ISSN 2383-6318(Print) / ISSN 2383-6326(Online)
KIISE Transactions on Computing Practices, Vol. 24, No. 9, pp. 442-455, 2018. 9
https://doi.org/10.5626/KTCP.2018.24.9.442

DSMS 환경에서 이상 탐지를 위한 SVM과 리샘플링 기법의 분석

(Analysis of an SVM with Resampling Techniques for Anomaly Detection in a DSMS Environment)

김 동 효 † 구 해 모 † 김 형 주 ††
(Donghyo Kim) (Heymo Kou) (Hyoung-Joo Kim)

요 약 실시간 스트림 데이터가 연속적으로 들어오는 DSMS(Data Stream Management System) 환경에서 그 데이터들의 이상여부를 판단하는 아키텍처를 고안한다. DSMS는 전통적인 데이터베이스관리시스템보다 스트림 데이터를 처리하는데 최적화된 시스템이며, 일부 제품에서는 SQL 대신 CQL(Continuous Query Language)을 사용한다. 따라서 DSMS에서 이상탐지를 수행하기 위해서는 이상탐지 모델을 CQL로 DSMS에 등록해야 한다. 본 논문도 이러한 DSMS 환경에서의 이상탐지 상황을 상정하고, 이상탐지 모델을 CQL로 구현하려한다. CQL로의 구현을 고려하여 이상탐지를 위한 클래스 예측 알고리즘은 SVM(Support Vector Machine)을 사용한다. 그리고 본 실험에서는 SVM의 검증 성능을 높이기 위한 실험을 진행한다. 데이터집합의 클래스가 불균형할 때 발생할 수 있는 학습모델의 검증 성능 저하 문제를 리샘플링기법을 적용시켜 해결한다. 또한, 학습한 SVM모델의 임계값(threshold)을 조정하여 검증 성능을 최적화한다. 최종적으로 리샘플링된 데이터로 학습하고 임계값 조정된 SVM모델을 CQL로 변환하는 작업을 수행한다. 이 과정은 두 개의 자동화된 변환 블록을 거쳐서 수행하도록 구현한다.

키워드: 이상 탐지, 서포트 벡터 머신, 리샘플링, DSMS, CQL

Abstract In the DSMS (Data Stream Management System) environment, which receives real-time stream data continuously, we devised an architecture to judge whether the data is abnormal or not. DSMS is optimized for processing stream data rather than traditional DBMS, and some products use CQL (Continuous Query Language) instead of SQL. Therefore, an anomaly-detection model must be registered as a CQL in order to perform anomaly detection in the DSMS. This paper assumes an anomaly-detection situation in such a DSMS environment and implements the anomaly-detection model in CQL. Considering the implementation in CQL, we used an SVM (Support Vector Machine) as a class-prediction algorithm for anomaly detection. We performed experiments to improve the validation performance of the SVM. We solved the problem that validation performance of a learned model declines when the dataset is imbalanced, by applying resampling techniques. In addition, we adjusted the threshold of the learned SVM model to optimize the validation performance. Finally, we converted the threshold-tuned SVM model learned by resampled dataset to CQL. This process was implemented by means of two automated transformation blocks.

Keywords: anomaly detection, support vector machine, resampling, DSMS, CQL

· 이 논문은 2017년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임(No.R0113-15-0005, 대규모 트랜잭션 처리와 실시간 복합 분석을 통합한 일체형 데이터 엔지니어링 기술 개발)

† 비 회 원 : 서울대학교 컴퓨터공학과
dhkim@idb.snu.ac.kr
hmkou@idb.snu.ac.kr
†† 종신회원 : 서울대학교 컴퓨터공학과 교수(Seoul Nat'l Univ.)
hjk@snu.ac.kr
(Corresponding author임)

논문접수 : 2018년 2월 8일
(Received 8 February 2018)
논문수정 : 2018년 6월 25일
(Revised 25 June 2018)
심사완료 : 2018년 6월 26일
(Accepted 26 June 2018)

정보과학회 컴퓨팅의 실제 논문지 제24권 제9호(2018. 9)

1. 서 론

실시간성과 연속성을 갖는 스트림 데이터가 거래, 금융, 통신과 같은 다양한 분야에서 발생하고 있으며 이것을 처리하기 위한 필요성이 증대해 왔다.

DSMS(Data Stream Management System)는 이러한 스트림 데이터를 처리하기 위한 시스템이다. 연속적으로 들어오는 데이터에 대해 질의를 수행하는 기능과, 특정 스트림 데이터를 저장하기 위한 저장소를 갖춘 일체화된 시스템이다. 어플리케이션 분야에 따라 다양한 DSMS 제품이 존재하는데 대표적으로 STREAM, PipelineDB, AURORA, TelegraphCQ 등이 있다.

본 논문의 연구도 DSMS 환경에서 실시간으로 발생하는 스트림 데이터의 이상 탐지를 수행하는 상황을 상정한다. 이상 탐지를 위해서는 DSMS에 질의로 등록할 클래스 예측 알고리즘이 필요한데, 여러 가지 머신 러닝 기법 중 SVM(Support Vector Machine)을 사용한다.

본문의 2장에서는 DSMS의 개념과 일부 DSMS에서 사용하는 질의인 CQL(Continuous Query Language)의 개념을 다룬다. 그리고 이상 탐지 모델로 SVM을 선택한 이유를 CQL의 관점에서 설명한다. 또한 SVM의 클래스 예측 성능을 높이기 위해 본 실험에 사용할 다양한 리샘플링 기법들을 소개한다. 여러 가지 리샘플링 기법을 적용하는 이유는 이상 탐지 분야에 속한 데이터 집합(dataset)은 정상클래스에 비해 비정상클래스의 수가 매우 적은 클래스 불균형 문제가 존재하기 때문이다.

이러한 클래스 불균형한 데이터로 학습한 분류기(classifier)는 예측 성능이 좋지 않은 문제를 보일 수 있다.[1][2][3] 이러한 문제를 개선하기 위해 언더 샘플링(under-sampling), 오버 샘플링(over-sampling), 콤비네이션 샘플링(combination-sampling)과 같은 리샘플링 범주들이 제안되었다. 각 리샘플링 범주마다 다양한 알고리즘이 존재하고 특정 머신러닝 기법과 데이터집합에 대해 어떤 리샘플링 기법을 적용 했을 때의 검증 결과가 우세하다고 미리 판단하기는 어렵다[4,5].

따라서 3장에서는 서포트 벡터 머신(SVM)과 이상 탐지 분야의 데이터집합에 대해 가장 좋은 검증 성능을 보이는 리샘플링 기법은 무엇인지 찾는 실험을 수행한다. 또한 학습한 SVM모델의 임계값(threshold)을 조정함으로써 분류기의 예측 성능이 어떻게 변화하는지 파악하고, 이때의 가장 좋은 검증 결과를 보이는 리샘플링 기법을 찾는다. 4장에서는 3장에서 수행한 실험의 결과를 표와 그래프로 나타내고, 그 결과와 원인을 분석한다. 5장에서는 DSMS 환경에서의 이상 탐지를 위한 아키텍처를 제안하고, 3~4장에서 채택한 최상의 성능을 보인 리샘플링 기법을 적용하여 아키텍처의 SVM모델을 구성한다. 그리고 학습한 SVM모델을 DSMS에 등록하기 위해 CQL로 변환하는 구현과정을 다룬다.

2. 관련 연구

2.1 DSMS와 CQL

DSMS[6]는 데이터가 연속적으로 들어오고, 들어오는 각 데이터에 대해 특정 질의를 반복적으로 수행하는 상황에서 기존의 DBMS보다 성능이 최적화된 시스템이다. [6]에서 등장하는 유형의 DSMS는 CQL을 사용하는데, 질의를 등록해두면 새로운 데이터가 들어오는 순간에 질의를 수행한다. 그림 1은 DBMS와 DSMS에서 스트림 데이터를 처리하는 구조를 나타낸다. DBMS에서는 들어온 데이터에 대해 특정 질의를 수행하기 위해서, 일단 데이터베이스에 저장하고 다시 불러오는 과정을 거쳐야 한다. 한편, DSMS에서의 스트림 데이터는 미리 등록해둔 질의를 먼저 거치고 난 뒤, 데이터베이스에 저장되거나 버려진다. 따라서 질의를 위해 기존의 DBMS처럼 데이터를 저장하고 불러오는 과정이 없기 때문에 DSMS는 실시간 스트림 데이터를 다루는 데에 적합하다. 또한, 질의를 사전에 등록하기 때문에 일회성 질의를 반복적으로 생성하는 비효율성이 없고, 방대한 양의 스트림 데이터를 모두 저장할 필요가 없기 때문에 데이터베이스의 부담이 줄어든다.

본문 5장에서 제안하는 스트리밍 사기 탐지 아키텍처는 이러한 유형의 DSMS를 바탕으로 설계되었다. 실시간 스트림 데이터가 DSMS로 들어올 때, 각 데이터의 이상 여부를 판단하는 예측 모델을 CQL로 변환하여 DSMS에 등록함으로써 스트림 데이터에 대한 이상 탐지를 수행한다.

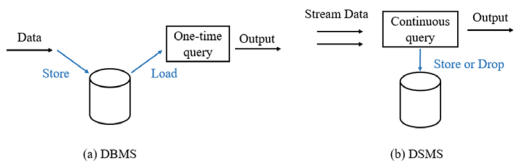

그림 1 DBMS와 DSMS의 스트림처리 구조의 차이
Fig. 1. Difference between the stream-processing structures of DBMS and DSMS

2.2 이상 탐지 목적의 SVM

SVM은 머신러닝의 지도학습 기법 중 하나로 분류(classification)문제에 적용할 수 있는 알고리즘이다[7]. 이상 탐지 분야는 정상과 비정상인 두 개의 클래스가 존재하는 분류 문제이므로 SVM을 사용하여 해결가능하다. 이상 탐지 분야에서 가장 큰 영역 중 하나인 네트워크 침입 탐지(network intrusion detection)에서 SVM

은 좋은 성능을 보인다[8-11] 이미지 분야에서의 이상 탐지도 잘 수행한다[12]. 신용카드 사기탐지(credit card fraud-detection)영역에서는 SVM을 사용하여 높은 TN (True Negative)비율을 얻을 수 있다[13]. 또한, 신용카드 사기탐지를 위한 비교실험에서 SVM은 인공신경망과 비슷한 수준의 좋은 성능을 보여준다[14]. 각 클래스별로 잘못 분류될 때의 페널티 가중치를 다르게 주는 방식의 연구도 존재한다[15].

클래스 분류를 위한 여러 가지 머신러닝 기법들 중에서 SVM을 선택한 이유는 학습한 모델을 DSMS에 등록하기 위해서 CQL로 변환하는 과정에 있다. SVM은 학습 결과로 계수(coefficients)와 절편(intercept)을 포함한 결정 경계(decision boundary) 모델을 생성한다. 이 모델의 클래스 예측함수는 하나의 부등식으로 구성되기 때문에 CQL로의 변환이 용이하다. 또한 마찬가지 이유로 스트림 데이터에 대해 클래스 예측(CQL 질의)을 빠르게 수행할 수 있다. 이러한 이유로 SVM을 이상 탐지 알고리즘으로 사용하였다.

2.3 리샘플링 기법

클래스 예측을 목적으로 하는 머신러닝 분류기를 구현할 때, 학습 데이터집합의 클래스가 불균형할 경우 분류기의 검증 성능이 좋지 않은 경우가 존재한다. 특히 이상 탐지와 같은 영역은 정상과 비정상 데이터의 클래스 불균형이 심하기 때문에 이러한 문제를 개선할 방법이 필요하다. 데이터집합의 불균형한 클래스 비율을 완화하기 위해 여러 가지 리샘플링 기법들이 제시되었다.

리샘플링 기법은 크게 세 가지 범주로 분류할 수 있는데, 그림 2는 각 범주에 대한 개념을 단순화 시켜 표현한 그림이다. 각 범주에 속하는 기법들은 클래스 간의 불균형한 비율을 개선하려는 공통점을 지니지만, 그 방법에 따라 차이가 존재한다. 언더샘플링 범주에 속한 기법은 다수 클래스에 속한 데이터를 제거하여 클래스 간 불균형을 완화한다. ENN[16], RENN[17], ALLKNN[17], TomekLinks[18], OneSidedSelection[2], NCR[19], Near-Miss-1,2[1]가 언더샘플링 기법에 해당한다.

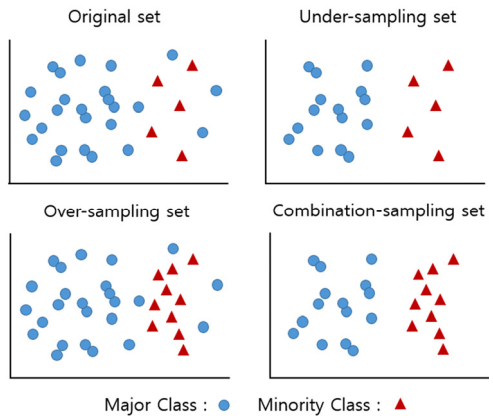

그림 2 언더 샘플링, 오버샘플링, 콤비네이션 샘플링의 개념을 추상화한 삽화

Fig. 2 Illustration abstracting the concepts of under-sampling, over-sampling, and combination sampling

오버샘플링 기법은 소수 클래스에 속하는 데이터를 새로 합성하여 추가한다. 즉 소수 클래스의 데이터를 증폭시켜 클래스 간의 비율을 조절하는 방법이다. 대표적으로 SMOTE[20], ADASYN[21] 기법이 존재한다.

콤비네이션 샘플링은 언더샘플링과 오버샘플링의 방식을 혼합한 형태이다. 소수 클래스 데이터는 증폭시키고 다수 클래스 데이터는 제거함으로써 클래스간의 비율을 조정한다. SMOTE+Tomek[22], SMOTE+ENN[3] 처럼 혼합된 방식이 제안되었다.

표 1은 각 범주에 속하는 리샘플링 기법들의 핵심적인 개념을 요약한 표이다. 논문 3장의 실험에서는 표 1의 기법들로 데이터를 리샘플링한다. 각 리샘플링 기법에 대한 자세한 내용은 부록에 기술하였다.

3. 리샘플링 및 SVM을 사용한 사기탐지 실험

3.1 실험 설계

연구에 사용한 데이터는 신용카드 거래 데이터[23]로 정상 또는 비정상(사기)으로 레이블링되어 있다. 이상 탐지 분야 중 하나인 사기 탐지(fraud detection) 영역에서 활용할 수 있는 데이터이다.

리샘플링된 신용카드 거래 데이터로 서포트 벡터 머신을 학습하고 그 결과를 검증하기 위해서, 전체 데이터를 80%의 트레이닝 데이터집합과 20%의 테스트 데이터집합으로 분할한다. 트레이닝 세트와 테스트 세트를 생성한 방식은 전체 데이터를 대상으로 임의 추출을 통해 분할하였다. 단, 트레이닝 세트와 테스트 세트 모두 정상 데이터와 사기 데이터의 비율이 원래 데이터에서의 0.0017 비율과 동일하도록 유지하며 생성하였다.

- Test set
  - class 0 (normal data) : 56863
  - class 1 (fraud data) : 98
- Original training set
  - class 0 (normal data) : 227452
  - class 1 (fraud data) : 394

실험 수행을 위해 먼저 관련연구 2.3에 나열된 각 샘플링 기법들을 원본 트레이닝 세트(original training set)에 적용시켜 각 기법별로 샘플링된 트레이닝 세트(sampled training set)를 생성한다.

3.1.1 (실험 1) 리샘플링 데이터집합으로 SVM 학습

각 리샘플링 기법을 적용시켜 생성한 샘플링된 트레

표 1 실험에 사용한 리샘플링 기법들의 범주와 개념요약
Table 1 Categories and concepts of resampling techniques used in the experiments

Algorithm	Category	Description
ENN	Under-sampling	K-NearestNeighbor을 이용해 다수 클래스의 데이터를 축소한다. 이웃한 데이터 중 자신과 같은 클래스보다 다른 클래스의 데이터 수가 더 많을 경우, 해당 데이터는 제외된다.
RENN	Under-sampling	리샘플링된 데이터집합의 변화가 없을 때 까지 ENN을 반복적으로 수행한다.
AllKNN	Under-sampling	ENN을 변형한 방식으로, k값을 설정하여 $1 \le i \le k$ 범위의 모든 i-NN을 수행한다. 각 수행 중 한번이라도 자신의 클래스와 판정이 다른 경우, 해당 데이터는 제외된다.
TomekLinks	Under-sampling	데이터 정리(cleaning) 기법으로, 거리를 기반으로 데이터 쌍의 집합(Tomek-links)을 정의한다. 이 집합에 속한 데이터들을 노이즈로 간주하여 제외시킨다.
OneSidedSelection	Under-sampling	TomekLinks를 수행한 뒤, 추가로 CondensedNearestNeighbor를 수행한다.
NCR	Under-sampling	ENN을 변형한 방식으로, 근처의 모든 이웃이 자신과 다른 클래스를 가질 경우에만 해당 데이터를 제외시킨다.
NearMiss-1	Under-sampling	다수 클래스의 각 데이터마다 반대 클래스에서 가장 가까운 데이터 3개를 뽑는다. 3개 데이터와의 평균 거리가 작은 순서대로 다수 클래스를 추출하여, 원하는 비율을 맞춘다.
NearMiss-2	Under-sampling	NearMiss-1에서 가장 먼 데이터를 3개 뽑도록 변형한 방식이다.
SMOTE	Over-sampling	소수 클래스에 속하는 데이터를 합성하여 데이터 수를 증폭시킨다. 합성된 데이터는 벡터 공간에서 기존의 소수 클래스 데이터들 사이에 위치하게 된다.
ADASYN	Over-sampling	SMOTE의 발전된 형태로, 데이터의 분포를 고려하여 클래스 간의 경계면에 가까울수록 더 많은 소수 클래스의 데이터를 합성한다.
SMOTE + Tomek	Combination-sampling	SMOTE를 수행한 뒤, TomekLinks를 사용한다.
SMOTE + ENN	Combination-sampling	SMOTE를 수행한 뒤, ENN을 사용한다.

이닝 세트로 서포트 벡터 머신을 학습시키고, 학습된 각 모델을 테스트 세트(test set)로 검증한다. 샘플링 기법들 중 클래스 비율을 파라미터로 조정할 수 있는 기법들은 다양한 클래스 비율로 모델을 만들어보고 그 검증 결과들을 비교해 본다. 그리고 어떤 리샘플링 기법을 사용한 데이터집합이 서포트 벡터 머신으로 학습했을 때 가장 좋은 성능을 보이는지 분석한다.

3.1.2 (실험 2) 학습한 SVM의 임계값 조정

샘플링된 데이터로 학습한 서포트 벡터 머신의 임계값을 조정함으로써 검증지표(validation metrics)가 어떻게 변화하는지 분석해보고 그 결과를 최적화한다. 학습한 SVM모델의 예측함수는 다음과 같다.

$$w_1x_1 + w_2x_2 + \cdots + w_nx_n + intercept > threshold$$

x는 입력 데이터의 벡터 값, w는 SVM을 학습함으로써 얻어진 계수, $intercept$는 결정경계의 상수 값이다. 좌변의 값이 $threshold$보다 크면 해당 데이터의 클래스를 사기로 예측한다. $threshold$의 디폴트값은 0인데, 이 값을 조정함으로써 검증지표의 성능을 향상시킨다.

3.2 서포트 벡터 머신(SVM)의 구현방식

연구를 진행하기에 앞서 서포트 벡터 머신의 어떠한 형태를 사용할 지를 결정할 필요가 있다. 커널 함수(kernel function)를 사용하여 구현하는 SVC(Support Vector Machine classifier)의 커널 함수에는 일반적으로 선형함수(Linear function), 다항함수(Polynomial function), 방사기저 함수(Radial Basis Function), 시그모이드 함수(Sigmoid function) 등이 쓰인다. 한편 커널을 선형함수(Linear function)로 사용한 SVC의 다른 구현으로, 커널 함수를 사용하지 않고 구현한 LinearSVC 방식도 존재한다.

데이터에 따라서 어떤 구현 또는 어떤 커널 함수를 사용했는지에 따라 모델의 성능과 학습 시간에 크게 달라 질 수 있다. 진행된 연구에서는 SVC를 많은 횟수로 생성해야하기 때문에 어떤 구현을 선택할지에 대한 판단이 필요하였다.

표 2는 3.1에서 설명한 신용카드 거래 데이터를 정규화 한 뒤 임의로 데이터를 추출해서 서포트 벡터 머신을 수행했을 때 걸리는 시간을 각 구현별로 측정한 결과이다. 데이터 개수가 1만 이하에서는 모두 빠르게 수

표 2 신용카드 데이터에서 임의 추출한 데이터 개수에 따른 서포트벡터머신의 학습시간
Table 2 Learning time of SVM depending on the amount of data randomly extracted from a credit-card dataset

samples	1,000	10,000	50,000	100,000	200,000
LinearSVC	0.032 (sec)	0.529	6.36	19.70	46.60
SVC kernel: Linear	0.019 (sec)	1.185	88.06	124.78	1040.72
SVC kernel: Polynomial	0.011 (sec)	0.276	3.16	7.79	49.18
SVC kernel: Radial basis function	0.055 (sec)	0.673	11.84	121.47	134.88
SVC kernel: Sigmoid	0.012 (sec)	0.275	1.82	4.42	9.90

행되지만, 데이터의 개수가 커질수록 수행 시간에 큰 차이가 발생한다. 커널 함수를 사용하는 SVC의 경우 커널 함수가 선형 함수, 방사기저 함수일 때는 수행 시간이 너무 길어져서 사용하기 부적합했다. 한편 커널 함수가 다항 함수, 시그모이드 함수인 경우에는 학습 시간은 짧지만, 검증결과가 매우 좋지 않았다.

따라서 학습시간이 빠르면서 검증 결과도 상대적으로 좋은 LinearSVC를 서포트 벡터 머신의 구현으로 선택하였다. LinearSVC는 커널 함수가 선형 함수인 SVC와 유사하지만, 데이터의 개수가 커져도 빠르게 수렴하기 때문에 학습시간이 적다는 장점을 지닌다. 신용카드 거래 데이터는 약 28만개의 데이터를 가지고 있기 때문에 데이터 개수에 대해 확장성(scalable)있고 검증 성능도 타당한 LinearSVC를 선택하였다.

3.3 데이터 설명 (Data description)

신용카드 거래 데이터[23]는 비식별화된 데이터집합이며 각 데이터 행은 신용 카드 거래 데이터를 나타낸다. 총 284,807개의 데이터 행(row)이 존재하며, 각 데이터는 30개의 비식별화된 속성(column)들을 가지고 있다. 그리고 데이터가 정상인 경우는 0, 사기 데이터인 경우는 1의 값을 갖는 클래스 속성이 존재한다. 총 284,807개의 데이터 중에 사기 데이터는 492개, 정상데이터는 284,315개이며 클래스 비율은 약 0.0017로 매우 불균형한 상태이다.

3.4 검증지표 (Validation metrics)

대부분의 이진 분류 문제(두 개의 클래스를 판별하는 문제)에서 분류기의 성능을 평가하는데 혼동 행렬(confusion matrix)을 사용한다. 혼동 행렬은 표 3과 같이 TP, FP, FN, TN의 네 가지 요소로 구성된다. 각 요소는 '양성(Positive)으로 예측해서 맞은 개수', '양성으로 예측해서 틀린 개수', '음성(Negative)으로 예측해서 틀린 개수', '음성으로 예측해서 맞은 개수'를 의미한다. 이 네 가지 요소를 사용해 도출 가능한 검증지표에는 여러 가지가 있다. 그 중 정확도, 재현율, 정밀도, F1 스코어를 채택하여 실험 결과의 검증지표로 사용하였다.

표 3 혼동 행렬
Table 3 Confusion Matrix

		predicted condition	
	total population	prediction positive	prediction negative
true condition	condition positive	**True Positive (TP)**	**False Negative (FN)**
	condition negative	**False Positive (FP)**	**True Negative (TN)**

• 정확도 (Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$) : 모든 클래스에 대한 예측 중 얼마나 그 예측이 맞았는지 비율을 나타낸다. 즉 분류기의 모든 클래스에 대한 예측 정확도를 나타낸다. 그런데 클래스가 매우 불균형하고 관심 있는 클래스가 극소수인 경우에는 이 측도만으로 분류기의 성능을 판단하는데 한계가 있다. 예를 들어 사기 탐지 문제에서 양성을 사기로 예측한다고 할 때, 대다수의 데이터는 정상이기 때문에 TN이 TP보다 압도적으로 커진다. 이런 경우 정확도는 매우 높더라도 정작 관심 있는 사기 데이터(TP)는 잘 잡아내는지 판단하기 어렵다. 따라서 재현율, 정밀도 같은 추가적인 지표들이 필요하다.

• 재현율 (Recall = $\frac{TP}{TP+FN}$) : 데이터집합의 모든 양성 클래스에 대해, 분류기가 양성 클래스로 예측한 비율을 나타낸다. 예를 들자면 데이터집합의 모든 사기 데이터 중 분류기가 얼마나 많은 사기 데이터를 걸러냈는지의 비율을 의미한다. 만약 모든 사기 데이터를 빠짐없이 필터링하는 것이 목적이라면, 이 측도를 높이는 것이 중요하다.

• 정밀도 (Precision = $\frac{TP}{TP+FP}$) : 분류기가 양성 클래스라고 예측한 데이터들에 대해, 실제로 양성인 데이터들의 비율을 나타낸다. 예를 들자면 분류기가 사기 데이터라고 예측한 데이터들 중 실제로 사기 데이터인 비율을 의미한다. 만약 분류기가 사기 데이터라고 예측했을 때의 정확성이 중요하거나, 사기 데이터라고 예측했는데 아닌 경우의 처리 비용이 크다면 이 측도를 높여야 한다.

• F1 스코어 (F1 score = $\frac{2TP}{2TP+FP+FN}$) : 재현율과 정밀도의 조화평균이다. 생성된 분류기를 재현율과 정밀도를 모두 반영한 하나의 수치로 표현하기 위해 이 측도를 사용한다.

4. 실험결과 및 분석

4.1 실험 환경

서포트 벡터 머신 및 샘플링 기법들은 모두 파이썬(v2.7)환경에서 구현되었다. 실험에 사용한 기기의 사양은 Intel®Xeon® CPU E5-2620 v4 @ 2.10GHz×2 , 8×2 cores, 80GB DDR4 RAM 2133MHz 이고, OS 환경은 Ubuntu 16.04.2 LTS이다. 외부 라이브러리로 imbalanced-learn[24]을 활용하였다.

4.2 리샘플링 데이터집합으로 학습한 SVM 검증 결과

표 4,5,6,7은 3.1.1 실험1을 수행하여 나온 결과들이다.

표 4의 ENN, RENN, ALLKNN, NCR, OneSidedSelection, TomekLinks는 클래스 비율을 임의로 설정할 수 없는 언더샘플링 기법이다. 반면에 NearMiss-1, NearMiss-2는 클래스 비율을 임의로 조정할 수 있는 언더샘플링 기법이다. 비율을 조정 가능한 기법에 대해서는 여러 가지 클래스 비율로 실험을 수행하였다. 비교 목적으로 표 4의 최상단 행은 샘플링 기법을 적용하지 않은 데이터집합으로 학습한 서포트 벡터 머신의 검증 결과를 넣었다. C.R은 소수의 클래스 데이터 수를 다수의 클래스 데이터 수로 나눈 클래스비율(class ratio)을 의미한다.

클래스 비율을 임의로 설정할 수 없는 언더샘플링 기법 중에서, TomeLinks를 사용했을 때의 정밀도가 0.9152로 가장 높았다. 재현율은 0.5510으로 상대적으로 낮지만, 전체 사기 데이터를 필터링하는 것보다 사기 예측의 정확도가 중요한 시스템이라면 TomkeLinks를 사용하는 것이 유리하다. 한편, 표 4에서 가장 높은 재현율과 F1 스코어를 보인 것은 RENN이다. 즉 언더샘플링 기법 중에서는 재현율과 정밀도를 종합적으로 고려했을 때 RENN의 검증 성능이 가장 좋았다.

표 4의 클래스 비율을 설정할 수 있는 기법 중에서, NearMiss-1은 매우 큰 FP 때문에 정밀도가 굉장히 낮게 측정된다. 클래스 비율을 낮출수록 FP가 감소하고 있지만 정밀도의 향상에 큰 기여를 하지 못하고 있다. 즉 NearMiss-1은 사기 탐지 영역에서 서포트 벡터 머신과 조합하기에 적절하지 않아 보인다. NearMiss-2는 NearMiss-1에 비해 좋은 성능을 보여준다. 클래스 비율이 1/10 일 때는 FP가 꽤 크지만, 클래스 비율을 감소시키면 FP가 급격히 줄어들어 정밀도가 향상된다. 클래스 비율이 1/130 일 때 F1 스코어가 최대가 되고 그 이후에는 다시 감소한다. 이것은 F1 스코어가 최대가 되는 어떤 클래스 비율의 지점이 존재함을 알 수 있다.

표 5의 ADASYN, SMOTE는 클래스 비율을 임의로 설정할 수 있는 오버 샘플링 기법이다. ADASYN는 클래스 비율이 1/10일 때는 정밀도가 낮은데, 클래스 비율이 감소할수록 재현율의 하락폭은 미미한 반면에 정밀도의 상승폭은 커진다. 클래스 비율이 감소할수록 F1 스코어가 점점 증가하다가 클래스 비율이 1/70일 때 최대가 되고, 그 이후로는 다시 감소한다.

SMOTE는 클래스 비율에 따른 F1 스코어의 변화가 상대적으로 적다. F1 스코어가 최대가 되는 것은 클래스 비율이 1/25 부근일 때이고, 그 이하의 비율에 대해서는 F1 스코어가 다시 감소하고 있다. 오버 샘플링 기법이므로 클래스 비율이 높을수록 더 많은 소수의 클래스 데이터를 생성하는데, 클래스 비율이 어느 적정수준보다 높아지면 오히려 검증 결과가 나빠진다는 것을 보여준다. 즉 과도하게 소수의 클래스를 오버 샘플링하면

표 4 언더 샘플링한 데이터집합으로 학습한 서포트벡터 머신의 검증 결과

Table 4 Validation results of SVM learned by under-sampling dataset

	C.R	Recall	Precision	F1 score	Accuracy
No sampling (Original dataset)	N/A	0.7142	0.8750	0.7865	0.9993

	C.R	Recall	Precision	F1 score	Accuracy
Edited NearestNeighbours	N/A	0.7142	0.8974	0.7954	0.9993
RepeatedEdited NearestNeighbours	**N/A**	**0.7346**	**0.8888**	**0.8044**	**0.9993**
AllKNN	N/A	0.7244	0.8875	0.7977	0.9993
Neighbourhood CleaningRule	N/A	0.7346	0.8780	0.8000	0.9993
OneSidedSelection	N/A	0.7346	0.8780	0.8000	0.9993
TomekLinks	N/A	0.5510	0.9152	0.6878	0.9991

	C.R	Recall	Precision	F1 score	Accuracy
NearMiss-1	1/10	0.8673	0.0057	0.0113	0.7397
	1/20	0.8367	0.0064	0.0127	0.7774
	1/30	0.8265	0.0068	0.0135	0.7923

	C.R	Recall	Precision	F1 score	Accuracy
NearMiss-2	1/10	0.8163	0.0497	0.0937	0.9728
	1/40	0.7346	0.6315	0.6792	0.9988
	1/70	0.7142	0.7446	0.7291	0.9990
	1/100	0.7142	0.8433	0.7738	0.9992
	1/130	0.7653	0.8241	0.7936	0.9993
	1/160	0.7040	0.8846	0.7840	0.9993
	1/200	0.7040	0.8734	0.7796	0.9993

표 5 오버 샘플링한 데이터집합으로 학습한 서포트벡터머신의 검증 결과

Table 5 Validation results of SVM learned by over-sampling dataset

	C.R	Recall	Precision	F1 score	Accuracy
ADASYN	1/10	0.8469	0.1836	0.3018	0.9932
	1/20	0.8163	0.4705	0.5970	0.9981
	1/30	0.8061	0.5895	0.6801	0.9987
	1/40	0.7959	0.6666	0.7255	0.9989
	1/50	0.7857	0.7264	0.7549	0.9991
	1/60	0.7857	0.7333	0.7586	0.9991
	1/70	0.7755	0.7835	0.7794	0.9992
	1/80	0.7755	0.7755	0.7755	0.9992
	1/90	0.8061	0.7181	0.7596	0.9991

	C.R	Recall	Precision	F1 score	Accuracy
SMOTE	1/10	0.8163	0.6896	0.7476	0.9990
	1/20	0.7959	0.7878	0.7918	0.9992
	1/25	**0.8061**	**0.7900**	**0.7979**	**0.9992**
	1/30	0.7857	0.7857	0.7857	0.9992

표 6 콤비네이션 샘플링한 데이터집합으로 학습한 서포트벡터머신의 검증 결과

Table 6 Validation results of SVM learned by combination-sampling dataset

	C.R	Recall	Precision	F1 score	Accuracy
SMOTETomek	1/10	0.8061	0.6869	0.7417	0.9990
	1/20	0.7959	0.7800	0.7878	0.9992
	1/25	0.7959	0.7878	0.7918	0.9992
	1/30	0.7857	0.7857	0.7857	0.9992
	1/35	0.7857	0.7857	0.7857	0.9992
	1/40	0.7857	0.7857	0.7857	0.9992

	C.R	Recall	Precision	F1 score	Accuracy
SMOTEENN	1/10	0.8163	0.6956	0.7511	0.9990
	1/15	0.8061	0.7669	0.7860	0.9992
	1/23	**0.7755**	**0.8539**	**0.8128**	**0.9993**
	1/30	0.7959	0.7878	0.7918	0.9992
	1/35	0.7857	0.7857	0.7857	0.9992
	1/40	0.7857	0.7857	0.7857	0.9992

표 7 여섯 가지의 리샘플링한 데이터집합으로 학습한 서포트벡터머신의 검증 결과 비교

Table 7 Comparison of validation results of SVM learned by six resampling techniques

	C.R	Recall	Precision	F1 score	Accuracy	Rank (F1)
SMOTEENN	**1/23**	**0.7755**	**0.8539**	**0.8128**	**0.9993**	**1**
RepeatedEdited NearestNeighbours	N/A	0.7346	0.8888	0.8044	0.9993	2
SMOTE	1/25	0.8061	0.7900	0.7979	0.9992	3
Nearmiss-2	1/30	0.7653	0.8241	0.7936	0.9993	4
SMOTETomek	1/25	0.7959	0.7878	0.7918	0.9992	5
ADASYN	1/70	0.7755	0.7835	0.7794	0.9992	6

서포트 벡터 머신의 성능을 오히려 나쁘게 만든다.

표 6에서 SMOTEENN은 클래스비율이 1/23일 때 가장 높은 F1 스코어를 갖는데, 모든 기법을 통틀어서 가장 높은 값을 도출하였다. SMOTETomek는 클래스 비율이 1/25일 때 가장 높은 F1 스코어를 가졌다. SMOTE 계열의 샘플링 기법들은(SMOTE, SMOTEENN, SMOTETomek) 공통적으로 클래스 비율이 1/23 ~ 1/25인 부근에서 가장 높은 F1 스코어를 가진다. 또한, 다른 샘플링 기법들에 비해 클래스 비율 변화에 의한 검증지표의 변화가 안정적이다.

표 7에서는 지금까지의 실험 결과 중 언더 샘플링, 오버 샘플링, 콤비네이션 샘플링 각각에 대해 최적의 클래스 비율에 대한 검증지표를 비교하였다. 클래스 비율을 임의로 설정할 수 없는 언더샘플링 기법 중에서는 가장 성능이 좋은 RENN을 대표로 선정하였다. NearMiss-1은 검증 결과가 매우 좋지 않았기 때문에 제외하였다.

총 여섯 가지 기법을 F1 스코어 기준으로 순위대로 나열하였고, 클래스 비율이 1/23인 SMOTEENN이 가장 높은 성능을 보여주었다. 4.2에서는 이 여섯 가지 기법의 리샘플링 된 데이터로 학습한 SVM모델의 임계값을 변화시킴으로써, 검증지표가 어떻게 변화하고 F1 스코어의 순위는 어떠한 변동이 있는지 실험한다.

4.3 임계값을 조정한 SVM 검증 결과

그림 3은 3.1.2 실험2를 수행한 결과로, 표 7의 각 기법에 대해 SVM모델의 임계값을 -0.5에서 +0.5까지 조정했을 때의 검증 결과를 나타낸 것이다. 각 샘플링을 사용했을 때의 그래프를 보면 RENN은 임계값<0 범위에서 F1 스코어가 최대인 지점이 존재했다. 임계값이 0보다 커지면 정밀도의 증가보다 재현율의 감소폭이 더 크기 때문에 F1 스코어가 감소했다. 따라서 임계값을 음수로 할당해 결정 경계(decision boundary)를 내림으로써 FN을 줄이고 F1 스코어를 향상시킬 수 있다.

Nearmiss-2는 임계값이 0인 부근에서 F1 스코어가 최대인 지점이 존재했다. 임계값>0 범위에서는 재현율의 급락으로 F1 스코어가 하락하고, 임계값<0 범위에서도 F1 스코어의 개선이 나타나지 않는다. 즉 초기 임계값에서 부근에서 최적의 F1 스코어가 도출된다.

ADASYN, SMOTE 계열 기법들은 임계값>0 범위에서 F1 스코어가 최대인 지점이 존재한다. 임계값<0 범위에서는 정밀도의 급락으로 F1 스코어가 모두 하락한다. 이 기법들은 비슷한 그래프의 경향성을 가지며, F1 스코어가 최대가 되는 임계값 또한 0.44~0.48로 모두 유사한 특징을 보였다.

4.4 실험 결과 분석

• 임계값(threshold) 조정 결과

표 8은 4.2의 실험에서 각 기법마다 최대의 F1 스코어를 가질 때의 임계값을 선택하고, 그 검증 결과들을 비교한 표이다. 표 7과 비교했을 때, 각 기법의 F1 스코어의 순위 변동이 있었다. 초기 임계값(=0)에서는 1위였던 RENN이 6위로 떨어진 반면, 오버 샘플링과 콤비네이션 샘플링 기법들이 모두 1~4위를 차지하였다.

특히 SMOTE 계열의 알고리즘들이 1~3위를 차지하여 좋은 성능을 보였다. 또한 F1 스코어가 최대일 때의 클래스 비율과 임계값이 유사하며, 임계값 조정에 따른 검증지표의 그래프 경향성도 거의 일치한다.

SMOTE 계열의 기법들이 서포트 벡터 머신과 조합했을 때, 가장 좋은 결과를 보인 이유는 안정적인 서포트 벡터의 증폭에서 찾을 수 있다. 서포트 벡터는 서포트 벡터 머신의 결정 경계를 생성하는 핵심적인 데이터

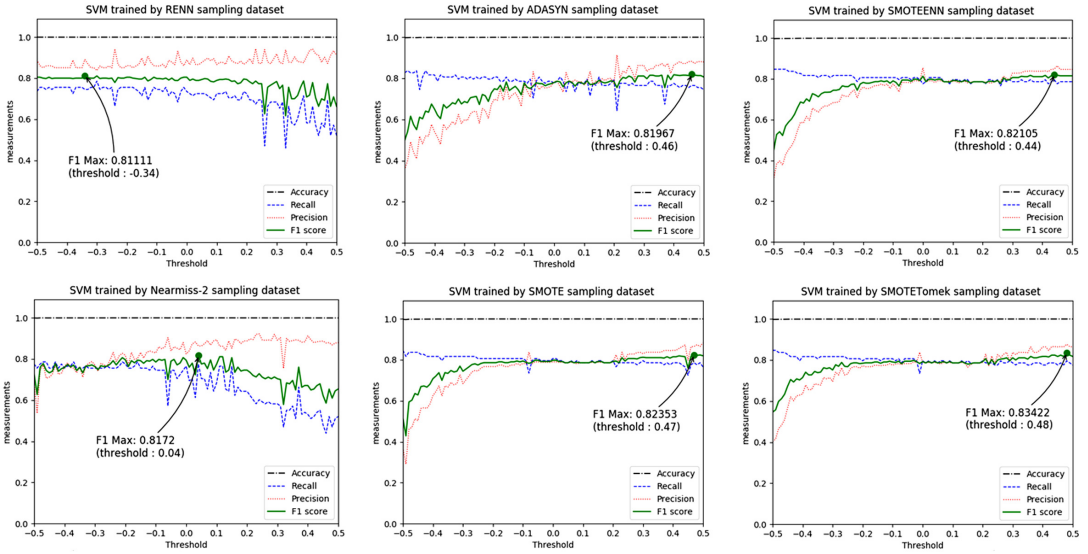

그림 3 여섯 가지의 리샘플링 데이터집합으로 학습한 서포트벡터머신 모델의 임계값 변화에 따른 검증 결과 비교

Fig. 3 Comparison of validation results depending on the threshold change of the SVM model learned by each of six resampling datasets

표 8 여섯 가지의 리샘플링 기법별 최적 임계값에서의 검증 결과 비교

Table 8 Comparison of validation results at optimal threshold for each of six resampling techniques

	C.R	Recall	Precision	F1 score	Accuracy	Rank (F1)	Prior Rank
SMOTETomek threshold : 0.48	**1/25**	**0.7959**	**0.8764**	**0.8342**	**0.9994**	**1**	**5**
SMOTE threshold : 0.47	1/25	0.7857	0.8651	0.8235	0.9994	2	3
SMOTEENN threshold : 0.44	1/23	0.7959	0.8478	0.8210	0.9994	3	1
ADASYN threshold : 0.46	1/70	0.7653	0.8823	0.8196	0.9994	4	6
Nearmiss-2 threshold : 0.04	1/130	0.7755	0.8636	0.8172	0.9994	5	4
RepeatedENN threshold : -0.34	N/A	0.7448	0.8902	0.8111	0.9994	6	2

들이다. SMOTE 알고리즘의 특성상 소수 클래스에 속한 데이터의 모든 분포에서 균일하게 새로운 샘플들이 생겨나고, 생성된 샘플들은 기존 데이터들 사이에 위치하기 때문에 원래의 군집을 크게 벗어나지 않는다. 즉 서포트 벡터가 될 가능성이 있는 경계면에 가까운 데이터들이 과도하게 증폭되지 않으며, 경계면에서 새로 생성된 샘플들은 기존의 서포트 벡터들과 크게 벗어나지 않는다. 이러한 특성이 안정적인 서포트 벡터의 증폭을 가능하게 하여 서포트 벡터 머신의 성능을 향상시키는 주요한 원인으로 추정된다.

• 정밀도-재현율의 상호교환(trade-off)

그림 3의 각 그래프를 살펴보면 임계값의 모든 범위에서 정밀도와 재현율의 상호교환이 존재한다, 샘플링 기법에 따라 임계값의 특정 구간에서 상호교환의 변동이 심한 특징을 나타낸다. 대다수의 경우 정밀도-재현율의 변동이 미미한 구간이 존재하다가, 둘 중 하나의 상승폭이 다른 것의 하락폭보다 큰 지점에서 F1 스코어가 최대가 된다.

• 리샘플링 데이터로 학습했을 때의 서포트벡터 수

각 리샘플링 데이터로 학습했을 때 발생하는 성능 차이를 설명하기 위해 서포트벡터 수를 측정하였다. 앞서 언급하였듯이, 서포트 벡터는 SVM의 결정 경계를 생성하는 핵심적인 데이터이다. 따라서 서포트 벡터 수가 많을수록 데이터에 보다 더 적합(fitting)한 결정경계를 생성할 가능성이 크다. 또한, 이분류 문제에서 서포트 벡터는 두 가지 클래스 중 하나에 속한다. 만약 서포트 벡터가 특정 클래스에 치우치게 분포한다면, 상대적으로 다른 클래스의 속한 서포트벡터 수는 적어지고, 이것은 결정경계를 생성하는데 한 쪽 클래스에 대한 정보가 덜 반영됨을 의미한다. 따라서 서포트벡터 수와 서포트벡터가 속한 클래스간의 비율을 살펴보는 것은 성능 차이를 설명하는데 유용한 방법이 된다.

표 9는 각 리샘플링된 데이터로 학습했을 때의 서포트벡터 수를 나타낸 표이다. 실험에는 linear한 SVM이 사용되었지만, 리샘플링 기법이 여러 종류의 커널에 유사한 영향을 주는지 확인하기 위해 linear, poly(polynomial), rbf(radial basis function) 커널에 대해 각각 측정하였다. 사기 클래스에 속한 서포트벡터 수, 정상 클래스에 속한 서포트벡터 수, 총 서포트벡터 수, 정상

표 9 리샘플링 데이터로 학습했을 때의 서포트벡터 수 및 서포트벡터가 속한 클래스 간 비율의 변화

Table 9 Change in the number of support vectors and ratio between classes when learned by resampling datasets

	Kernel	Number of support vectors in *Fraud class*	Number of support vectors in *Normal class*	Total number of support vectors	Ratio
Original Dataset	linear	189	1247	1436	0.15
	poly	130	516	646	0.25
	rbf	303	2631	2934	0.12
RENN	linear	147	1048	1195	0.14
	poly	97	506	603	0.19
	rbf	279	2645	2924	0.11
Nearmiss-2	linear	188	245	433	0.77
	poly	125	392	517	0.32
	rbf	294	1534	1828	0.19
ADASYN	linear	2210	4616	6826	0.48
	poly	953	1249	2202	0.76
	rbf	1467	2879	4346	0.51
SMOTE	linear	1896	3983	5879	0.48
	poly	1159	1557	2716	0.74
	rbf	1532	2832	4364	0.54
SMOTETomek	linear	1891	3977	5868	0.48
	poly	1161	1557	2718	0.75
	rbf	1536	2833	4369	0.54
SMOTEENN	linear	1882	4365	6247	0.43
	poly	1144	1538	2682	0.74
	rbf	1519	2828	4347	0.54

클래스에 대한 사기 클래스인 서포트벡터 수의 비율을 차례로 표기하였다.

최상단의 Original Dataset은 리샘플링 기법을 적용하지 않고 학습했을 때의 결과이다. 언더샘플링 기법 중 RENN은 큰 변화가 없는 반면에, Nearmiss-2는 정상 클래스에 속한 서포트벡터 수가 현저히 감소한다. 이로 인하여 linear커널의 경우 서포트벡터의 클래스 비율이 0.77까지 높아지고, 이것은 결정 경계를 형성하는데 양쪽 클래스의 정보가 균형 있게 반영될 수 있음을 의미한다. 즉, 전체 서포트벡터 수는 감소하고 서포트벡터의 클래스 비율은 높아지고 있으므로 Nearmiss-2는 좀 더 일반화(generalization)된 분류기를 만드는데 기여할 것으로 추측할 수 있다.

한편, 오버샘플링과 콤비네이션샘플링 기법에 속하는 ADASYN와 SMOTE계열 기법들은 전반적으로 서포트벡터 수를 증가시킨다. linear커널의 경우 사기 클래스에 속한 서포트벡터 수는 약 10배 증가하고, 정상 클래스에 속한 서포트벡터 수도 약 3배 이상 증가하였다. 사기 클래스에 속한 서포트벡터 수의 증가폭이 더 크기 때문에, 클래스 비율 또한 3배 정도 상승하였다. 즉, 각 클래스의 서포트벡터 수가 증가하고 서포트벡터의 클래스 비율도 높아지므로, ADASYN와 SMOTE계열 기법들은 주어진 데이터에 대해 좀 더 적합한 분류기를 만드는데 기여할 것으로 추정된다.

• 리샘플링 선택 전략

실험결과를 종합해보면 SMOTE 계열의 알고리즘들이(SMOTE, SMOTETomek, SMOTEENN) 사기 탐지 도메인에서 서포트 벡터 머신과의 궁합이 잘 맞았다. 샘플링 카테고리 측면에서는 콤비네이션 샘플링, 오버샘플링, 언더 샘플링 순으로 서포트 벡터 머신의 성능에 기여하였다.

높은 재현율과 F1 스코어를 원한다면 SMOTE 계열의 샘플링 기법을 적용하는 것이 유리하다. 만약 재현율의 성능보다 정밀도가 중요한 어플리케이션이라면, F1 스코어는 희생하더라도 정밀도가 높은 RENN을 선택하는 전략도 유효하다.

실험결과 SMOTE 계열의 샘플링 기법들은 유사성이 존재하기 때문에, 어느 하나의 기법에 대해 최적의 클래스 비율과 임계값을 찾았다면, 그 파라미터 값을 다른 SMOTE 계열 기법에 적용시켜볼 수 있다. 즉, 클래스 비율과 임계값을 탐색하는 범위를 축소시키고 SMOTE 계열 기법간의 검증 결과들을 빠르게 비교해 볼 수 있다.

5. 스트리밍 사기 탐지 아키텍쳐 (Streaming fraud detection architecture)

5.1 DSMS와 CQL 환경에서의 사기 탐지

스트림 환경에서 실시간으로 신용카드 거래 데이터가 들어온다고 할 때, 이 데이터가 정상인지 사기인지 짧은 시간 내에 판단할 수 있어야 한다. 실시간 스트림 환경에서는 많은 양의 데이터가 지속적이고 빠르게 들어오기 때문에 각 데이터의 클래스를 예측하는 분류기의 처리시간이 빨라야 한다.

전통적인 데이터베이스시스템으로 이러한 사기 탐지 시스템을 구현한다고 가정한다면, 분류기의 예측 함수가 SQL로 구현될 것이다. 인풋으로 들어온 데이터는 일단 데이터베이스에 저장되어야 하고, SQL을 수행할 때 그 데이터를 다시 불러와야 한다. 또 SQL은 1회성 질의이므로 각 데이터에 대해 매번 호출해야 한다. 즉 각 데이터를 분류하는 과정에서 데이터를 저장하고, 다시 불러오고, 1회성 질의를 매번 생성하는 비효율적 상황이 발생한다.

한편 DSMS으로 사기 탐지 시스템을 구현하면, 이러한 비효율적인 상황을 개선할 수 있다. DSMS는 스트림 데이터가 들어오는 동시에 CQL로 질의를 수행한다. 따라서 DSMS의 CQL을 사용하면 전통적인 DBMS처럼 SQL을 수행하기 위해 데이터를 저장하고 불러오는 과정이 생략된다. 또한, 질의를 시스템에 미리 등록하여 사용하므로, 동일한 1회성 질의를 반복적으로 생성하는 비효율이 발생하지 않는다.

5.2 모델 설계

그림 4는 DSMS 환경에서의 사기 탐지 시스템의 구조를 도식화하여 나타내고 있다. 이상 탐지 분야에 사기 탐지가 포함되므로, 사용하는 데이터집합에 따라서 이상

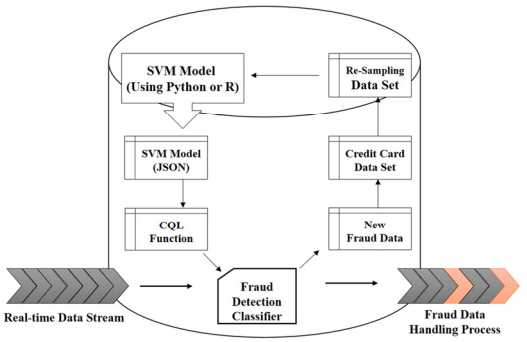

그림 4 DSMS 환경에서의 사기 탐지 아키텍처 모델
Fig. 4 Fraud-detection architecture model in DSMS environment

탐지 아키텍처 모델로 사용할 수 있다. 전체 프로세스는 하나의 일체형 시스템 내에서 수행된다. 이 일체형 시스템은 CQL을 지원하는 DSMS 또는 그러한 기능을 탑재 가능한 DBMS로 가정한다.

사기 탐지 분류기를 만들기 위해, 해당 도메인의 신용카드 데이터를 준비한다. 이 데이터는 클래스가 불균형할 것이므로 리샘플링 기법을 통해 클래스 불균형을 완화한다. 이 때 선택할 샘플링 기법은 4.3의 분석결과를 참고하여 도메인의 목적에 적합한 샘플링 기법을 선택한다.

샘플링 데이터집합이 생성되면, 이 데이터로 서포트 벡터 머신을 학습한다. CQL을 지원하는 DSMS에는 파이썬 또는 R을 지원한다. 즉 리샘플링한 데이터집합으로 SVM을 학습시키는 과정도 일체형 시스템 내에서 구현할 수 있다.

학습한 SVM모델을 CQL의 예측 함수로 바꾸기 위해서 두 번의 변환 과정을 거친다. 먼저 파이썬 또는 R로 학습한 SVM모델을 JSON(JavaScript Object Notation) 형식으로 변환하고, 이 변환된 JSON 형식으로부터 클래스 예측을 위한 CQL 함수를 생성한다.

최종적으로 생성한 CQL 함수를 아키텍처에 등록하여 실시간 스트림으로 들어오는 데이터에 대해 정상인지 아닌지 판별하게 된다. 비정상 데이터로 판정된다면 그에 맞는 처리 프로세스가 수행될 것이고, 그러한 비정상 데이터들 중 확실하게 판정된 유형은 따로 모아서 트레이닝 데이터집합에 추가한다. 추가된 새로운 비정상 데이터들은 주기적으로 모델을 업데이트 할 때 반영될 것이다.

5.3 CQL로의 자동 변환

• 학습한 SVM모델의 JSON으로의 변환

SVM모델을 바로 CQL로 변환하지 않고 중간에 JSON 형식을 거치는 이유는 모델링 블록과 CQL 변환 블록을 독립적으로 만들기 위함이다. 모델링 블록이 시스템 환경의 변화로 바뀌거나 업데이트 된다고 해도, JSON에서 CQL로 변환하는 블록은 바뀔 필요가 없게 된다.

SVM모델을 JSON 형식으로 바꿀 때 모델의 계수(coefficients), 절편(intercept), 컬럼 유형(columnTypes), 컬럼 크기(columnSize) 정보가 필요하다.

그림 5는 신용카드 데이터집합에 클래스 비율 1/25인 SMOTETomek 리샘플링 기법을 적용하고, 그 리샘플링한 데이터집합으로 SVM을 학습한 뒤 임계값을 0.48로 적용한 모델을 JSON 형식으로 변환한 결과이다.

svmModelToJSON 함수를 정의하여 SVM모델을 인자로 받아 그림 5와 같은 JSON 파일을 생성하도록 구현하였다. 임계값은 절편에 포함하도록 구현하여 JSON 형식에 따로 나타나지 않는다.

• JSON 모델에서 CQL 함수로의 변환

그림 5의 SVM모델 정보가 들어있는 JSON 파일을 사용하여 그림 6의 두 개의 CQL 함수를 생성한다.


```
{
    "columnTypes": ["float", "float", "float", "float", "float", "float", "float", "float", "float", "float", "float", "float", "float", "float", "float",
                    "float", "float", "float", "float", "float", "float", "float", "float", "float", "float", "float", "float", "float", "float", "float"],
    "columnSize": 28,
    "intercept": -1.6544153333988623,
    "coefficients": [ 0.027903484203155 71, -0.022887402511577497, 0.017839175412158948, 0.09380533590813894, 0.0016683812088693906, -0.025325489783290 23,
                      -0.006339766089412071, -0.05788975363029631, -0.06225813125522529, -0.11593586194409071, 0.058913292280469896, -0.10689903139602813,
                      -0.032715076916087785, -0.13403300510845162, -0.007410655619843989, -0.051716969400218755, -0.048705542900056166, -0.040102831347627 82,
                       0.0016327214147015914, -0.03564606571424532, 0.030705563255842713, 0.03403872886270334, -0.034515595167230 53, 0.015431041194855407,
                      -0.008804159520203683, -0.03824953814182458, -0.007127310015015494, 0.017755043801295004]
}
```


그림 5 서포트벡터머신 모델의 JSON 형식으로의 변환결과
Fig. 5 Result of converting the SVM model to JSON format

```
CREATE FUNCTION get_score(float, float, float, float, float, float, float, float, float, float, float, float, float, float, float,
                          float, float, float, float, float, float, float, float, float, float, float, float, float, float) RETURNS float AS '
SELECT 0.027903484203 2*$1 + -0.022887402511 6*$2 + 0.017839175412 2*$3 + 0.093805335909 1*$4 + 0.001668381208 7*$5 + -0.025325489783 3*$6
    + -0.006339766089 41*$7 + -0.057889753630 3*$8 + -0.062258131255 2*$9 + -0.115935861944*$10 + 0.058913292280 5*$11 + -0.106899031396*$12
    + -0.032715076916 1*$13 + -0.134033005108*$14 + -0.007410655619 84*$15 + -0.051716969400 2*$16 + -0.048705542900 1*$17 + -0.040102831347 6*$18
    + 0.001632721414 7*$19 + -0.035646065714 2*$20 + 0.030705563255 8*$21 + 0.034038728862 7*$22 + -0.034515595167 2*$23 + 0.015431041194 9*$24
    + -0.008804159520 2*$25 + -0.038249538141 8*$26 + -0.007127310015 02*$27 + 0.017755043801 3*$28 + -1.6544153333' LANGUAGE SQL;

CREATE FUNCTION predict(float, float, float, float, float, float, float, float, float, float, float, float, float, float, float,
                        float, float, float, float, float, float, float, float, float, float, float, float, float, float) RETURNS integer AS '
BEGIN
IF (SELECT get_score($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11, $12, $13, $14, $15, $16, $17, $18, $19, $20, $21, $22, $23, $24, $25, $26, $27, $28) as score) < 0 THEN RETURN 0;
ELSE RETURN 1;
END IF;
END'
LANGUAGE PLPGSQL;
```

그림 6 JSON 형식으로 저장된 서포트벡터머신 모델의 CQL 함수로의 변환결과
Fig. 6 Result of converting the SVM model in JSON format to CQL functions

svmCQLBuilder 함수를 정의하여 그림 5와 같은 JSON 파일을 인풋으로 받고, get_score([,columns]), predict ([,columns]) 두 개의 CQL 함수를 생성한다.

get_score CQL함수는 인풋 데이터의 컬럼들과 계수 간의 벡터 내적에 절편을 더한 값을 리턴한다. 이때의 절편은 임계값까지 포함하여 계산된 값이다.

predict CQL함수는 내부에서 get_score 함수를 호출하여 리턴 된 값이 0 이상이면 1을 리턴(사기 데이터로 예측)하고, 그 외의 경우에는 0을 리턴(정상 데이터로 예측)한다.

6. 결 론

사기 탐지 도메인의 데이터집합을 언더 샘플링, 오버 샘플링, 콤비네이션 샘플링 기법들을 적용하여 리샘플링하고, 각각의 샘플링된 데이터집합으로 서포트 벡터 머신을 학습시켜 그 결과를 검증하였다. 클래스 비율을 임의로 설정할 수 있는 샘플링 기법에 대해서는 클래스 비율을 변화시키며 검증 결과가 어떻게 변화하는지 실험하였다.

또한 서포트 벡터 머신의 임계값을 조정하면서 각 기법의 검증지표 그래프의 변화를 분석하였다. 이를 통해 각 리샘플링 기법에서 최대의 F1 스코어를 도출하는 임계값 지점을 파악하고, 그 결과들을 비교함으로써 서포트 벡터 머신과 가장 좋은 궁합을 보이는 몇 가지 샘플링 후보군(SMOTETomek, SMOTE, SMOTEENN)을 도출 할 수 있었다. 추가적으로 도메인의 목적에 따라 특정 검증지표가 중요하다고 할 때, 어떤 샘플링 기법을 선택하고 임계값의 튜닝 범위는 어떻게 잡아야 할지에 대한 전략을 수립할 수 있었다.

최종적으로, 스트림 환경에서 전통적인 DBMS와 SQL을 사용할 때 발생할 수 있는 문제점을 파악하고, DSMS와 CQL을 활용하여 그러한 문제점을 해결할 수 있는 이상 탐지 아키텍쳐를 고안하였다. 그리고 실험에서의 리샘플링과 임계값 조정을 통해 검증 성능을 최적화한 SVM모델을 아키텍쳐의 분류기로 등록하기 위하여 CQL로 변환하였다. 분류기를 CQL로 변환하는 과정은 두 개의 자동화된 변환 블록을 통하여 구현하였다.

7. 미래 작업(Future work)

이번 연구에 사용된 신용카드 데이터처럼 사기 탐지를 위해 공개된 데이터는 거의 없는 상황이다. 즉 사기 탐지 영역에서 여러 개의 데이터집합을 가지고 실험을 수행하는 것에는 한계가 존재하였다. 한편, 좀 더 범용적인 이상 탐지 영역에는 공개된 데이터가 상당 수 존재한다. 따라서 다음 연구에서는 사기 탐지 영역의 데이터집합 이외의 이상 탐지 영역에 속한 여러 가지 데이터집합을 실험에 사용해본다. 그리고 서포트 벡터 머신과 궁합이 잘 맞는 샘플링 기법들이 사기 탐지 영역에서 도출한 결과들과 같은지 비교해 본다. 만약 그렇다면 일반적인 이상 탐지 영역에서, 해당 샘플링 기법들과 서포트 벡터 머신을 함께 쓰는 것이 검증 성능에 좋다는 주장을 강화시킬 수 있다.

또한, 추후 연구에서는 이상 탐지를 위한 분류기를 서포트 벡터 머신뿐만 아니라 랜덤 포레스트(random forest), 인공 신경망(artificial neural network) 등 다양한 머신러닝 기법들로 사용해본다. 즉, 학습한 결과로 좀 더 복잡한 클래스 예측 함수를 갖는 분류기에 대하여 CQL로 변환하는 연구를 수행하려한다.

References

[1] I. Mani and I. Zhang, "kNN Approach to Unbalanced Data Distributions: A Case Study involving Information Extraction," *Proc. of ICML workshop on learning from imbalanced datasets*, 2003.

[2] M. Kubat and S. Matwin, "Addressing the Curse of Imbalanced Training Sets: One-Sided Selection," *Proc. of ICML*, Vol. 97, pp. 179-186, 1997.

[3] G. E. Batista, R. C. Prati, and MC Monard, "A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data," *SIGKDD Explorations*, Vol. 6, No. 1, pp. 20-29, Jun. 2004.

[4] C. Drummond and R. C. Holte, "C4.5, Class Imbalance, and Cost Sensitivity: Why Under-Sampling beats Over-Sampling, *Proc. of the ICML'03 Workshop on Learning from Imbalanced Datasets*, Vol. 11, pp. 1-8, 2003.

[5] N. V. Chawla, "C4.5 and Imbalanced Data sets: Investigating the effect of sampling method, probabilistic estimate, and decision tree structure," *Proc. of the ICML'03 Workshop on Learning from Imbalanced Datasets*, 2003.

[6] R. Motwani et al., "Query Processing, Resource Management, and Approximation in a Data Stream Management System," *CIDR*, pp. 245-256, 2003.

[7] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, Vol. 20, No. 3, pp. 273-297, Sept. 1995.

[8] S. Mukkamala, G. Janoski, and A. Sung, "Intrusion detection using neural networks and support vector machines," *Proc. of IJCNN*, Vol. 2, pp. 1702-1707, 2002.

[9] H. Wenjie, L. Yihua, and Vemuri V. Rao, "Robust anomaly detection using support vector machines," *Proc. of ICML*, pp. 282-289, 2003.

[10] Q.A. Tran, H. Duan, and X. Li, "One-class Support Vector Machine for Anomaly Network Traffic Detection," *The 2nd Network Research Workshop of the 18th APAN*, 2004.

[11] L. Khan, M. Awad, and B. Thuraisingham, "A New

Intrusion Detection System Using Support Vector Machines and Hierarchical Clustering," *The VLDB Journal*, Vol. 16, No. 4, pp. 507-521, Oct. 2007.
[12] A. Banerjee, P. Burlina. C. Diehl, "A Support Vector Method for Anomaly Detection in Hyperspectral Imagery," *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 44, No. 8, pp. 2282-2291, Jul. 2006.
[13] R.C. Chen, T.S. Chen, and C.C. Lin, "A New Binary Support Vector System for Increasing Detection Rate of Credit Card Fraud," *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 20, No. 2, pp. 227-239, 2006.
[14] R.C. Chen, S.T. Luo, X. Liang, V.C.S. Lee, "Personalized approach based on SVM and ANN for detecting credit card fraud," *Proc. of the IEEE International Conference on Neural Networks and Brain*, pp. 810 - 815, 2005.
[15] E. H. Zheng, C. Zou, J. Sun and L. Chen, "SVM-based Credit Card Fraud Detection with Reject Cost and Class-dependent Error Cost," *Proceedings of the PAKDD Workshop*, 2009.
[16] D. L. Wilson, "Asymptotic Properties of Nearest Neighbor Rules Using Edited Data," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 2, No. 3, pp. 408-421, 1972.
[17] 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.
[18] I. Tomek, "Two Modifications of CNN," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 6, No. 11, pp. 769-772, Nov. 1976.
[19] J. Laurikkala, "Improving Identification of Difficult Small Classes by Balancing Class Distribution," *Conference on Artificial Intelligence in Medicine in Europe - Artificial Intelligence in Medicine*, pp. 63-66, 2001.
[20] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of artificial intelligence research*, Vol. 16, No. 1, pp. 321-357, Jan. 2002.
[21] H. He, Y. Bai, E.A. Garcia, and S. Li, "ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning," *Proc. of IEEE International Joint Conference on Neural Networks*, pp. 1322-1328, 2008.
[22] G. E. Batista, A. L. Bazzan, and M. C. Monard, "Balancing Training Data for Automated Annotation of Keywords: a Case Study," *WOB*, 2003.
[23] A. Dal Pozzolo et al., "Calibrating Probability with Undersampling for Unbalanced Classification," *Computational Intelligence, 2015 IEEE Symposium Series on*, pp. 159-166, IEEE, 2015.
[24] G. Lemaitre, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning," *Journal of Machine Learning Research*, Vol. 18, No. 17, pp. 1-5, 2017.

부 록

1. Under-sampling Methods

1972년 Wilson이 고안한 **ENN(EditedNearestNeighbor)[16]**은 K-NearestNeighbor을 이용해 데이터집합을 축소하는 방법이다. 그 당시에는 KNN의 성능을 높이기 위한 목적으로 만들어졌지만, 언더 샘플링의 목적으로 사용할 수 있다.

알고리즘은 다음과 같다. 모든 데이터에 대해 KNN을 수행한다. KNN을 통해 해당 데이터는 k개의 이웃한 데이터 중 다수가 속한 클래스(사기 또는 정상)로 분류된다. 즉, KNN을 통해 모든 데이터는 사기 또는 정상 클래스로 판정된다.

한편, 데이터집합의 각 데이터는 자신의 사기여부를 나타내는 클래스 값을 이미 가지고 있다. 이제 각 데이터에 대해 원래 데이터가 갖고 있는 사기여부를 나타내는 클래스 값과, KNN을 수행한 결과로 판정된 클래스 값이 일치하는지 확인한다. 그리고 원래의 클래스와 KNN으로 판정된 클래스가 같은 데이터만 새로운 샘플 데이터집합에 포함시킨다. 이와 같은 ENN기법을 다수 클래스에 속한 데이터에 적용시킴으로써, 다수 클래스 데이터를 축소하는 언더샘플링 기법으로 활용 가능하다.

RENN(RepeatedNearestNeighbors)[17]은 ENN을 여러 번 수행하는 것이다. ENN으로 새로 생성된 샘플 집합에 대해 또 다시 ENN을 수행하고, 이 과정을 더 이상 샘플링 데이터집합의 변화가 없을 때까지 반복 수행한다. 그 결과, 각 클러스터에 대해 소속이 확실한 데이터들만 남게 된다.

AllKNN[17]은 ENN을 수행할 때, $1 \leq i \leq k$ 범위의 모든 i-NN에 대해 수행한다. 예를 들어 $k=5$이면, $k=1, 2, 3, 4, 5$에 대해 KNN을 모두 수행한다. 그리고 각각의 i-NN에서 판정한 클래스 중 하나라도 원래 클래스와 다른 경우, 데이터집합에서 제외시킨다. 따라서 모든 i-NN에 대해 판정된 클래스들과 자신의 원래 클래스가 일치하는 데이터들만 최종 데이터집합에 포함된다. 이 기법도 ENN, RENN과 마찬가지로 다수 클래스 데이터에 적용시킴으로써 클래스 불균형 문제를 해결하는데 사용된다.

TomekLinks[18]는 데이터 정리(cleaning) 기법으로 사용되거나, 다수 클래스에 적용하여 언더샘플링 기법으로 활용된다. 이 기법은 각각 한 쌍의 데이터의 거리를 계산하는데, 각 쌍을 구성하는 데이터가 다른 어떤 노드와도 더 짧은 거리를 갖지 않으면 그 한 쌍의 데이터를 Tomek-link라 정의한다. 그리고 이 링크(link)에 속한 데이터들을 노이즈 또는 경계에 매우 가까운 것으로 간주하여 제거하는 기법이다.

OneSidedSelection[2]은 TomekLinks를 수행한 뒤, CNN(CondensedNearestNeighbor)추가로 수행한다.

CNN은 클래스 간의 경계에서 멀리 떨어진 다수의 클래스 데이터를 제거하는 목적으로 사용된다. Tomek-Links로 노이즈와 경계면에 매우 가까운 다수 클래스 데이터를 삭제한다. 그리고 CNN을 통해 경계면에서 멀리 떨어진 다수 클래스 데이터도 제거한다. 즉, 다수 클래스에 속한 데이터 중 경계면에 너무 가깝거나 먼 것을 제외하여 축소시키는 방식이다.

NCR(NeighborhoodCleaningRule)[19]은 Wilson의 ENN을 기반으로 하여 알고리즘을 변형한 기법이다. ENN은 다수 클래스 데이터 중에서 특정 데이터의 클래스가 KNN으로 판정한 클래스와 다르면 제거하였다. NCR에서는 KNN으로 판정할 때, 다수결이 아닌 만장일치로 판단한다. 즉 어떤 데이터의 클래스와 이웃하는 k개의 모든 이웃의 클래스가 다른 경우에만 제거하고, 만장일치가 아닌 경우에는 제거하지 않는다.

또한 결정적인 차이는 소수 클래스 데이터에 대해서도 ENN을 수행한다. 단, 판정된 결과와 불일치하는 소수 클래스 데이터를 삭제하는 것이 아니라, 그 소수 데이터의 K-NearestNeighbor인 다수 클래스 데이터들을 제거한다.

NearMiss-1,2 [1]기법은 거리 기반의 리샘플링 기법이다. NearMiss-1은 다수 클래스의 각 데이터마다 소수 클래스에 속한 가장 가까운 데이터 3개를 뽑는다. 뽑은 3개의 데이터에 대해 각각 유클리디안 거리를 계산하고, 계산된 3개의 거리 값의 평균을 구한다. 이제, 다수 클래스의 각 데이터마다 계산된 평균 거리 값이 존재한다. 이 값을 기준으로 데이터를 오름차순으로 정렬한다.

샘플링 집합에는 소수 클래스 데이터들이 모두 들어가 있다. 이제 원하는 클래스 비율이 될 때까지 정렬된 다수 클래스 데이터들을 샘플링 집합에 차례대로 집어넣는다.

NearMiss-2는 다수 클래스의 각 데이터마다 소수 클래스의 가장 먼 데이터 3개를 도출한다. NearMiss-1과 마찬가지로, 다수 클래스의 각 데이터마다 계산된 3개의 값의 평균을 구해서 오름차순으로 정렬한 뒤, 원하는 클래스 비율이 될 때까지 정렬된 다수 클래스 데이터들을 순서대로 추출하여 샘플링 집합에 추가한다.

2. Over-sampling Methods

SMOTE(Synthetic Minority Over-sampling Technique)[20]는 소수 클래스의 데이터를 생성하여 소수 클래스에 속한 데이터 수를 증폭시키는 대표적인 오버샘플링 기법이다. 새로운 데이터는 기존의 소수 클래스 데이터 사이에서 합성되는 특징을 갖는다. 간략한 알고리즘은 다음과 같다.

편의상 데이터의 속성들을 연속변수라 할 때, 각각의 데이터를 연속된 벡터공간에 나타낸다. 임의의 k를 설정하여, 각 소수 클래스 데이터에 대해 K-Nearest Neighbors를 구한다. 이제 각 데이터와 이웃 간에 k개의 벡터 차를 산출해 내고, 산출된 벡터에 0부터 1사이의 임의의 값을 곱해준다. 즉, 이웃한 두 데이터 사이의 임의의 위치에 새로운 소수 클래스 데이터들이 생성된다. 이 과정을 데이터집합의 클래스 비율이 초기에 입력한 원하는 비율이 될 때까지 반복한다. 이 기법을 통해 과적합(overfitting) 문제를 회피하고 소수의 클래스 데이터를 잘 분산된 형태로 증폭시키게 된다.

ADASYN(Adaptive Synthetic Sampling Approach for Imbalanced Learning)[21]는 SMOTE 방법론 위에 데이터의 분포를 함께 고려한 오버샘플링 기법이다. SMOTE는 각 데이터가 소수 클래스 분포의 어디에 위치하는지 상관하지 않고 새로운 소수 클래스 데이터를 합성한다.

그러나 클래스 분류를 위한 머신러닝 알고리즘을 수행할 때, 어떠한 결정 경계가 형성되는지가 성능에 핵심적인 영향을 준다. 따라서 어떤 머신러닝 기법에서는 클래스 간의 경계에 가까운 데이터들이 많을수록 결정 경계 형성에 유리할 수도 있다. 이러한 아이디어에 착안하여 ADASYN는 소수 클래스 데이터 중에서도 학습이 어려운(경계면에 위치한) 데이터를 증폭시키는데 집중한다. 즉 데이터의 분포를 고려하여 경계면에 위치할수록 가중치를 주고, 가중치가 높은 쪽에서 새로운 합성 데이터를 더 많이 생성한다.

3. Combination-sampling Methods

SMOTE+Tomek[22]는 오버샘플링 기법인 SMOTE를 사용하여 클래스 간의 비율을 맞추고, 더 나아가 데이터집합을 정리하는 목적에서 TomekLinks를 활용한다. 단순히 다수 클래스 데이터의 수를 줄이는 것만이 아니라, 결정 경계의 반대편에 잘못 속한 노이즈 데이터들을 제거하는 의미도 있다. 이것은 다수 클래스뿐만 아니라 증폭된 소수 클래스에도 해당한다. 즉 오버샘플링을 바탕으로 불균형한 클래스 데이터의 비율을 맞추는 것을 기본으로 하고, 다른 클래스로 깊이 들어와 있는 데이터들을 제거한다. 이를 통해 데이터집합을 보다 잘 정의된 클러스터 집합으로 만들 수 있다.

SMOTE+ENN[3]은 SMOTE를 기반으로 오버샘플링을 수행하고, 데이터 정리 목적으로 TomekLinks 대신 ENN을 사용한다. ENN은 각 클래스에 대해 상대 클래스로 깊이 들어가 있는 데이터를 제거한다. 이 과정

에서 KNN이 활용되며, ENN의 특성상 TomekLinks보다 좀 더 깊은 데이터 정리를 수행한다.

김 동 효
2016년 홍익대학교 컴퓨터공학부 학사
2016년~현재 서울대학교 컴퓨터공학부 석박사통합과정 재학 중. 관심분야는 데이터베이스, 데이터 마이닝

구 해 모
정보과학회 컴퓨팅의 실제 논문지
제 24 권 제 3 호 참조

김 형 주
정보과학회 컴퓨팅의 실제 논문지
제 24 권 제 3 호 참조