment0	2,308	19	0.143
yeast1	1,484	8	0.289
vehicle	846	18	0.235
pima	768	8	0.349
winsconsin	683	9	0.350
haberman	306	3	0.265
glass0	214	9	0.327
ecoli-0_vs_1	200	7	0.350
iris0	150	4	0.333

성능평가를 위해 2/3은 train data set으로 1/3은 test data set으로 분할하였다. 본 논문에서 측정하는 모델의 성능은 Precision(정확도), Recall(재현율), Error ratio(에러 비율), F score로 측정하였다. Precision, Recall, Error ratio은 F score가 같을 경우 참고하기 위함이며 최종적으로 모델의 성능은 F score를 기반으로 측정하였다. 모델의 Precision은 (식 1)과 같고 Recall은 (식 2)와 같으며 Error ratio은 (식 3)과 같으며 F score는 (식 4)와 같다.

$$Precision = \frac{True\ Positive\ Count}{False\ Positive\ Count + True\ Positive\ Count} \quad (식\ 1)$$

$$Recall = \frac{True\ Postive\ Count}{False\ Negative\ Count + True\ Positive\ Count} \quad (식\ 2)$$

$$Error\ ratio = \frac{Correct\ Count}{Total\ Count} \quad (식\ 3)$$

$$F\ Score = 2 \times (\frac{Precision \times Recall}{Precision + Recall}) \quad (식\ 4)$$

(식 1)의 True Positive Count는 모델이 소수 클래스라고 예측한 것이 실제로 소수 클래스인 수를 나타낸다. False Positive Count는 모델이 다수 클래스라고 예측한 것이 실제로 다수 클래스 인 수를 나타낸다. (식 2)의 False Negative Count는 소수 클래스라고 예측한 것이 실제로 다수 클래스 인 수를 나타낸다. (식 3)의 Correct Count는 모델이 소수 클래스로 예측한 것이 실제로 소수 클래스인 수와 다수 클래스로 예측한 것이 실제로 다수 클래스인 수의 합이다. Total Count는 전체 수이다.

[표 2]는 17개 기법 별 10개 데이터의 정확도, 재현율, 에러비율, F1 점수의 평균 테이블이다. 각 데이터 별로 기법을 적용하여 나온 정확도와 재현율, 에러 비율과 F1 score를 평균을 낸 것이다. [표 2]에 대한 세부 내용은 부록 1에서 보다 자세히 확인할 수 있다.

표 2. 각 데이터 별 기법의 정확도

샘플링 방법	Precision	Recall	Error_rate	F score
Recommended Method	0.7599	0.8887	0.1260	0.8151
Smote + TomekLinks	0.8051	0.8255	0.1111	0.8143
ALLKNN	0.7439	0.9192	0.1484	0.8107
Repeated Edited Nearest Neighbours	0.7439	0.9192	0.1484	0.8107
NCR	0.7445	0.9136	0.1518	0.8092
Random Under Sampling	0.7574	0.8808	0.1302	0.8088
Smote + ENN	0.7565	0.8779	0.1327	0.8080
Edited Nearest Neighbours	0.7942	0.8305	0.1140	0.8079
ADASTN(Adaptive Synthetic Sampling)	0.7882	0.8296	0.1239	0.8073
Tomek Links	0.8199	0.8024	0.1071	0.8066
smote	0.7908	0.8208	0.1188	0.8044
Random Over Sampling	0.8059	0.7892	0.1163	0.7958
Near Miss-3	0.7433	0.8313	0.1516	0.7815
One Side Selection	0.7659	0.8060	0.1630	0.7693
Condensed Nearest Neighbour	0.7106	0.8489	0.1843	0.7608
Near Miss-1	0.6331	0.8749	0.2362	0.7023
Near Miss-2	0.6306	0.8964	0.2726	0.6973

성능평가 결과, 제안하는 기법이 F1 점수가 가장 높아서 가장 우수한 기법으로 측정되었다. 특히, 제안하는 기법은 Random Under Sampling 기법을 기반으로 하는 방법으로 기존의 Random Under Sampling 기법보다 정확도, 재현율, 에러 비율 모두 성능이 우수하게 측정되었다.

Ⅴ. 결론

본 논문에서는 머신러닝을 위한 불균형 데이터 처리 방법을 제안했다. 제안하는 기법은 샘플링 방법을 중심으로 다수 클래스 의 모집단 분포를 잘 추출하도록 검증하여 불균형 데이터를 처리하는 방법이다. 제안하는 방법은 샘플링 방법을 통해 데이터 불균형으로부터 초래되는 오류를 해결했다. 16가지 기존 기법들과 10개의 데이터 셋으로 검증해본 결과 제안하는 기법이 우수한 기법임을 성능평가를 통해 검증하였다. 본 논문에서 제안된 방법은 데이터가 불균형한 환경에서의 이탈, 사기탐지, 장애탐지 등에 활용될 수 있다. 향후 연구에서는 현재 성능평가를 수행한 F지표 이외에도 다양한 성능 평가를 통해 제안하는 기법의 우수성을 증명할 것이다. 또한 현재 본 논문에서는 샘플링 기법을 통해 불균형 문제는 해결하였는데, 불균형 데이터 셋을 모델링 방법으로 해결하는 기법을 연구하여 성능을 향상시킬 예정이다.

참 고 문 헌

[1] Shaza M. Abd Elrahman and Ajith Abraham, "A review of class imbalance problem," Journal of

Network and Innovative Computing, Vol.1, pp.332-340, 2013.

[2] Haibo He and Edwardo A. Garcia, "Learning from imbalanced data," IEEE Transactions on Knowledge & Data Engineering, Vol.21, No.9, pp.1263-1284, 2009.

[3] Arpit Singh and Anuradha Purohit, "A survey on methods for solving data imbalance problem for classification," International Journal of Computer Applications, Vol.127, No.15, pp.37-41, 2015.

[4] Rushi Longadge, Snehlata S. Dongre, and Latesh Malik, "Class imbalance problem in data mining review," Internation Journal of Computer Science and Network, Vol.2, No.1, pp.1-6, 2013.

[5] Joffrey L. Leevy, Taghi M. Khoshgoftaar, Richard A. Bauder, and Naeem Seliya, "A survey on addressing high-class imbalance in big data," Journal of Big Data, Vol.5, No.1, pp.1-30, 2018.

[6] Zhaohui Zheng, Xiaoyun Wu, and Rohini Srihari, "Feature selection for text categorization on imbalanced data," ACM Sigkdd Explorations Newsletter, Vol.6, No.1, pp.80-89, 2004.

[7] Peng Cao, Dazhe Zhao, and Osmar Zaiane, "An optimized cost-sensitive SVM for imbalanced data learning," Proc. Pacific-Asia conference on knowledge discovery and data mining, pp.280-292, 2013.

[8] Peng Cao, Dazhe Zhao, and Osmar R. Zaïane, "A PSO-based cost-sensitive neural network for imbalanced data classification," Proc. Pacific-Asia conference on knowledge discovery and data mining, pp.452-463, 2013.

[9] Alberto Fernándeza, Salvador García, María José del Jesus, and Francisco Herrera, "A study of the behaviour of linguistic fuzzy rule based classification systems in the framework of imbalanced data-sets," Fuzzy Sets and Systems, Vol.159, No.18, pp.2378-2398, 2008.

[10] S. Picek, A. Heuser, A. Jovic, S. Bhasin, and F. Regazzoni, "The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations," 2018.

[11] Z. Chen, Q. Yan, H. Han, S. Wang, L. Peng, L. Wang, and B. Yang, "Machine learning based mobile malware detection using highly imbalanced network traffic," Information Sciences, Vol.433, pp.346-364, 2018.

[12] Dennis L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," IEEE Transactions on Systems, Man, and Cybernetics, Vol.3, pp.408-421, 1972.

[13] I. Tomek, "An experiment with the edited nearest-neighbor rule," IEEE Transactions on systems, Man, and Cybernetics, Vol.6, No.6, pp.448-452, 1976.

[14] I. Tomek, "Two Modifications of CNN," IEEE Transactions on Systems, Man and Cybernetics, Vol.6, No.11, pp.769-772, 1976.

[15] Kubat, Miroslav, and Stan Matwin, "Addressing the curse of imbalanced training sets: one-sided selection," Proc. International Conference on Machine Learning, Vol.97, pp.179-186, 1997.

[16] J. Laurikkala, "Improving identification of difficult small classes by balancing class distribution," Proc. Conference on Artificial Intelligence in Medicine in Europe - Artificial Intelligence in Medicine, pp.63-66, 2001.

[17] Mani, Inderjeet and I. Zhang, "kNN approach to unbalanced data distributions: a case study involving information extraction," Proc. workshop on learning from imbalanced datasets, Vol.126, 2003.

[18] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," Journal of artificial intelligence research, Vol.16, No.1, pp.321-357, 2002.

[19] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," Proc. IEEE International Joint Conference on Neural

Networks, pp.1322-1328, 2008.
[20] Batista, Gustavo EAPA, Ana LC Bazzan, and Maria Carolina Monard, "Balancing Training Data for Automated Annotation of Keywords: a Case Study," Proc. Workshop on Bioinformatics, 2003.
[21] Batista, Gustavo EAPA, Ronaldo C. Prati and Maria Carolina Monard, "A study of the behavior of several methods for balancing machine learning training data," SIGKDD Explorations, Vol.6, No.1, pp.20-29, 2004.
[22] https://sci2s.ugr.es/keel/imbalanced.php?order=ir#sub10, 2019.8.18.

저 자 소 개

이 규 남(Kyunam Lee) 준회원

- 2016년 2월 : 충북대학교 정보통계학과(이학사)
- 2016년 2월 : 충북대학교 심리학과(문학사)
- 2018년 2월 : 충북대학교 빅데이터학과(공학석사)
- 2018년 3월 ~ 현재 : 미래에셋대우 빅데이터팀 매니저

〈관심분야〉 : 빅데이터, 추천 시스템, 머신러닝, 그래프분석, 주가예측, 로보어드바이저, 트레이딩 시스템, 뇌공학 등

임 종 태(Jongtae Lim) 정회원

- 2009년 2월 : 충북대학교 정보통신공학과 (공학사)
- 2011년 2월 : 충북대학교 정보통신공학과(공학석사)
- 2015년 8월 : 충북대학교 정보통신공학과(공학박사)
- 2015년 9월~ 2019년 8월 : 충북대학교 정보통신공학과 Postdoc
- 2019년 10월 ~ 현재 : 충북대학교 전자정보대학 정보통신공학부 초빙조교수

〈관심분야〉 : 소셜 미디어, 빅데이터, 시공간 데이터베이스, 위치 기반 서비스 등

복 경 수(Kyoungsoo Bok) 종신회원

- 2009년 2월 : 충북대학교 수학과(이학사)
- 2000년 2월 : 충북대학교 정보통신공학과(공학석사)
- 2005년 8월 : 충북대학교 정보통신공학과(공학박사)
- 2005년 3월 ~ 2008년 2월 : 한국과학기술원 정보전자연구소 Postdoc
- 2008년 3월 ~ 2011년 2월 : 가인정보기술 연구소 연구원
- 2011년 3월 ~ 2019년 8월 : 충북대학교 전자정보대학 정보통신공학부 초빙교수
- 2019년 9월 ~ 현재 : 원광대학교 SW융합학과 조교수

〈관심분야〉 : 데이터베이스 시스템, 이동 객체 데이터베이스, 이동 P2P 네트워크, 소셜 네트워크 서비스, 빅데이터 등

유 재 수(Jaesoo Yoo) 종신회원

- 1989년 2월 : 전북대학교 컴퓨터공학과(공학사)
- 1991년 2월 : 한국과학기술원 전산학과(공학석사)
- 1995년 2월 : 한국과학기술원 전산학과(공학박사)
- 1995년 2월 ~ 1996년 8월 : 목포대학교 전산통계학과 전임강사
- 1996년 8월 ~ 현재 : 충북대학교 전자정보대학 정보수

〈관심분야〉 : 데이터베이스 시스템, 멀티미디어 데이터베이스, 센서 네트워크, 바이오인포매틱스, 빅데이터 등

부 록

종류	행 길이	불균형 비율	샘플링 기법	Precision	Recall	Error_rate	F score
ecoli	200	0.350	EditedNearestNeighbours	1.000	0.979	0.014	0.989
			RepeatedEditedNearestNeighbours	1.000	0.958	0.027	0.979
			ALLKNN	1.000	0.958	0.027	0.979
			TomekLinks	1.000	1.000	0.000	1.000
			Condensed Nearest Neighbour	1.000	0.979	0.014	0.989
			OneSideSelection	1.000	0.979	0.014	0.989
			NCR(Condensed Nearest Neighbour + Edited Nearest Neighbours)	1.000	0.958	0.027	0.979
			NearMiss-1	1.000	0.979	0.014	0.989
			NearMiss-2	1.000	0.979	0.014	0.989
			NearMiss-3	1.000	0.979	0.014	0.989
			RandomUnderSampling	1.000	0.958	0.027	0.979
			RandomOverSampling	1.000	0.979	0.014	0.989
			smote	1.000	0.979	0.014	0.989
			ADASTN(Adaptive Synthetic Sampling)	1.000	0.938	0.041	0.968
			Smote + Tlinks	1.000	0.979	0.014	0.989
			Smote + ENN	1.000	0.979	0.014	0.989
			Recommended Method	0.962	1.000	0.014	0.980
glass	214	0.327	EditedNearestNeighbours	0.774	0.857	0.155	0.814
			RepeatedEditedNearestNeighbours	0.658	0.893	0.225	0.758
			ALLKNN	0.658	0.893	0.225	0.758
			TomekLinks	0.778	0.750	0.183	0.764
			Condensed Nearest Neighbour	0.727	0.857	0.183	0.787
			OneSideSelection	0.786	0.786	0.169	0.786
			NCR(Condensed Nearest Neighbour + Edited Nearest Neighbours)	0.649	0.857	0.239	0.738
			NearMiss-1	0.561	0.821	0.324	0.667
			NearMiss-2	0.735	0.893	0.169	0.806
			NearMiss-3	0.667	0.857	0.225	0.750
			RandomUnderSampling	0.706	0.857	0.197	0.774
			RandomOverSampling	0.808	0.750	0.169	0.778
			smote	0.767	0.821	0.169	0.793
			ADASTN(Adaptive Synthetic Sampling)	0.733	0.786	0.197	0.759
			Smote + Tlinks	0.821	0.821	0.141	0.821
			Smote + ENN	0.750	0.857	0.169	0.800
			Recommended Method	0.667	0.857	0.225	0.750
haberman	306	0.265	EditedNearestNeighbours	0.474	0.290	0.317	0.360
			RepeatedEditedNearestNeighbours	0.449	0.710	0.356	0.550
			ALLKNN	0.449	0.710	0.356	0.550
			TomekLinks	0.467	0.226	0.317	0.304
			Condensed Nearest Neighbour	0.429	0.290	0.337	0.346
			OneSideSelection	0.500	0.226	0.307	0.311
			NCR(Condensed Nearest Neighbour + Edited Nearest Neighbours)	0.421	0.774	0.396	0.545
			NearMiss-1	0.489	0.710	0.317	0.579
			NearMiss-2	0.323	0.645	0.525	0.430
			NearMiss-3	0.224	0.355	0.574	0.275
			RandomUnderSampling	0.415	0.548	0.376	0.472
			RandomOverSampling	0.318	0.226	0.386	0.264
			smote	0.333	0.290	0.396	0.310
			ADASTN(Adaptive Synthetic Sampling)	0.400	0.452	0.376	0.424
			Smote + Tlinks	0.407	0.355	0.356	0.379
			Smote + ENN	0.390	0.516	0.396	0.444
			Recommended Method	0.556	0.645	0.267	0.597
iris	150	0.333	EditedNearestNeighbours	1.000	1.000	0.000	1.000
			RepeatedEditedNearestNeighbours	1.000	1.000	0.000	1.000

			ALLKNN	1.000	1.000	0.000	1.000
			TomekLinks	1.000	1.000	0.000	1.000
			Condensed Nearest Neighbour	0.440	1.000	0.560	0.611
			OneSideSelection	0.440	1.000	0.560	0.611
			NCR(Condensed Nearest Neighbour + Edited Nearest Neighbours)	1.000	1.000	0.000	1.000
			NearMiss-1	1.000	1.000	0.000	1.000
			NearMiss-2	1.000	1.000	0.000	1.000
			NearMiss-3	1.000	1.000	0.000	1.000
			RandomUnderSampling	1.000	1.000	0.000	1.000
			RandomOverSampling	1.000	1.000	0.000	1.000
			smote	1.000	1.000	0.000	1.000
			ADASTN(Adaptive Synthetic Sampling)	1.000	1.000	0.000	1.000
			Smote + Tlinks	1.000	1.000	0.000	1.000
			Smote + ENN	1.000	1.000	0.000	1.000
			Recommended Method	1.000	1.000	0.000	1.000
pages	5,472	0.102	EditedNearestNeighbours	0.802	0.932	0.027	0.862
			RepeatedEditedNearestNeighbours	0.727	0.925	0.038	0.814
			ALLKNN	0.727	0.925	0.038	0.814
			TomekLinks	0.853	0.901	0.023	0.876
			Condensed Nearest Neighbour	0.824	0.932	0.024	0.875
			OneSideSelection	0.853	0.901	0.023	0.876
			NCR(Condensed Nearest Neighbour + Edited Nearest Neighbours)	0.724	0.944	0.037	0.819
			NearMiss-1	0.110	0.938	0.681	0.197
			NearMiss-2	0.104	1.000	0.769	0.188
			NearMiss-3	0.863	0.901	0.022	0.881
			RandomUnderSampling	0.643	0.994	0.050	0.780
			RandomOverSampling	0.881	0.870	0.022	0.875
			smote	0.807	0.938	0.025	0.868
			ADASTN(Adaptive Synthetic Sampling)	0.786	0.938	0.028	0.856
			Smote + Tlinks	0.806	0.932	0.026	0.865
			Smote + ENN	0.731	0.963	0.035	0.831
			Recommended Method	0.714	0.975	0.037	0.824
pima	768	0.349	EditedNearestNeighbours	0.546	0.694	0.295	0.611
			RepeatedEditedNearestNeighbours	0.464	0.906	0.382	0.614
			ALLKNN	0.464	0.906	0.382	0.614
			TomekLinks	0.596	0.659	0.264	0.626
			Condensed Nearest Neighbour	0.523	0.812	0.311	0.636
			OneSideSelection	0.588	0.671	0.268	0.626
			NCR(Condensed Nearest Neighbour + Edited Nearest Neighbours)	0.448	0.859	0.402	0.589
			NearMiss-1	0.574	0.729	0.272	0.642
			NearMiss-2	0.409	0.847	0.461	0.552
			NearMiss-3	0.598	0.682	0.260	0.637
			RandomUnderSampling	0.562	0.800	0.276	0.660
			RandomOverSampling	0.588	0.671	0.268	0.626
			smote	0.584	0.694	0.268	0.634
			ADASTN(Adaptive Synthetic Sampling)	0.590	0.694	0.264	0.638
			Smote + Tlinks	0.604	0.682	0.256	0.641
			Smote + ENN	0.516	0.776	0.319	0.620
			Recommended Method	0.539	0.729	0.299	0.620
segment	2,308	0.143	EditedNearestNeighbours	1.000	0.982	0.003	0.991
			RepeatedEditedNearestNeighbours	1.000	0.982	0.003	0.991
			ALLKNN	1.000	0.982	0.003	0.991
			TomekLinks	1.000	0.973	0.004	0.986
			Condensed Nearest Neighbour	0.932	0.982	0.013	0.956
			OneSideSelection	0.991	0.982	0.004	0.986
			NCR(Condensed Nearest Neighbour + Edited Nearest Neighbours)	1.000	0.982	0.003	0.991
			NearMiss-1	0.838	0.982	0.030	0.905
			NearMiss-2	0.982	0.991	0.004	0.987

			NearMiss-3	0.908	0.982	0.017	0.944
			RandomUnderSampling	0.991	0.982	0.004	0.986
			RandomOverSampling	1.000	0.982	0.003	0.991
			smote	1.000	0.973	0.004	0.986
			ADASTN(Adaptive Synthetic Sampling)	0.991	0.982	0.004	0.986
			Smote + Tlinks	1.000	0.973	0.004	0.986
			Smote + ENN	0.973	0.982	0.007	0.978
			Recommended Method	0.965	0.982	0.008	0.973
vehicle	846	0.235	EditedNearestNeighbours	0.833	0.968	0.050	0.896
			RepeatedEditedNearestNeighbours	0.738	1.000	0.079	0.849
			ALLKNN	0.738	1.000	0.079	0.849
			TomekLinks	0.952	0.968	0.018	0.960
			Condensed Nearest Neighbour	0.822	0.968	0.054	0.889
			OneSideSelection	0.968	0.968	0.014	0.968
			NCR(Condensed Nearest Neighbour + Edited Nearest Neighbours)	0.795	1.000	0.057	0.886
			NearMiss-1	0.440	0.952	0.279	0.602
			NearMiss-2	0.423	0.968	0.300	0.588
			NearMiss-3	0.698	0.968	0.100	0.811
			RandomUnderSampling	0.803	0.984	0.057	0.884
			RandomOverSampling	0.937	0.952	0.025	0.944
			smote	0.923	0.968	0.025	0.945
			ADASTN(Adaptive Synthetic Sampling)	0.909	0.968	0.029	0.938
			Smote + Tlinks	0.923	0.968	0.025	0.945
			Smote + ENN	0.756	1.000	0.071	0.861
			Recommended Method	0.775	1.000	0.064	0.873
winsconsin	683	0.350	EditedNearestNeighbours	0.966	0.988	0.018	0.977
			RepeatedEditedNearestNeighbours	0.966	1.000	0.013	0.983
			ALLKNN	0.966	1.000	0.013	0.983
			TomekLinks	0.977	0.988	0.013	0.983
			Condensed Nearest Neighbour	0.933	0.977	0.035	0.955
			OneSideSelection	0.966	0.988	0.018	0.977
			NCR(Condensed Nearest Neighbour + Edited Nearest Neighbours)	0.956	1.000	0.018	0.977
			NearMiss-1	0.944	0.988	0.027	0.966
			NearMiss-2	0.988	0.977	0.013	0.982
			NearMiss-3	0.966	0.988	0.018	0.977
			RandomUnderSampling	0.966	1.000	0.013	0.983
			RandomOverSampling	0.977	1.000	0.009	0.989
			smote	0.956	1.000	0.018	0.977
			ADASTN(Adaptive Synthetic Sampling)	0.956	1.000	0.018	0.977
			Smote + Tlinks	0.956	1.000	0.018	0.977
			Smote + ENN	0.956	1.000	0.018	0.977
			Recommended Method	0.966	1.000	0.013	0.983
yeast	1,484	0.289	EditedNearestNeighbours	0.547	0.615	0.261	0.579
			RepeatedEditedNearestNeighbours	0.437	0.818	0.361	0.569
			ALLKNN	0.437	0.818	0.361	0.569
			TomekLinks	0.576	0.559	0.249	0.567
			Condensed Nearest Neighbour	0.476	0.692	0.312	0.564
			OneSideSelection	0.567	0.559	0.253	0.563
			NCR(Condensed Nearest Neighbour + Edited Nearest Neighbours)	0.452	0.762	0.339	0.568
			NearMiss-1	0.375	0.650	0.418	0.476
			NearMiss-2	0.342	0.664	0.471	0.451
			NearMiss-3	0.509	0.601	0.286	0.551
			RandomUnderSampling	0.488	0.685	0.302	0.570
			RandomOverSampling	0.550	0.462	0.267	0.502
			smote	0.538	0.545	0.269	0.542
			ADASTN(Adaptive Synthetic Sampling)	0.517	0.538	0.282	0.527
			Smote + Tlinks	0.534	0.545	0.271	0.540
			Smote + ENN	0.493	0.706	0.298	0.580
			Recommended Method	0.455	0.699	0.333	0.551