J. Internet Comput. Serv.
ISSN 1598-0170 (Print) / ISSN 2287-1136 (Online)
http://www.jics.or.kr

MLOps를 위한 효율적인 AI 모델 드리프트 탐지방안 연구☆

A Study on Efficient AI Model Drift Detection Methods for MLOps

이 예 은[1] 이 태 진[1*]
Ye-eun Lee Tae-jin Lee

요 약

오늘날 AI(Artificial Intelligence) 기술이 발전하면서 실용성이 증가함에 따라 실생활 속 다양한 응용 분야에서 널리 활용되고 있다. 이때 AI Model은 기본적으로 학습 데이터의 다양한 통계적 속성을 기반으로 학습된 후 시스템에 배포되지만, 급변하는 데이터의 상황 속 예상치 못한 데이터의 변화는 모델의 성능저하를 유발한다. 특히 보안 분야에서 끊임없이 생성되는 새로운 공격과 알려지지 않은 공격에 대응하기 위해서는 배포된 모델의 Drift Signal을 찾는 것이 중요해짐에 따라 모델 전체의 Lifecycle 관리 필요성이 점차 대두되고 있다. 일반적으로 모델의 정확도 및 오류율(Loss)의 성능변화를 통해 탐지할 수 있지만, 모델 예측 결과에 대한 실제 라벨이 필요한 점에서 사용 환경의 제약이 존재하며, 실제 드리프트가 발생한 지점의 탐지가 불확실한 단점이 있다. 그 이유는 모델의 오류율의 경우 다양한 외부 환경적 요인, 모델의 선택과 그에 따른 파라미터 설정, 그리고 새로운 입력데이터에 따라 크게 영향을 받기에 해당 값만을 기반으로 데이터의 실질적인 드리프트 발생 시점을 정밀하게 판단하는 것은 한계가 존재하게 된다. 따라서 본 논문에서는 XAI(eXplainable Artificial Intelligence) 기반 Anomaly 분석기법을 통해 실질적인 드리프트가 발생한 시점을 탐지하는 방안을 제안한다. DGA(Domain Generation Algorithm)를 탐지하는 분류모델을 대상으로 시험한 결과, 배포된 이후 데이터의 SHAP(Shapley Additive exPlanations) Value를 통해 Anomaly score를 추출하였고, 그 결과 효율적인 드리프트 시점탐지가 가능함을 확인하였다.

☞ 주제어 : 인공지능, 머신러닝 모델, 드리프트 탐지, 설명가능한 인공지능, 머신러닝 운영

ABSTRACT

Today, as AI (Artificial Intelligence) technology develops and its practicality increases, it is widely used in various application fields in real life. At this time, the AI model is basically learned based on various statistical properties of the learning data and then distributed to the system, but unexpected changes in the data in a rapidly changing data situation cause a decrease in the model's performance. In particular, as it becomes important to find drift signals of deployed models in order to respond to new and unknown attacks that are constantly created in the security field, the need for lifecycle management of the entire model is gradually emerging. In general, it can be detected through performance changes in the model's accuracy and error rate (loss), but there are limitations in the usage environment in that an actual label for the model prediction result is required, and the detection of the point where the actual drift occurs is uncertain. there is. This is because the model's error rate is greatly influenced by various external environmental factors, model selection and parameter settings, and new input data, so it is necessary to precisely determine when actual drift in the data occurs based only on the corresponding value. There are limits to this. Therefore, this paper proposes a method to detect when actual drift occurs through an Anomaly analysis technique based on XAI (eXplainable Artificial Intelligence). As a result of testing a classification model that detects DGA (Domain Generation Algorithm), anomaly scores were extracted through the SHAP(Shapley Additive exPlanations) Value of the data after distribution, and as a result, it was confirmed that efficient drift point detection was possible.

☞ keyword : Artificail Inteligence, Machine Learning Model, Drift Detection, XAI, MLOps

1. 서 론

최근 몇 년 동안 머신러닝 기술이 발전하면서 금융, 제조, 의료, 보안분야에 이르기까지 일상생활 속에서 다양하게 사용됨에 따라 생성되는 데이터의 양 또한 지속해서 증가하고 있다. 특히 보안 분야에서는 서비스거부공격, 스팸, 피싱 등 다양한 유형의 사이버공격은 몇 년간 기하급수적인 속도로 증가하게 되면서 더욱더 혁신적이

[1] Department of Information Security, Hoseo University., Chungnam, 31499, Korea.
* Corresponding author (kinjecs0@gmail.com)
[Received 4 May 2023, Reviewed 24 May 2013(R2 12 September 2023), Accepted 25 September 2013]
☆ 이 논문은 2022년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2022-0-000-89, 사이버공격 대응을 위한 Life-cycle 기반 공격그룹 식별 및 유형 분석 기술 개발)

고 효율적으로 처리하고자 머신러닝 기술이 도입되고 있다[1]. 이처럼 빅데이터 시대에서 원활하게 시스템을 운영하기 위해 고성능의 모델도 중요하지만, 모델이 생성되고 배포된 이후까지 전체적인 Lifecycle을 관리하는 것이 점점 중요해지는 추세이다. 이처럼 머신러닝은 수많은 데이터의 잠재력을 활용하는 기술로써 실제 환경에서 사용될 때 시간이 지나서도 신뢰할 수 있는 성능을 유지하는 능력이 중요해짐에 따라 MLOps 개념에 관한 관심이 증가하고 있다. MLOps는 생산적인 머신러닝을 설계하고 유지관리하는 것을 목적으로 기본적으로 머신러닝에 대한 통찰력을 제공하고 실제 환경에서 지속해서 좋은 성능을 유지할 수 있는 모델 관리 프로세스를 의미한다. 특히 네트워크 공격을 모니터링 및 분석하는 과정에서 알려지지 않은 예상치 못한 공격이 실시간으로 짧은 시간 동안 이뤄지는 상황에서 발생하는 수천 개의 데이터를 다루기 위해서는 모델 관리가 절대적으로 필수이다. 같은 네트워크 공격이라도 시간이 흐름에 따라 다양해지는 공격자의 공격 방법으로 공격 데이터의 분포도 변화하게 된다. 예를 들어 도메인 이름을 주기적으로 동적으로 생성하는 사이버공격인 DGA의 경우 수천 개의 도메인이 생성되어 시스템을 감염시키며 이후 방어의 추적을 회피하기 위해 지속해서 새로운 도메인을 생성하게 된다. 이런 상황 속에서 새로운 공격으로 인한 모델의 성능저하 문제는 사회에 있어 큰 위험을 초래하게 되므로 문제를 방지하기 위해 관리자는 데이터의 재구성 및 모델의 재학습 등의 후속 조치를 통해 모델을 지속해서 운영할 수 있어야 한다.

운영 중인 머신러닝 시스템의 성능이 저하되고 수용할 수 없게 되는 경우를 드리프트가 발생했다고 한다. 드리프트는 모델 드리프트, 데이터 드리프트, 개념 드리프트 등으로 다양한 원인으로 드리프트가 발생할 수 있다. 가장 흔한 이유로는 데이터 분포의 변화를 원인으로 기존 학습된 모델의 품질이 저하될 수 있다. 이를 막기 위해 지속해서 모델을 모니터링을 하는 과정을 통해 신속하고 정확한 시점에 조치를 할 수 있도록 Drift Signal을 빠르게 파악하는 것이 중요하다. 그러므로 배포된 모델의 상태를 모니터링하면서 장기간 좋은 성능을 유지할 수 있도록 효율적인 드리프트 탐지 방법이 필요하다.

일반적으로 모델의 정확도와 오류율이 초기성능과 비교해 일정 수준 이상의 차이가 발생하면 드리프트가 발생하였음을 의심할 수 있다. 오류율의 경우 모델의 예측값이 실제 정답과 얼마나 차이가 있는지 수치화한 값으로 드리프트 탐지 목적으로 많이 사용된다. 그 이유는 모델이 훈련되는 과정에서 데이터에 의존하게 되는 특징이 존재하기 때문이다. 따라서 데이터 분포가 변경될 때 모델 예측의 판단기능이 더는 올바르게 작동하지 않게 되고 이에 따라 발생하게 되는 오류율을 측정함으로써 드리프트가 발생했음을 확인할 수 있다. 하지만 성능에 의한 드리프트 탐지를 위해서는 기본적으로 정답지가 있어야 하므로 실제 환경에 적용하기 위해서는 많은 시간과 비용이 소요된다. 또한 많은 양의 전체 데이터에서 실제 드리프트가 발생한 구간을 정확히 탐지하는 부분에서도 부족한 모습을 보인다.

따라서 본 논문에서는 네트워크 환경에서 공격 분류모델이 운영된다고 가정하였을 때, 새로 유입되는 전체 시계열 데이터를 대상으로 XAI 지표를 활용하여 Anomaly 기반 드리프트가 발생한 시점을 탐지하는 방안을 제안한다.

본 논문 구성은 다음과 같다. 2장에서는 MLOps 및 모델 드리프트 탐지에 대한 관련연구 및 한계점을 제시하며, 3장에서는 본 논문에서 제안하는 모델의 Framework와 XAI 기반 Anomaly 분석을 통한 드리프트 탐지방안을 제시한다. 4장에서는 DGA 분석모델을 통해 실제 검증을 수행하였고, 5장에서는 결론을 맺는다.

2. 관련 연구

2.1 모델 관리

현재 머신러닝(Machine Learning)은 의료, 금융, 소셜미디어와 같은 일상생활 속 많은 측면에서 활발하게 사용되고 있다[2]. 그러므로 머신러닝 모델은 의사결정 프로세스에 상당한 영향을 미칠 수 있기에 구축과정에서 시스템의 운영목적을 제대로 수행할 수 있도록 설계된다. 하지만 모델 설계자의 기대와 다르게 시간이 지나면서 배포된 모델이 실제 환경에서 지속해서 좋은 성능을 유지하기는 쉽지 않다. 이러한 이유로 관리자가 ML 모델을 배포한 뒤 초기성능과 유사하도록 운영하기 위해서는 모델 분석, 데이터 검증, 테스트 및 디버깅, 모델 모니터링 등의 요소 기반 ML모델의 관리/감독을 통해 안정적인 운영을 해야한다[3].

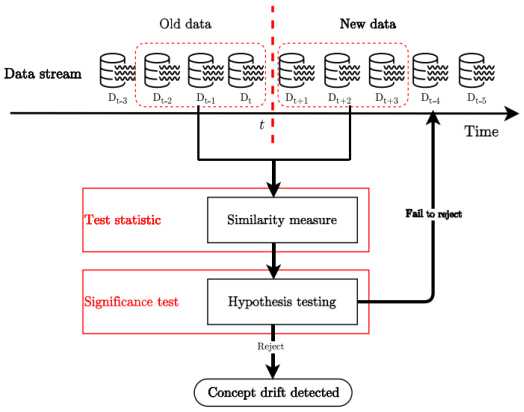
최근 이러한 문제를 인식하여 ML 모델을 관리하고자 DevOps의 기능과 합쳐진 MLOps는 비교적 새롭게 떠오르고 있는 연구 분야로 아직 초기 단계지만 점차 관심이 증가하고 있는 추세이다[4]. MLOps는 기존 학습모델에 새로운 데이터 입력 시 발생하는 문제를 신속하게 확인

하고 재학습을 통해 모델을 지속해서 관리하기 위한 전체적인 프로세스를 의미한다. 이때 ML의 개발과 운영 시스템을 통합하는 5가지 기능을 실현한다[5]. 순서대로 (1) 데이터 수집/전송, (2) 데이터 변환 (3) 지속적인 ML 재훈련 (4) 지속적인 ML 재배포 (5) 최종 사용자에게 생산/프레젠테이션을 출력한다. 즉, MLOps는 특정 시스템 환경에 모델이 배포된 이후 안정적이고 효율적으로 머신러닝 모델이 운영 및 유지되는 것을 목표로 한다. 따라서 이러한 일련의 과정 전체를 자동화 및 통합하는 과정에서 최종적으로는 사용자에게 효율적인 모니터링 파이프라인을 제공하고 모델링을 거쳐 릴리스 후 배포까지 되는 것이 전체적인 MLOps의 목적으로 인식할 수 있다.

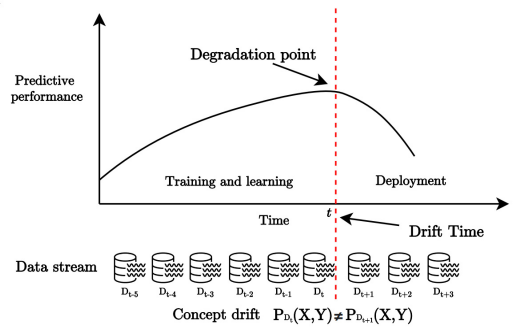
2.2 모델 드리프트 탐지

그림 1의 Concept drift detection framework는 드리프트 탐지의 일반적인 예시로 배포 이전 학습 데이터와 배포 이후 새로운 데이터의 유사성 비교 및 성능 측정을 통해 Drift Signal를 탐지하며, 그림 2의 Performance-based approach mechanism은 일반적인 성능기반 드리프트 측정 방식을 나타낸다[6]. 드리프트는 앞에서 언급한 ML의 개발과 운영 시스템을 통합하는 5가지 기능 중 지속적인 ML 재훈련을 위한 근거로 이러한 드리프트 탐지 방법은 크게 Supervised와 Unsupervised 2가지 방법으로 분류된다. Supervised 환경에서는 표본이 되는 학습 데이터와 새롭게 유입된 테스트데이터에 모두 label이 존재해야 드리프트 탐지가 가능하다.

유사성에는 정확도, 재현율, 민감도 오류율 등의 성능 변화에 따른 탐지방식이 있다. 그중 정확도는 드리프트 발생 시점에서 모델의 성능이 저하됨을 관찰하는 것이다. Abdulbasit A. Darem외 연구에서는 드리프트가 악성코드 분류 정확도에 미치는 영향을 입증하고자 6개의 모델을 대상으로 실험을 진행했다[7]. 실험을 위해 사전에 'DS1' 악성코드 데이터로 모델을 학습하였고 이후 'DS2-DS10'의 최신 악성코드 데이터를 입력으로 성능을 측정한 결과 6개 분류모델에서 특정 시점 이후부터 점차 분류 정확도가 저하되는 것을 확인할 수 있었다. 즉, 지속해서 변화하는 악성코드의 feature value와 label 관계의 변화로 인한 드리프트가 발생했음을 확인할 수 있었다. 하지만 해당 방법은 라벨링의 과정이 필요하므로 실시간 분석 및 신속한 탐지 측면에서 사용하기 쉽지 않다는 단점이 있다.



(그림 1) 개념 드리프트 탐지 프레임워크[6]
(Figure 1) Concept drift detection framwork[6]



(그림 2) 성능기반 접근 메커니즘[6]
(Figure 2) Performance-based approach mechanism[6]

모델 오류율(loss) 변화측정은 모델 모니터링 과정에서 드리프트를 탐지할 수 있는 주된 방법이다. 오류율은 기본적인 모델을 평가하는 지표로 최적의 머신러닝 모델을 생성하기 위해 오류율을 최소화하는 것이 목표이다. 이때 분류모델과 회귀모델에 따라 적절한 오류율 함수(loss function)를 선택할 수 있다. 먼저 회귀모델의 loss function 중 MSE(Mean Squared Error)는 실제값과 예측값의 차이를 제곱하여 평균화한 값으로 생성된 모델이 데이터와 차이가 어느 정도 있는지 확인할 수 있고 분류모델의 예측값은 실수형인 회귀모델과 다르게 0과 1로 분류되기 때문에 주로 entropy 방식을 통해서 loss값이 산출된다. 즉 산출과정에서 데이터의 분포가 반영되기 때문에 이를 활용해 데이터 내의 드리프트 발생 여부를 확인할 수 있다.

하지만 Daniel Vela외 연구에서는 모델 loss값은 선택한 모델과 사용자가 설정한 파라미터 그리고 새롭게 유

입되는 입력데이터 간의 관계와 여러 외부 환경요인들로 인해 드리프트 탐지에서 불확실한 결과를 보일 수 있다는 한계점을 언급하였다[8]. 실험을 위해 무작위로 선택된 배포 시간(t0)에서 새로운 데이터가 입력되는 기간(dt)에 따라 3가지 시나리오를 생성 후 각각의 MSE값을 산출하여 그래프를 통해 비교하였다. 그 결과 모델 드리프트 탐지에 오직 loss만을 활용하는 것은 정확한 발생 시점을 탐지하기에 불확실한 정보임을 보여주었다.

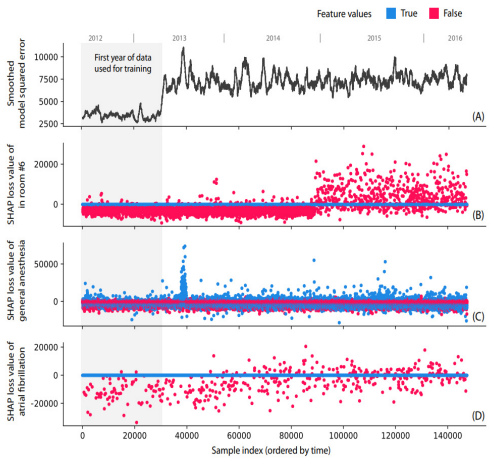
첫 번째는 사용된 환자 및 날씨 데이터의 상당한 변화에도 불구하고 장기간 점진적으로 유사하게 loss값이 산출되는 경우이다. 이러한 상황이 지속해서 이어지게 되면 전체 데이터 흐름에서 드리프트가 발생한 정확한 시점을 탐지 및 판단하기 어려워 최종적으로 분석가가 Drift Signal에 대한 확신을 가질 수 없으므로 모델 고장의 원인이나 성능검사 및 재학습의 원인으로 사용하기에 어렵다.

두번째는 시간에 따른 점진적인 오류율의 증가가 아니라 오랜 시간 우수한 성능을 유지하던 중 갑작스러운 loss값의 변화가 나타났을 경우이다. 단순히 과거와 다르게 갑작스럽게 변화한 구간이 드리프트라고 판단할 수도 있겠지만, 실험결과에서 NN모델은 약 1년 이후부터 성능이 저하되었지만, RV 모델은 1-2년까지는 좋은 성능을 보이고 이후 성능이 저하되는 모습을 보였다. 즉, 사용자가 선택한 모델과 특정 데이터에 대한 모델의 적합성에 따라 loss값의 차이가 존재할 수 있음을 나타낸다.

마지막으로 단순 시간적 증가가 오류율 증가의 원인이 아닐 수 있음을 보여준다. 만약 어떠한 데이터를 대상으로 분류하기 위해 NN 모델과 XGBoost 모델을 선택 및 학습한 이후 새로운 데이터와의 loss값을 측정하는 경우 설계자가 모델을 생성할 때 선택한 설정, 하이퍼 매개변수, 학습 데이터의 크기 등의 추가적인 외부 요인으로 인해 loss값에 변동성이 발생한다는 사실을 알 수 있다. 따라서 loss값은 데이터 관점이 아닌 모델 관점으로써 영향을 크게 받기에 실제 드리프트가 발생한 시점을 탐지하기에는 불확실한 결과를 제공함을 알 수 있다.

2.3 XAI 기반 모델 드리프트 탐지

기술발전에 따라 점점 더 많은 분야에서 머신러닝 모델이 사용되면서 AI 시장의 규모도 점차 확대되고 있으나 모델의 높은 성능으로 인한 복잡성과 해석성 및 투명성과의 trade-off 관계로 인하여 AI 시스템의 판단을 100% 신뢰할 수 없다는 점이 실질적인 활용에서 걸림돌로 작용하고 있다. 높은 예측성능을 보이더라도 모델의 블랙박스(Black Box) 문제는 분석가가 어떠한 근거를 통해서 AI가 판단하였는지 알 수 없게 만든다. 그래서 나온 개념이 XAI(eXplainable Artificial Intelligence)이다. AI 결과에 관한 판단 근거를 제공함으로써 AI기술의 신뢰도를 높이고 이로써 AI의 활용범위를 확장시킬 수 있는 기술로 의료/헬스케어 분야 서비스 외에도 제조/보안분야 등 여러 방면에서 현재 많은 연구가 진행되고 있다[9].



(그림 3) 개념 드리프트 탐지 프레임워크[11]
(Figure 3) Concept drift detection framwork[11]

Lundberg 등은 모델의 설명 가능성을 위해 XAI기술로써 Shapley value 기반 SHAP(SHapley Additive exPlanations)을 제안했다[10]. SHAP value는 각 feature 별 모델 예측에 대한 기여도를 측정하여 모델 판단의 근거를 제공하는 기술이다. Feature의 기여도는 단일 특징을 제거하였을 때 모델 판단이 변화하는 정도를 의미하며 모델 판단에 중요한 특징일수록 높은 값이 산출된다.

Lundberg, S.M.외 연구에서는 SHAP value를 통해 드리프트 탐지와 기여하는 feature를 발견할 수 있음을 보인다[11]. 그림 3은 SHAP value를 활용하여 모델 배포 이후 모니터링의 예시를 나타내는 결과 그래프이다. 특히 (a)와 (d)를 비교하여 드리프트 탐지 활용의 가능성을 주장한다. 실험을 위해 2015년 이후부터 room no.6 label과 room no 13 label을 변경한 테스트데이터를 사용하였다. 이후 기존 loss만을 사용하였을 때와 SHAP value를 사용하였을 때의 드리프트 탐지결과를 비교한다. 그 결과 (a)의 그래프는 드리프트 탐지에서 흔하게 사용하는 모델 예측의 전반적인 손실 그래프로 2012년에는 학습데이터의 loss값으로 낮은 수치를 보이지만, 테스트 데이터가 입력되는 시점에서는 불가피한 손실의 증가 폭을 보인다. 드리프트

탐지 관점에서 예상대로라면 2015년부터 데이터 라벨의 변경으로 loss 값이 증가해야 하지만, 해당 그래프를 통해서는 드리프트가 발생한 정확한 시작과 진행 상황을 추측하기 힘들다. 하지만 (b)는 정확히 드리프트가 발생한 지점에서 SHAP value를 통해 오류 증가에 대한 기여도 변화를 통합하여 검출될 수 있음을 보였다. (c)는 시스템 내부적 측정방식의 문제 발생으로 인한 특정 기능 오류를 감지할 수 있음을 보이며 (d)는 모델의 손실 함수에 대한 시간적 흐름에 따른 SHAP value이다. 점진적인 드리프트가 일어난 경우, 그래프에 나타난 것과 같이 SHAP Loss의 변화를 통해 드리프트 발생을 인지할 수 있음을 주장한다.

결국 드리프트 탐지는 모델이 배포된 이후 업데이트가 필요한 시점을 알리는 신호이므로, 다음 장에서는 이를 탐지하는 방안을 제시한다.

3. 제안 모델

이장에서는 XAI 중 모델 예측 과정에서 feature의 기여도를 나타내는 SHAP value를 활용하여 데이터 변화로 인한 드리프트를 세밀하게 탐지하고 변화가 발생한 시간대 추적을 위한 방법을 제안하고자 한다.

3.1 제안 방법

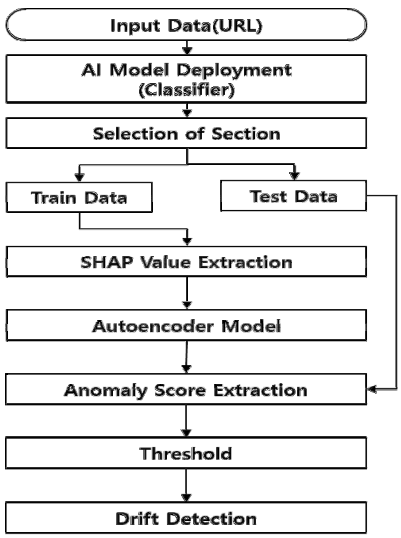
일반적인 드리프트 탐지는 모델에 입력되는 데이터를 공격과 정상으로 구분할 필요 없이 전체를 대상으로 Anomaly 측정을 통해 탐지할 수 있다. 하지만 본 실험에서는 드리프트 유발을 위한 변화하는 정상 데이터를 만들지 못했기에 공격일 때의 시계열과 정상일 때의 시계열을 나누어서 사전에 설정한 임계값에 따라 드리프트를 탐지하고자 한다. 본 논문에서 제안하는 Framework는 그림 4와 같으며 자세한 내용은 아래와 같다.

3.1.1 DGA 분류 모델 생성

먼저 DGA 공격과 정상 데이터를 대상으로 URL 문자열의 통계적인 특징을 반영한 Lexical 기반 Featuring을 적용한 후 분류를 위해 XGBoost 모델을 생성한다. XGBoost는 Gradient boosting 알고리즘 기반으로 여러 개의 분류기를 생성하는 앙상블 학습을 통해 높은 예측값 산출이 가능한 모델이다. 좀 더 세밀한 학습을 위해 세부 파라미터를 조정하였으며 DGA인 경우 0, 정상이면 1로 분류되는 이진 분류 탐지모델을 대상으로 언제 모델이 갱신되어야 하는지 판단하고자 한다.

3.1.2 구간 설정

먼저 데이터의 구간을 선정하는 과정은 드리프트를 추적하기 위한 가장 첫 번째 단계로써 그림 1에서 볼 수 있듯이 드리프트를 추정하기 위해서는 Old Data와 New Data가 존재한다. 이때 Old Data는 배포 이전의 학습 데이터이며 드리프트 탐지를 위한 참조 데이터를 의미한다. New Data는 모델이 배포된 이후 드리프트가 발생하게 될 탐지 대상의 테스트데이터로, 해당 데이터 내에서 드리프트를 탐지하고자 한다.



(그림 4) 제안 프레임워크
(Figure 4) Proposed framework

3.1.3 SHAP 추출

데이터의 통계적인 특성은 생성된 시기뿐만 아니라 전 기간에 걸쳐서 빠르게 변화한다. 그러므로 데이터 내의값들은 여러 가지 단위와 범위로 구성되기 때문에 데이터가 변화한 정도를 측정하는 과정에서 어려움이 존재한다. 따라서 이를 보완하기 위해 일반적으로는 일정값 범위로 스케일링하는 과정을 거치는 등의 모델학습에 적합한 값으로 변화를 주어야 한다. 이를 위해 앞선 Lundberg. S.M. 연구에서 모델 loss값에 기여한 SHAP value를 통해서 드리프트를 탐지할 수 있다라는 주장을 참고하여 통계적

변화량을 측정하는 과정에서 SHAP value를 활용하고자 한다. 따라서 전체 데이터를 대상으로 SHAP 값을 수식 1과 같이 추출하고 이후 추가적으로 입력데이터의 feature 중 모델의 결과와 성능에 중요한 Feature를 선별하여 향상된 드리프트 탐지를 하고자 한다.

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|(|F|-|S|-1)!}{|F|!} [f_{S\cup\{i\}}(x_{S\cup\{i\}}) - f_S(x_S)] \quad (1)$$

3.1.4 드리프트 탐지 모델 생성

이 과정에서는 드리프트를 실질적으로 탐지하는 모델을 생성한다. 배포 이전의 학습 데이터 전체를 대상으로 SHAP을 적용한 후 전체 SHAP value에서를 활용하여 Anomaly 모델인 Autoencoder을 생성하고자 한다. 이때 Autoencoder 는 입력과 출력의 편차를 기반으로 복원 오차 값을 통해 정상과 다른 패턴을 보이는 이상 패턴을 탐지할 수 있어서 주로 이상 탐지 목적의 연구에서 자주 사용되는 모델이다. 이에 따라 본 실험에서는 사용하는 데이터 세트에 적합한 탐지를 위해 공격과 정상의 시계열을 나누어서 진행하였으며, 각각의 SHAP value를 학습한 두 개의 Autoencder 모델을 생성하였다. 추가로 세밀한 학습을 위해 activation은 'relu', epoch는 100 batch_size는 256의 세부 파라미터를 사용하였다. 이때 Activation을 위해 'relu'를 사용한 이유는 다른 활성화 함수 대비 계산이 효율적이기에 학습 속도가 빠르고, sigmoid와 같은 활성화 함수의 경우 기울기 소멸 문제(Vanishing gradient problem)을 초래할 수 있지만, 해당 함수는 양수 값에서 1이라는 점에서 이러한 문제를 줄일 수 있기에 더 효율적인 모델을 만들 수 있다는 점에서 활용하였다.

3.2 모델 드리프트 탐지

배포된 모델의 드리프트를 탐지하고자 생성한 Autoencoder 탐지모델에 테스트데이터를 입력으로 Anomaly score을 추출하여 사전에 설정한 Threshold 이상이면 드리프트가 발생한 시점으로 판단하고자 한다.

제안 방법의 유용성을 검증하고자 배포된 분류모델의 loss값과 비교하고자 한다. 분류모델의 loss값을 산출은 XGBoost의 내장된 이진 분류의 오류함수인 error metric을 사용하였다. 이후 분류모델에 학습 데이터와 드리프트가 포함된 테스트데이터를 입력으로 loss값을 산출하고 사전에 설정한 Threshold 이상인 경우 드리프트라고 판단하고자 한다. 이를 통해 최종적으로 loss값을 통한 드리프트 탐지와 Anomaly score를 통한 드리프트 탐지 방법을 비교하였다. 마지막으로 드리프트가 포함되지 않은 경우와 포함되었을 때 발생하는 모델의 성능저하를 확인하고자 모델의 정확도를 측정하여 비교 및 검증하고자 한다.

4. 실험 결과

4.1 데이터 셋

모델 배포 이전과 이후의 드리프트 구간을 찾기 위해 사전에 데이터의 구간 선정이 필요하다. 따라서 본 논문에서는 NetLab에서 제공하는 DGA 데이터 세트를 활용하고 실험을 위해 새로운 DGA 기술이 도입되는 시점을 추가한 데이터를 사용한다. NetLab은 Oihoo 360의 보안팀으로 2014년에 창립되어 보안 데이터 중 특히 봇넷, 허니팟, 대규모 DNS 데이터 및 보안 데이터 관련 연구에 주력하는 팀으로 본 연구에서는 해당 팀에서 제공하는 DGA 데이터를 사용하고자 한다. 이때 드리프트의 정상 데이터는 배포 이전의 정상 데이터와 다른 성격의 변화한 데이터를 생성하지 못하였기 때문에 전체 시계열에서 공격과 정상 2개의 시계열로 나누어서 진행하고자 한다. 따라서 공격 데이터의 NO.1과 NO.2는 DGA 중 'pykspa_v1'에 해당하는 같은 유형이며 NO.3은 성격이 다른 DGA 중 'flubot'으로 구성된다. 배포 이전과 배포 이후 드리프트 데이터의 DGA 공격유형의 차이는 표 3과 같다. 정상과 공격의 전체 데이터 구성은 표 1과 표 2와 같다. 공격 시계열 데이터에서 배포 이전 학습 데이터는 5,031개, 배포 이후 테스트데이터 구간은 2,012개, 드리프트 유발 데이터 구간은 500개 데이터를 사용한다. 정상 시계열 데이터는 배포 이전 학습 데이터는 4,969, 배포 이후 테스트데이터 구간은 1,988, 드리프트 유발 데이터 구간은 500개를 사용하여 각각 시계열 데이터 대상 드리프트 탐지 결과를 산출한다.

추가로 본 실험의 목적은 드리프트 시간대를 찾는것이 목적이므로 임의로 데이터에 시계열 인덱스를 추가한다. 먼저 공격의 배포 이전 모델은 2002-01-01부터 2015-10-10까지의 데이터로 학습된다고 가정한다. 배포 이후 테스트데이터는 2015-10-11부터 입력되며 2021-04-13부터 드리프트가 발생한 시점이다. 다음 정상의 배포 이전 모델은 2002-01-01부터 2015-08-26까지의 데이터로 학습된다. 배포 이후 테스트데이터는 2015/08/27부터 입력되며 2021-02-04부터 드리프트가 발생한 시점이다. 최종적으로 본

실험은 드리프트가 발생한 시간대를 탐지하는 것을 목표로 한다.

(표 1) 공격 데이터 세트
(Table 1) Attack dataset

No.	Data Type	Number of Data
1	Train	5,031
2	Test	2,012
3	Drift	500

(표 2) 정상 데이터 세트
(Table 2) Normal dataset

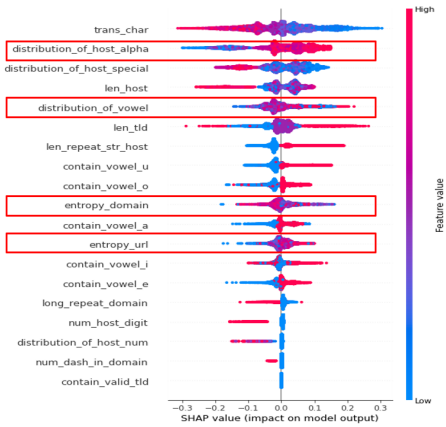
No.	Data Type	Number of Data
1	Train	4,969
2	Test	1,988
3	Drift	500

(표 3) 공격유형 비교
(Table 3) Attack type comparison

	pykspa_v1	flubot
TLD	biz, com, net	ru, com, cn, su
Length	6-15, a~z	15, a~y
Period	2 day	1 month
Count	5,000	5,000
Example	agadss.biz	bsgeijagbavgavk,cn

4.2 SHAP 추출

드리프트 구간 탐지를 위해 기본으로 전체 학습 데이터 대상 SHAP value를 추출한다. XAI 기술 중 하나인 SHAP의 예측 결과는 설명자(explainer)에 의해 설명된다. SHAP의 Explainer는 모델에서 가장 주요한 몇 가지 feature을 강조해주는 역할을 하며 파라미터로 학습 데이터와 생성한 모델을 사용한다. 본 실험에서는 Lundberg가 제안한 방법으로 Decision Tree, Random Forest, Gradient Boosted Tree와 같은 트리 기반 모델을 위한 Tree Explainer SHAP을 사용한다. 해당 방법은 빠르고 정확한 Shapley value를 계산한다. 추출된 SHAP value를 기반으로 특정 시간대의 드리프트가 발생한 구간에 대해서 드리프트 탐지모델을 통해 Anomaly score를 추출하여 그래프로 시각화하여 분석하고자 한다. 이를 활용하여 모델 예측에 주요하게 기여한 feature를 선정하고자 그림 5와 같이 Global 관점의 Plot인 Summary Plot을 분석하여 붉은색 구역로 표시한 특징값이 섞인 특징을 제외하고 총 19개 중 10개의 feature을 선정한 특징목록은 표 4와 같다.



(그림 5) SHAP 요약 플롯
(Figure 5) SHAP summary plot

(표 4) 중요 특징 리스트
(Table 4) Importance feature list

No.	Feature Name
1	trans_char
2	distribution_of_host_special
3	len_host
4	len_tld
5	len_repeat_str_host
6	contain_vowel_u
7	contain_vowel_o
8	contain_vowel_a
9	contain_vowel_i
10	contain_vowel_e

4.3 실험 결과

네트워크 환경에서 지속 유입되는 URL 데이터를 대상으로 DGA 여부를 탐지하는 배포된 모델을 대상으로 3가지 실험을 통해서 드리프트를 탐지하고 비교하였다. 첫 번째, 모델 정확도는 측정을 위해 데이터의 라벨링하는 과정을 거쳐 성능이 저하되었을 때 드리프트라고 판단할 수 있다. 두 번째, 모델 loss는 이전 loss 값과 드리프트라 의심되는 증가한 loss 값과의 비교계산을 통해서 판단할 수 있다. 마지막으로 제안 방법은 드리프트 탐지모델로 산출된 Anomaly score가 Threshold이상인 경우 기존방법 대비 드리프트가 발생한 시점의 시간대를 탐지할 수 있었다.

4.3.1 모델 오류율 기반 드리프트 탐지

제안 방법의 실험에 앞서 일반적인 드리프트 탐지 방

법 중 모델의 loss 값의 변화를 통해 드리프트를 탐지하고자 한다. 따라서 배포 이전 학습 데이터로 학습된 모델에 대해서 전체 시계열 데이터 중 공격과 정상의 시계열로 나눈 테스트데이터를 입력으로 드리프트가 포함된 테스트와 포함되지 않은 테스트에서 변화하는 모델의 loss 값을 비교하였다. 이때 사용한 loss function은 이진 분류데이터에서 주로 사용하는 error metric으로 전체 데이터 중 잘못 분류한 비율을 의미한다.

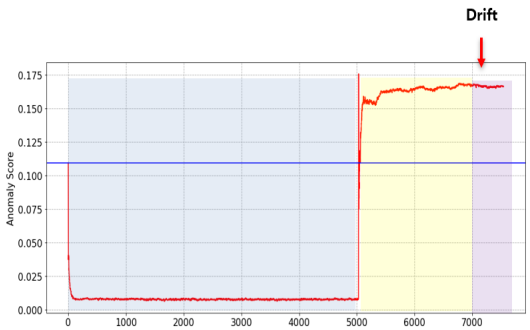
우선, 공격에 대해 loss 값을 산출한 결과는 그림 6과 같다. loss 산출을 위해서 XGBoost 분류모델에 입력으로 전체 학습 데이터 중 공격만 추려진 데이터와 드리프트가 포함된 공격 테스트데이터를 입력으로 추출을 진행하였다. 그 결과 테스트데이터가 입력되는 시점부터 Threshold 이상으로 드리프트 데이터로 의심할 수 있다. 하지만 앞서 설정한 데이터 구간에 따르면 6,900 이후 약 7000 index 지점 이상부터 실제 드리프트가 발생한 지점이지만, loss값의 결과만으로는 해당 지점에서 드리프트가 발생한 것을 확인할 수 없었다. 즉 모델에 입력된 데이터의 변화로 loss값이 증가하였지만 실제 데이터가 변화한 구간에서 증가한 것이 아니라 단순 입력된 테스트 전체 구간에서 loss값이 증가하여진 것으로 보이기에 실제 드리프트가 발생한 지점을 탐지하기에 어려움이 존재한다는 사실을 알 수 있었다.

정상인 경우에도 같은 과정을 통해서 진행하였으나 데이터를 구성하는 과정에서 드리프트를 위한 변화하는 데이터를 생성하지 못하였다. 따라서 그림 7과 같이 모델의 loss값이 Threshold 이하로 산출됨에 따라 정상 데이터는 드리프트가 아닌 것으로 판단할 수 있다. 이러한 결과를 통해 드리프트 발생 시 실시간으로 정확한 탐지가 어렵다는 사실을 알 수 있었다.

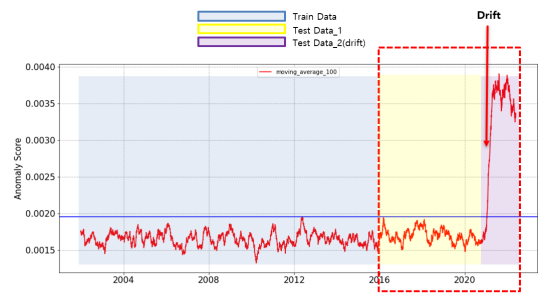
(그림 7) 정상 시계열에 대한 모델 손실
(Figure 7) Model loss for normal time series

4.3.2 XAI를 활용한 Anomaly 기반 드리프트 탐지

공격 구간 드리프트 탐지모델에 입력되는 학습 데이터는 테스트데이터 중 공격으로만 추려진 SHAP value 값을 사용하며, 이때 일부는 학습 데이터와 유사한 데이터이고 나머지는 드리프트가 포함된 데이터를 사용하여 드리프트 탐지를 진행한다. 그래서 전체 7,543개의 공격 데이터를 활용하여 Autoencoder 학습한 뒤 Anomaly Score를 추출하여 배포 이전 공격 학습 데이터의 Anomaly score와 드리프트가 포함된 배포 이후 공격 테스트데이터의 Anomaly score 흐름을 비교한다. 이때 Anomaly score는 1초씩 WindowSize 100으로 슬라이딩 윈도우를 진행한다. 탐지에 앞서 사전에 설정한 공격 데이터 구성에서 드리프트가 탐지되는 시점은 7,043 index로 2021-04-13에 발생하게 된다. 탐지 결과 전체 공격 시계열에서 Threshold 이상인 드리프트 구간은 100개 단위의 슬라이딩 윈도우로 인해 앞의 100개 size를 제외하여 6,944 index에 탐지되고 2020-10-10 이후부터 드리프트가 발생한 시점으로 판단할 수 있다. 그 결과는 그림 8과 같으며 드리프트가 탐지



(그림 6) 공격 시계열에 대한 모델 손실
(Figure 6) Model loss for attack time series



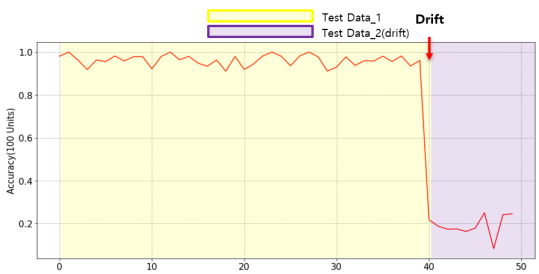
(그림 8) 이상탐지기반 드리프트 탐지
(Figure 8) Anomaly-based drift detection

된 테스트데이터를 확대한 결과는 그림 9와 같다. X축은 공격 데이터의 전체 시간대이며 Y축은 Anomaly score를 의미한다. 이는 단순히 입력된 데이터의 전체가 아니라 특정 구간에서 드리프트가 발생했음을 알 수 있으며, 이는 실시간 탐지에 있어서 loss값 기반 탐지와 차별점이 있음을 알 수 있다.

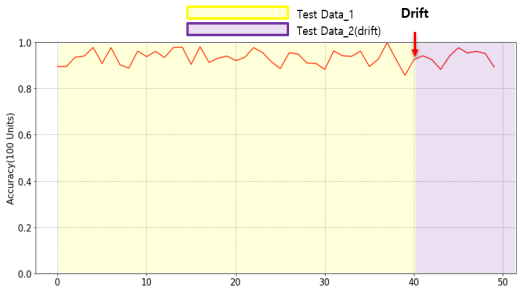
(그림 9) 테스트 데이터의 드리프트 탐지
(Figure 9) Drfit detectin in test data

4.3.3 모델 정확도 기반 드리프트 탐지

마지막으로 드리프트가 포함된 테스트데이터 대상 정상과 공격 각각에 대해서 변화하는 모델의 성능 비교를 통해 검증하고자 한다. 우선 드리프트가 포함되지 않은 테스트데이터의 경우 0.95의 높은 성능을 보였다. 그에 반해 드리프트가 포함된 테스트데이터의 경우 0.87로 드리프트가 포함되지 않았을 때 비해 전체적으로 성능이 0.08 이하로 저하됨을 알 수 있었다. 다음은 각 공격과 정상 시계열 대상 100개 단위씩 탐지율을 측정한 결과, 그림 10, 그림 11과 같다. 정상이면 배포 이전 학습 데이터와 다르지 않기에 높은 정확도가 유지됨을 보이지만, 공격이면 특정 시점에서 성능이 확연하게 저하되는 것을 보인다. 이를 통해 모델에 기존 학습 데이터와 다른 유형



(그림 10) 공격의 모델 정확도
(Figure 10) Model accuracy in attack



(그림 11) 정상의 모델 정확도
(Figure 11) Model accuracy in normal

데이터의 입력 시 모델의 품질이 저하된다는 사실을 검증할 수 있었다. 그러나 실험환경에서는 정답지가 존재함에 따라 성능을 비교하여 드리프트가 발생했음을 알 수 있지만, 실제 환경에서는 라벨이 없는 Unsuperevised 환경이기에 성능 비교를 통한 드리프트 탐지에는 어려움이 존재한다.

5. 결론 및 향후 연구과제

오늘날 머신러닝 모델의 역할이 확장됨에 따라 모델 관리의 중요성이 점차 중요하게 여겨지고 있다. 이때 모델은 시간이 지나면서 변화하는 데이터로 인해 성능이 저하되는 드리프트가 발생하게 되고, 이로 인해 기존 모델의 무력화와 전체 시스템의 문제를 발생시키게 된다. 이런 문제를 예방하기 위해 일반적으로 모델의 정확도와 loss값을 추적하여 드리프트를 탐지하고자 한다. 하지만 정확도는 모델 예측과 결과에 대한 정답이 존재하는 경우에만 사용할 수 있어, 정답지가 없는 환경에서는 적용하기 어렵고 사후적인 방식이기에 실시간 탐지를 할 수 없다는 단점이 있다. 따라서 주로 모델의 loss값을 통해 드리프트 탐지가 이뤄지게 되는데 Diniel Vela외 연구에서와같이 모델의 loss값을 통한 탐지는 데이터 변화에 따른 세밀한 탐지를 하지 못한다는 단점이 존재하였다. 드리프트가 포함된 테스트데이터에 대해서 전체적으로 Threshold 이상의 높은 loss값을 보였지만 실질적으로 드리프트가 발생한 위치를 알 수 없었기에 드리프트만을 원인으로 loss값이 증가한 것이 맞는지에 대한 불확실성으로 인해 사용에 어려움이 있었다.

본 논문은 이를 보완한 안정적인 모델 관리를 위해 SHAP value을 활용한 Anomaly 기반 드리프트 시점 탐지 방안을 제안하였다. 배포 이후의 데이터에 대해서 SHAP

value 기반 Anomaly score를 산출하였고 사전에 설정한 데이터의 드리프트 구간에서 정확히 Threshold 이상의 값이 산출됨에 따라 제안 방법의 유효성을 검증할 수 있었다. 이때 탐지하는 과정에서 슬라이딩 윈도우 사이즈를 줄인다면 신속한 탐지는 가능하겠지만, 너무 작은 값은 드리프트 판단에 있어 noise를 줄 수 있으므로 운영하는 상황에 따라 적절한 운용이 필요할 것으로 보인다.

또한 향후 모델 관리의 필요성과 중요성이 증가함에 따라 본 연구에서 제안한 방법론의 범용성을 검증하기 위해 다양한 유형의 데이터 셋 대상 드리프트 탐지의 효율성 및 SHAP value외의 다른 XAI 기법들과의 통합을 통해 더욱 정확한 드리프트 탐지방안의 연구가 필요할 것으로 보인다. 또한 드리프트가 탐지되었을 때 해당 드리프트의 원인을 분석하고 이를 해결하려는 방안을 추가로 진행한다면 데이터 패턴의 변화나 외부 환경적 요인들이 드리프트에 어떠한 영향을 주는지 분석함으로써 앞으로의 연구에 크게 기여할 수 있을 것이라 기대된다.

참고문헌(Reference)

[1] Sarker, I. H., Kayes, A. S. M., Badsha, S., Alqahtani, H., Watters, P., & Ng, A., “Cybersecurity data science: an overview from machine learning perspective,” Journal of Big data, 7, 1-29, 2020. https://doi.org/10.1186/s40537-020-00318-5

[2] Bhatt, U., Xiang, A., Sharma, S., Weller, A., Taly, A., Jia, Y., ... & Eckersley, P., “Explainable machine learning in deployment,” in Proceedings of the 2020 conference on fairness, accountability, and transparency, pp. 648-657, January 2020. https://doi.org/10.1145/3351095.3375624

[3] Spjuth, O., Frid, J., & Hellander, A., “The machine learning life cycle and the cloud: implications for drug discovery,” Expert opinion on drug discovery, 16(9), 1071-1079, 2021. https://doi.org/10.1080/17460441.2021.1932812

[4] John, M. M., Olsson, H. H., & Bosch, J., “Towards mlops: A framework and maturity model,” in 2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), IEEE, pp. 1-8, September 2021. https://doi.org/10.1109/SEAA53835.2021.00050

[5] Tamburri, D. A., “Sustainable mlops: Trends and challenges,” in 2020 22nd international symposium on symbolic and numeric algorithms for scientific computing (SYNASC), IEEE., pp. 17-23, September 2020. https://doi.org/10.1109/SYNASC51798.2020.00015

[6] Bayram, F., Ahmed, B. S., & Kassler, A., “From concept drift to model degradation: An overview on performance-aware drift detectors,” Knowledge-Based Systems, 108632, 2022. https://doi.org/10.1016/j.knosys.2022.108632

[7] Darem, A. A., Ghaleb, F. A., Al-Hashmi, A. A., Abawajy, J. H., Alanazi, S. M., & Al-Rezami, A. Y., “An adaptive behavioral-based incremental batch learning malware variants detection model using concept drift detection and sequential deep learning,“ IEEE Access, 9, 97180-97196, 2021. https://doi.org/10.1109/ACCESS.2021.3093366

[8] Vela, D., Sharp, A., Zhang, R., Nguyen, T., Hoang, A., & Pianykh, O. S., “Temporal quality degradation in AI models,” Scientific Reports, 12(1), 11654, 2022. https://doi.org/10.1038/s41598-022-15245-z

[9] Xu, F., Uszkoreit, H., Du, Y., Fan, W., Zhao, D., & Zhu, J., “Explainable AI: A brief survey on history, research areas, approaches and challenges,” in Natural Language Processing and Chinese Computing: 8th CCF International Conference, NLPCC 2019, Dunhuang, China, October 9–14, 2019, Proceedings, Part II 8, pp. 563-574, Springer International Publishing, 2019. https://doi.org/10.1007/978-3-030-32236-6_51

[10] Lundberg, S. M., & Lee, S. I., “A unified approach to interpreting model predictions,” Advances in neural information processing systems, 30, 2017. https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html

[11] Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., ... & Lee, S. I., “From local explanations to global understanding with explainable AI for trees,” Nature machine intelligence, 2(1), 56-67, 2020. https://doi.org/10.1038/s42256-019-0138-9

◐ 저 자 소 개 ◑

이 예 은(Ye-eun Lee)
2019년 3월~현재 호서대학교 컴퓨터공학부 학석사과정
관심분야 : 악성코드 분석, 침입 탐지, 이상징후 탐지, 정보보호, AI
관심분야 : 데이터베이스, etc.
E-mail : judiaye4477@gmail.com

이 태 진(Tae-jin Lee)
2003년 2월 포항공과대학교 컴퓨터공학과
2008년 2월 연세대학교 컴퓨터공학과 석사
2017년 2월 아주대학교 컴퓨터공학과 박사
2013년 1월~2017년 2월 한국 인터넷진흥원 팀장
2017년 3월~현재 호서대학교 컴퓨터공학부 교수
관심분야 : 시스템 보안, 침해사고 대응, Trustworhty AI
E-mail : kinjecs0@gmail.com

논문 2024-1-7 http://dx.doi.org/10.29056/jsav.2024.03.07

머신러닝 드리프트에 대한 2-단계 데이터 품질 평가 방법론

최옥주*, 김유경**†

Two-steps Data Quality Assessment Methodology for Handling Drift of Machine Learning

Okjoo Choi*, Yukyong Kim**†

요 약

빅 데이터 분석이나 머신러닝 모델과 같은 데이터 기반의 정보 기술 분야에서 데이터 품질은 시스템 전체의 품질과 직접적으로 연결된다. 특히 머신러닝 모델의 훈련에 사용된 데이터의 속성은 시간이 지나면서 변화하게 되는데, 이로 인해 모델의 정확도가 떨어지거나 설계된 것과 다르게 작동할 수 있게 된다. 이러한 현상을 드리프트(drift)라고 한다. 드리프트는 데이터 수집 문제나 시장의 변동성 등 다양한 이유로 인해 발생할 수 있다. 데이터 드리프트는 즉시 감지되기 어렵고, 예측이 부정확해 지기 때문에 예측을 기반으로 내린 비즈니스 결정에 어려움을 겪을 수 있다. 드리프트를 관리하기 위해 필요한 작업은 드리프트의 유형이나 범위 및 성격에 따라 달라진다. 적절한 조치를 취하려면 드리프트 식별 뿐만 아니라 데이터 품질 관리 및 평가와 함께 드리프트 비율에 대한 임계값 설정 및 사전 경고 구성을 위한 반복 가능한 절차를 확립하는 것이 중요하다. 본 논문에서는 머신러닝 프로젝트에서 발생하는 드리프트 문제를 데이터의 품질평가 메트릭을 통해 관리할 수 있는 2단계 데이터 품질평가 프레임워크를 제안하고, 드리프트 탐지를 위한 드리프트 유형에 따른 평가 메트릭스와 평가 절차를 정의한다.

Abstract

Data quality of data-based information technologies such as big data analysis and machine learning directly affects the quality of the entire system. In particular, the properties of the data used to train machine learning models change over time, causing the model to become less accurate or behave differently than it was designed to. This phenomenon is called drift. Drift can occur for a variety of reasons, including data collection issues or market volatility. Data drift is difficult to detect immediately and can lead to inaccurate predictions, compromising business decisions based on it. The actions required to manage drift will depend on the type, extent, and nature of the drift. To take appropriate action, it is important to establish repeatable procedures for identifying drift, controlling and assessing data quality, setting thresholds for drift rates, and configuring proactive warnings. In this paper, we propose a two-step data quality assessment framework that can manage drift problems that occur in machine learning projects through data quality assessment indicators. In addition, evaluation indices and evaluation procedures according to drift type for drift detection are also defined.

한글키워드 : 데이터 품질 평가, 데이터 품질 메트릭, 데이터 드리프트, 개념 드리프트

keywords : Data quality assessment, Data quality metric, Data Drift, Concept Drift

* 배재대학교 AI·소프트웨어공학부
** 숙명여자대학교 기초공학부
† 교신저자: 김유경(email: ykim.be@sookmyung.ac.kr)

접수일자: 2024.03.05. 심사완료: 2024.03.16.
게재확정: 2024.03.20.

1. 서 론

데이터 품질은 모든 머신러닝 프로젝트의 성공 요인 중 가장 중요한 요소라고 할 수 있다. 데이터 품질이 좋지 않으면, 모델의 부정확성 문제뿐만 아니라 예측 결과에 대해 신뢰할 수 없게 되는 문제가 발생한다. 이와 함께 머신러닝 모델의 훈련에 사용된 데이터의 속성이 시간이 지나면서 변화하게 되는데, 이로 인해 모델의 정확도가 떨어지거나 설계된 것과 다르게 작동할 수 있게 된다. 이러한 현상을 드리프트(drift)라고 한다. 즉, 머신러닝 모델이 사용되는 환경의 변화로 인해 모델의 정확한 예측 능력이 저하되는 것이다[1].

데이터 드리프트 문제는 시간 경과에 따른 머신러닝 모델의 정확도 변화 문제로, 일반적으로 모델이 훈련에 사용되는 데이터와 상당히 다른 데이터에 대해 추론을 실행하기 때문에 발생한다. 예를 들어, 데이터를 기반으로 회사의 주가를 예측하도록 설계된 머신러닝 모델이 안정적인 시장의 데이터로 훈련되었다면, 처음에는 좋은 결과를 얻을 수 있지만, 시간이 지나면서 시장의 변동성이 커지게 된다면 데이터의 통계적 속성이 변하기 때문에 더 이상 주가를 정확하게 예측하지 못할 수 있다.

드리프트는 데이터 수집 문제나 시장의 변동성 등 다양한 이유로 인해 발생할 수 있다. 데이터 드리프트는 즉시 감지되기 어렵고, 예측이 부정확해 지기 때문에 예측을 기반으로 내린 비즈니스 결정에 어려움을 겪을 수 있다. 드리프트를 관리하기 위해 필요한 작업은 드리프트의 유형이나 범위 및 성격에 따라 달라진다. 어떤 상황에서는 새 데이터를 사용하여 모델을 다시 학습하여 드리프트를 관리할 수 있지만, 다른 경우에는 처음부터 다시 시작해야 할 수도 있다. 머신러닝 모델의 예측이 더 이상 도움이 되지 않을 때 드리프트 문제가 발생하는 것은 단순히 데이터 뿐만 아니라 모델 자체의 문제일 수도 있다. 모델은 변화하는 상황을 인식하지 못하기 때문이다. 적절한 조치를 취하려면 드리프트 식별 뿐만 아니라 데이터 품질 관리 및 평가와 함께 드리프트 비율에 대한 임계값 설정 및 사전 경고 구성을 위한 반복 가능한 절차를 확립하는 것이 중요하다.

따라서 본 논문에서는 머신러닝 프로젝트에서 발생하는 드리프트 문제를 데이터의 품질평가 메트릭을 통해 해결해 보고자 한다. 이를 위해 2단계 데이터 품질평가 프레임워크를 제안하고, 드리프트 탐지를 위한 평가 메트릭스와 평가 절차를 정의한다.

2. 관련 연구

2.1 드리프트 개념과 유형

드리프트는 크게 개념 드리프트(Concept drift)와 데이터 드리프트(Data drift)로 나뉜다. 먼저 모델 드리프트(Model drift)라고도 부르는 개념 드리프트는 머신러닝 모델이 수행하도록 설계된 작업이 시간이 지남에 따라 변경될 때 발생한다. 예를 들어, 이메일 내용을 기반으로 스팸을 탐지하도록 머신러닝 모델을 훈련했다고 하면, 사람들이 받는 스팸메일의 유형이 크게 변경되면 머신러닝 모델은 더 이상 스팸을 정확하게 감지할 수 있다. 개념 드리프트는 다음 그림 1과 같이 크게 4가지 범주로 나눌 수 있다[1].

모델의 성능이 저하되는지 확인하기 위해서 정확도, 오류율, 또는 클릭율과 같은 KPI(Key Performance Indicator)메트릭을 활용할 수 있다. 입력되는 데이터가 크게 변하지 않는 일부 비전이나 언어 모델들은 재학습이 없어도 몇 년 동안 지속될 수도 있다. 하지만 유입되는 새로운 데이

터가 매번 동일하다는 보장은 없다. 코로나와 같이 예측하기 어려운 외부 요인이 발생하면 아무리 안정적으로 여겨진 데이터라도 변할 수 있다.

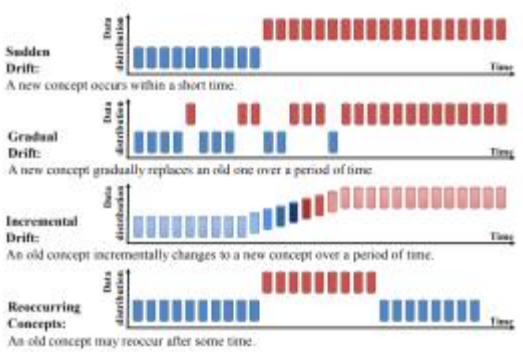


그림 1. 개념 드리프트의 종류
Fig. 1. Types of concept drift
[Source: https://arxiv.org/pdf/2004.05785.pdf]

데이터 드리프트는 공변량 이동(covariate shift)이라고도 한다. 입력 데이터의 분포가 시간에 따라 변할 때 발생하는데, 예를 들면, 연령과 소득을 기준으로 고객이 제품을 구매할 가능성을 예측하도록 훈련된 머신러닝 모델에서, 시간이 지남에 따라 고객의 연령 및 소득 분포가 크게 변하는 경우 모델은 더 이상 구매 가능성을 정확하게 예측하지 못할 수 있게 된다[1].

과거의 여러 연구에서는 지속적으로 도착하는 스트림 데이터의 맥락에서 데이터 드리프트 문제를 조사하고 이를 해결하기 위한 다양한 솔루션을 제안했다[2-5].

특성 x를 사용하여 레이블 y를 예측하는 지도 문제(supervised problem)에서 데이터 드리프트의 주요 원인은 공변량 이동이나 기능 x와 레이블 y 간의 기본 관계가 변경되는 경우로 정의한다[3]. 공변량 이동은 특성 값 x 분포의 변화를 말한다. 데이터 드리프트 문제에 대한 기존 솔루션은 창기반 방법, 이동감지 방법, 그리고 앙상블 기반 방법으로 광범위하게 분류할 수 있다. 새 모델을 훈련하기 위해 기존 데이터에 대해 슬라이딩 창을 사용하는것이 창 기반 방법이다[6]. 이동 감지 방법은 통계 테스트를 사용하여 데이터 드리프트를 감지하고 데이터 드리프트가 발생했음을 감지한 경우에만 모델을 재교육한다[7]. 앙상블 기반 방법은 이전 훈련 데이터에 대한 모델 앙상블을 훈련하고 예측의 가중 평균을 취한다[8].

드리프트의 감지(Detect) 방법은 머신러닝 모델 기반 접근과 통계적 접근방식으로 크게 구분해 볼 수 있다. 머신러닝 모델 기반 접근 방식은 들어오는 입력 데이터의 표류 여부를 감지하는 모니터링 방식이며, 통계적 접근방식은 두 확률 분포 간의 차이를 계산하여 드리프트를 탐지하는 방식으로, PSI(Population Stability Index), KL Divergence, JS Divergence, KS 테스트 및 Wasserstein Metric 등이 있다[9].

[9]에서는 머신러닝 모델의 데이터 드리프트를 감지하는 오픈 소스 활성 학습 도구 키트인 Encord Active를 사용하여 데이터 드리프트 감지하는 방법에 대해 소개하고 있다. [10]에서는 데이터 스트림의 분포가 다양한 시계열 데이터에 대한 다양한 드리프트 감지 방법을 비교 및 평가하고, 성능을 유지하기 위한 즉각적인 경고와 적절한 조치를 통해 시계열 모델의 실시간 분석에 적합한 워크플로우를 제안한다. [11]에서는 머신러닝 모델의 시계열 데이터를 위한 데이터 선정 메커니즘을 제안했던 이전 연구를 적용한 합성 데이터를 이용하여 적대적 분류기(Adversarial classifier)를 이용한 데이터 선택의 효과를 확인한다.

이외에도 [12]에서는 개념 드리프트 감지기 구성에 대한 기존 기술을 비교 분석하였고, [13]에서는 기존의 데이터 품질 관련 연구들을 비교 분석하고, 그 결과로 빅 데이터의 데이터 셋에 대한 라이프사이클을 정의하였다. [14]에서는 클라

우드 제공업체의 두 가지 실제 배포를 연구하여 소프트웨어 제품에서 데이터 드리프트가 미치는 영향을 기술하고 있다. 이 연구에서는 데이터 드리프트 문제에 대한 기존 솔루션은 확장성, 실측 대기 시간, 혼합 유형의 데이터 드리프트와 같은 실제 문제를 해결하도록 설계되지 않은 점을 지적하며, 데이터 드리프트 문제에 대한 확장 가능한 솔루션을 제안하고 있다. Pan의 연구에서는 이러한 현상에 대처하기 위해 정기적으로 모델을 업데이트하고, 데이터 드리프트를 유발하는 특징을 찾아 효과적으로 학습하고 모델 성능을 향상시키고자 하였다[15].

2.2 드리프트 감지 기술

일반적으로 드리프트 탐지기(Drift detector)는 학습모델의 예측 결과를 분석하고 특정 결정 모델을 적용하여 데이터 분포의 변화를 검출한다. 즉, $(\overrightarrow{x_i}, y_i)$ 형태의 샘플들이 제공되는 경우, 여기서 $\overrightarrow{x_i}$는 속성 벡터이고, y_i는 그에 대응되는 부류(class)라고 하자. 각 샘플에 대해 학습모델은 예측값($\hat{y_i}$)을 생성한다. 실제 결과(y_i)와 비교하여 예측이 올바른지($\hat{y_i} = y_i$) 아닌지($\hat{y_i} \neq y_i$) 결정한다. 이 접근 방식을 따르는 가장 잘 알려진 방법은 DDM[16], EDDM[17] 및 STEPD[18] 이다.

드리프트 탐지 방법들은 다양한 전략이나 통계를 사용하여 학습모델의 성능을 모니터링하고 개념 드리프트가 언제 발생하는지를 결정한다. 일반적으로 특정 수준보다 더 낮은 신뢰 수준이 나타나는 경우 경고를 발생시키고, 이런 경고는 개념 드리프트가 발생했음을 의미하게 된다. 여러 방법론들은 이러한 지점에서 머신러닝의 새 인스턴스를 생성하여 병렬로 훈련이 시작되도록 규정한다. 결국 개념 드리프트가 확인되면, 새로운 모델이 원래 모델을 대체하게 된다.

DDM(Drift Detection Method)은 오류율과 해당 표준 편차를 분석하여 스트림의 개념 드리프트를 감지한다. 각 위치 i에 대해 잘못된 예측을 할 확률인 오류율 p_i와 표준 편차 $s_i = \sqrt{p_i \times (1-p_i)/i}$를 정의한다. 샘플의 분포가 고정적으로 유지된다면 샘플 i의 수가 증가함에 따라 오류율 p_i가 감소해야 하므로, 오류율이 증가하면 데이터 분포에 변경이 있었고, 결과적으로 현재 기본 학습자가 오래되었다고 판단하게 된다.

EDDM(Early Drift Detection Method)은 DDM과 유사하지만 오류율이 아닌 두 개의 연속된 오류 사이의 거리를 모니터링한다. 따라서 개념이 고정되면 오류 사이의 거리가 증가하는 경향이 있으며, 감소하면 경고와 드리프트가 트리거 된다. EDDM은 점진적인 개념 드리프트를 탐지하는 데 적합하고, 갑작스러운 개념 드리프트 디텍션에는 DDM이 더 적합하다고 알려져 있다.

STEPD(Statistical test of equal proportions Detection)은 처리된 데이터들을 최근(recent)과 이전(older)으로 명명된 윈도우를 정의하여, 두 윈도우에 대해 계산된 동일한 비율의 통계 테스트를 연속성 수정과 함께 사용하는 방식이다. 이 두 윈도우에 대한 머신러닝의 정확도는 동일하다고 예상할 수 있다. 최근 윈도우의 정확도에서 상당한 차이가 감지되면 경고 및 드리프트 신호가 표시된다.

ADWIN(Adaptive Windowing)은 가변 크기의 인스턴스 슬라이딩 윈도우(W)를 사용한다. 드리프트가 감지되면 W의 크기가 줄어들고 개념이 길어질수록 W의 크기가 커진다. 동적으로 조정된 두 개의 하위 윈도우가 저장되어 이전 데이터와 최신 데이터를 나타내고, 이들 하위 윈도우의 평균 차이가 지정된 임계값 보다 높을 때 드리프트를 감지하게 된다[19].

PHT(Page-Hinkley Test)는 개념 드리프트 감지에도 사용되는 순차 분석 기술이다. 관찰된 값인 기본 학습자의 실제 정확도와 처리된 인스턴스까지의 평균을 계산한다. 드리프트가 발생하면 분류기가 새 인스턴스를 올바르게 분류하지 못하기 시작하여 현재 정확도와 평균 정확도가 감소하게 된다. 이 두 값 사이의 누적(U_T) 및 최소 차이(m_T)가 계산된다. U_T 값이 높을수록 관측된 값이 이전 값과 상당히 다르다는 것을 의미한다. $U_T - m_T$가 허용된 변화 크기인 지정된 임계값을 초과하면 드리프트가 감지된다. 임계값이 높을수록 거짓양성은 줄어 들지만 거짓음성이 많아지고 일부 탐지가 지연될 수 있다[20].

기존의 드리프트 탐지 연구는 주로 개념 드리프트에 관한 것으로 특히 시계열 데이터와 데이터스트림 등 특정 데이터셋에 대한 연구에 한정되어 있다. 그러나 데이터 드리프트는 시간에 따른 입력 데이터 분포의 변화를 의미하는 반면, 개념 드리프트는 모델이 예측하려는 입력 데이터와 출력 변수 간의 관계 변화를 나타내며, 머신러닝 모델을 다룰 때 이해해야 하는 다소 다른 현상이라고 할 수 있다. 또한, 데이터 드리프트는 데이터 검색과 데이터 모델링, 통계 계산을 위한 지표 마련 등이 포함되는 프로세스를 통해 이루어져야 한다[1]. 따라서 데이터 드리프트를 조기에 감지하고 적시에 개입해서 머신러닝 모델의 정확한 예측 제공이 가능하도록 반복 가능한 절차를 통한 통합적이고 체계적인 접근방법이 필요하다.

3. 2-단계 데이터 품질 평가 방법론

본 연구에서는 머신러닝 모델에서 활용하는 데이터 품질 평가 메트릭을 사용한 2-단계 데이터 품질 평가 방법론을 제안한다. 2-단계 품질평가는 머신러닝 모델을 학습하기 전에 좋은 데이터(Good data)를 얻기 위한 데이터 품질 평가와 머신러닝 모델 학습 이후에 시간이 지남에 따라 드리프트 현상 발견을 위한 데이터 품질 평가를 수행한다. 본 연구에서 제안하는 방법론은 다음의 주요 기능으로 구성되어 있다.

(1) DQM Selection (Data Quality Metric) : G_DQM & D-DQM
(2) One-Step: G-DQA
(3) Two-Step: D-DQA

머신러닝 시스템은 목적에 맞은 데이터를 활용하여 머신러닝 모델을 학습하여 구축한 후에도 데이터 변형이나 목적에 맞는 데이터가 유입되는지 지속적인 모니터링을 통해 모델의 성능 저하가 발견되는 드리프트 현상이 발생하는지 주기적으로 확인해야 한다. 머신러닝 모델의 드리프트 현상이 발생했다는 것은 데이터의 예측 분석 결과의 정확성이 떨어진다는 것을 의미한다.

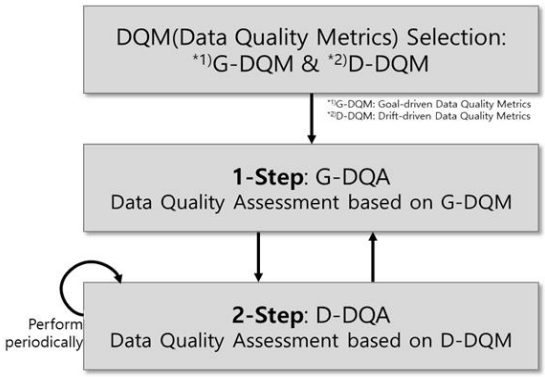


그림 2. 2-단계 데이터 품질평가
Fig. 2. Two-Steps Data quality assessment

따라서 본 연구에서 제안하는 데이터 품질 평가의 첫 번째 단계는 머신러닝 모델 구축 목표에 맞는 목적 기반 메트릭(G-DQM: Goal-driven Data Quality Metric)을 활용하여 데이터의 품질을 평가한다. 두 번째 단계는 머신러닝 모델 학

습 이후 시간이 지나면서 머신러닝 모델의 성능이 저하되는 드리프트 현상을 발견하기 위해 D-DQM (Drift-driven Data Quality Metric)를 활용하여 데이터의 품질을 평가하고 드리프트 유형을 확인한다. 두 번째 단계의 데이터 품질 평가는 주기적으로 수행하고 드리프트 유형에 따라 G-DQM 선정부터 다시 할지 첫 번째 단계를 반복할지 결정한다. 본 연구는 기존 연구 [21]에서 제안한 데이터 품질 평가 프레임워크를 확장하여, 그림 2와 같은 2단계 데이터 품질 평가 방법론을 제안한다.

3.1 데이터 품질 메트릭 선정

본 연구에서는 데이터 품질 메트릭을 선정하기 위해 Victor Basili가 제안한 SW 품질메트릭 선정 방법인 GQM(Goal Question Metric)을 도입하였다. 다수의 문헌과 표준에서 제시한 다양한 데이터 품질 메트릭을 목적이 다른 다양한 도메인의 머신러닝 모델에서 데이터 품질을 평가하는데 활용하는 것은 적절하지 않다. 따라서 본 연구에서는 기존에 제시된 수많은 메트릭 중에서 머신러닝 프로젝트에서 요구하는 목적에 맞게 메트릭을 선정하기 위하여 GQM 방법을 도입하였다.

3.1.1 G-DQM (Goal-Driven Data Quality Metric)

G-DQM은 머신러닝 시스템을 구축하고자 하는 목표를 설정하고 각 목표를 달성하기 위한 질문과 질문을 해결하는 단계로 메트릭을 선정한다.

G-DQM은 프로젝트 성공을 위해 달성하고자 하는 것은 무엇인지, 이를 위해 필요한 것은 무엇인지를 고려하고 빅 데이터 특성 및 머신러닝 모델을 적용하는 분야의 비즈니스 요구사항을 고려하여 목표를 설정한다.

각 목표에 대한 질문은 1:1이 될 수도 있고 1:N이 될 수도 있다. 또한 하나의 질문은 두 개의 목표를 달성할 수 있다. 따라서 목표-질문-메트릭은 다음 그림 3과 같은 관계로 표현할 수 있다.

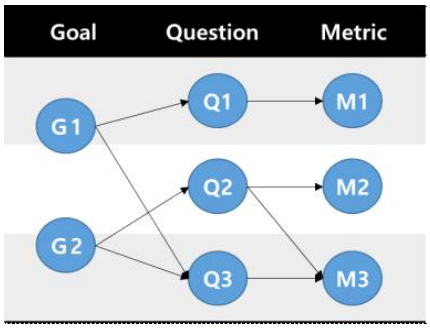


그림 3. 목표-질문-메트릭
Fig. 3. Goal-Question-Metric(GQM)

또한, 경우에 따라 목표 달성에 해당하는 메트릭을 직접적으로 매핑 할 수도 있다. 목표-(질문)-메트릭 선정 과정은 머신러닝 모델을 구축하는 목적과 용도에 적합한 메트릭을 도출할 때까지 과정을 반복한다.

메트릭을 선정하면 프로젝트에서 요구하는 품질 기준(Baseline)도 결정한다. 품질 기준은 데이터를 활용하는 목표 및 도메인에 따라 다르게 지정할 수 있다. 예를 들어, 표 1과 같이 목표와 메트릭이 정의되었다면, 국방이나 의료 등 안전성(Safety)이 매우 중요한 분야의 Accuracy 기준은 0.95로 지정할 수 있으나 다른 도메인에서는 0.9로 지정할 수도 있다.

표 1. 목표-메트릭 예시
Table 1. Goal-Metric example

Goal	Metric	Expression
데이터가 정확해야 함	Accuracy	$\frac{\text{(1-Number of data with errors)}}{\text{Total number of data}}$
중복 최소화	Redundancy	$\frac{\text{Number of duplicate data}}{\text{Total number of data}}$

3.1.2 D-DQM (Drift-Driven Data Quality Metric)

머신러닝 모델을 구축한 이후에는 그림 4와 같이 주기적으로 데이터 셋이 추가되므로 데이터 품질 평가도 주기적으로 수행해야 한다.

특히, 머신러닝 시스템을 운영하는 시간이 경과하면서 모델을 학습한 당시와 입력 데이터의 유형이 달라지거나 데이터와 모델의 해석 방법이 달라지는 경우 모델의 성능이 저하되어 데이터 예측의 정확도가 저하되는 드리프트 현상이 발생한다. 드리프트가 발생하면 머신러닝 모델은 새로 입력된 데이터를 사용하여 모델을 다시 학습시켜야 한다. 이렇게 모델을 업데이트해야 목적에 맞게 성능 좋은 머신러닝 시스템을 지속적으로 운영할 수 있다.

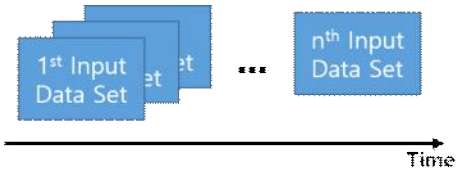


그림 4. 시간경과에 따른 데이터 셋 추가
Fig. 4. Adding data sets over time

본 연구에서는 다음 표 2와 같이 드리프트 유형에 따라 메트릭을 선정하여 주기적으로 D-DQM 기반 데이터 품질 평가를 수행한다.

표 2. 드리프트 유형별 메트릭
Table 2. Metrics by drift type

드리프트 유형	메트릭
Concept Drift(CD): Concept Drift-DQM	- Accuracy
Data Drift(DD): Data Drift-DQM	- Consistency - Completeness

개념 드리프트는 모델의 정확도를 평가하고 데이터 드리프트는 입력 데이터 패턴의 일관성 또는 완전성을 평가한다. 드리프트 메트릭은 현재 입력된 데이터를 평가하고 이전에 입력한 데이터를 비교하여 기준을 벗어나는 경우 드리프트 현상이 발생하였다고 간주한다.

3.2 2-단계 데이터 품질 평가

머신러닝 모델의 데이터 품질평가는 모델 구축 및 운영하는 전 과정에서 주기적으로 수행되어야 머신러닝 시스템의 품질이 보증된다.

본 연구에서는 기존에 저자가 제안한 목적 기반 메트릭(G-DQM)을 사용한 데이터 품질 평가 방법[21]과 드리프트 기반 메트릭(D-DQM)을 활용한 데이터 품질 평가 방법을 추가하여 2-단계 데이터 품질 평가 방법론을 제안한다. 그림 5에서 보는 것과 같이, 1-Step 데이터 품질 평가는 머신러닝 모델 학습 전에 데이터 자체를 평가하는 단계이고 2-Step 데이터 품질 평가는 머신러닝 모델 학습 이후 모델과 데이터의 관계, 현재 데이터와 이전 데이터와의 관계를 평가하는 단계이다.

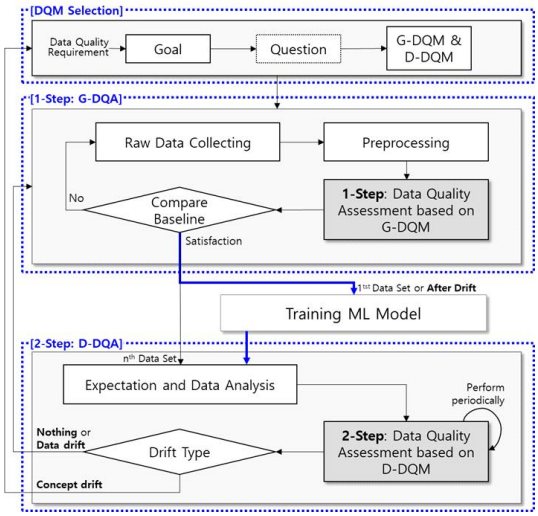


그림 5. 2-단계 데이터 품질 평가
Fig. 5. Two-Steps data quality assessment process

3.2.1 G-DQA: 목적기반 데이터 품질 평가

머신러닝 모델의 첫 번째 단계의 데이터 품질 평가는 최초의 머신러닝 모델을 구축하는 시점이나 머신러닝 모델을 운영하면서 개념 드리프트가 발생하는 시점에서 수행한다.

G-DQA(G-DQM based Data Quality Assessment)는 목적 기반 메트릭(G-DQM)을 사용하여 데이터 품질을 평가한다. G-DQM은 하나의 머신러닝 시스템에 여러 개를 선정할 수 있으며 G-DQM 계산식을 사용하여 데이터를 평가한 값이 품질 기준을 만족하면 다음 단계를 진행한다. G-DQA의 알고리즘이 표3에 기술되었다.

표 3. One-Step G-DQA 알고리즘
Table 3. One-Step G-DQA algorithm

Algorithm G-DQA(i, $G-DQM_i$, $Baseline_i$) :
//i: number of quality metrics
//$G-DQM_i$: metric expression
//$Baseline_i$: threshold value

for each $G-DQM_i$ where $1 \le i \le n$ **do**
  $A_i \leftarrow$ calculated $G-DQM_i$ value
  if $A_i > Baseline_i$ **then**
    print "No Satisfaction"
    Raw Data Collecting
  endif
endfor

if 1st Data set **or** After Drift **then**
  Training ML Model
else
  Data Analysis using ML Model
endif

즉, G-DQM 수식을 사용하여 데이터 품질 평가한 값 A_i은 품질 메트릭 개수(n) 만큼 산출된다. 품질 기준 $Baseline_i$과 비교하여 품질 평가 값 A_i가 $Baseline_i$ 보다 작으면, 품질 평가 값이 기준을 만족하여 데이터 품질이 확보된 것으로 간주한다. 여기에서 i는 1부터 메트릭 개수 n까지 나타낸다. 예를 들어, accuracy와 consistency의 품질 값은 각각 $A_{accuracy}$와 $A_{consistency}$이 되고, 품질 기준은 $Baseline_{accuracy}$와 $Baseline_{consistency}$로 표기한다.

데이터 품질 평가는 선정한 모든 메트릭의 평가 값이 각 품질 기준을 만족해야만 머신러닝 모델을 학습하거나 머신러닝 모델을 사용하여 예측 분석을 수행한다. 그러나 한 개라도 메트릭 평가가 품질 기준을 만족하지 못하면 새로운 데이터를 수집하고 데이터 평가를 다시 수행한다.

3.2.2 D-DQA: 드리프트기반 데이터 품질 평가

두 번째 단계의 D-DQA(D-DQM based Data Quality Assessment)는 드리프트 기반 메트릭(D- DQM)을 사용하여 데이터 품질을 평가한다.

머신러닝 시스템에서는 대량의 데이터가 주기적으로 데이터가 입력되므로 D-DQA도 주기적으로 드리프트 유형 별로 수행한다.

본 연구에서 드리프트 감지를 위해서는 드리프트 유형 별 D-DQM을 사용한 데이터를 평가한다. (n-1)번째 데이터 셋의 데이터를 평가 값과 n번째 데이터 셋의 데이터 평가 값을 비교한다. 비교한 값이 각 드리프트 메트릭의 DT(Drift Threshold)를 만족하지 못하면 드리프트 현상이 발견된 것이라 간주하고 머신러닝 모델을 새로 학습시킨다.

DT는 기존 데이터 셋과 신규로 입력된 데이터 셋의 유형 변경을 허용하는 범위이며 드리프트 메트릭마다 다르게 지정할 수 있다. 표 4의 알고리즘에 기술된 것과 같이, 예를 들어, 개념 드리프트의 메트릭을 accuracy로 선정하였다면 (n-1)번째 데이터 셋의 accuracy 평가 값 (Acc_{n-1})과 n번째 데이터 셋의 accuracy평가 값 (Acc_n)을 비교한다. 비교 값이 $DT_{Accuracy}$범위를

벗어나면 개념 드리프트가 발생한 것으로 간주한다. 데이터 드리프트도 마찬가지로 (n-1)번째 데이터 셋과 n번째 데이터 셋의 데이터 드리프트 메트릭 consistency 평가 값을 각각 산출하고, $DT_{Consistency}$범위 내에 있는지 확인한다.

표 4. Two-Step D-DQA 알고리즘
Table 4. Two-Step D-DQA algorithm

Algorithm D-DQA(i, DT_i) :
//$i \in (Acc, Con, Cmp)$: drift metrics
//Acc : Accuracy, Con : Consistency,
//Cmp : Completeness
// i_n : metric value for n-th data set
//DT_i: threshold value for each i

if $(Acc_{n-1} - Acc_n) > \pm DT_{Acc}$ **then**
    print "Concept Drift"
    Selection of G-DQM
else if $(Con_{n-1} - Con_n) > \pm DT_{Con}$ **or**
        $(Cmp_{n-1} - Cmp_n) > \pm DT_{Cmp}$ **then**
    print "Data Drift"
    Training ML Model in 1-Step G-DQA
endif

DT는 시스템 초기에는 임의로 정할 수 있으나 시간이 지남에 따라 평가 값이 수집이 되면 통계 시그마(σ)값을 활용하여 DT를 조정한다. 예를 들어, DT를 6-σ로 선정하는 경우 3-σ로 선정하는 시스템 보다 더 높은 데이터 품질을 요구하므로 구축하는 목적이나 도메인에 따라 DT를 결정한다.

4. 결론 및 향후 연구과제

데이터 기반 정보기술 분야에서 데이터 품질은 프로젝트의 성공 요인 중 가장 중요한 요소라고 할 수 있다. 특히 머신러닝 모델의 훈련에 사용된 데이터의 속성은 시간이 지나면서 변화하게 되는데, 이로 인해 모델의 정확도가 떨어지거나 설계된 것과 다르게 작동할 수 있게 되는 드리프트 현상이 큰 문제가 된다. 데이터 드리프트 문제는 시간 경과에 따른 머신러닝 모델의 정확도 변화 문제로, 일반적으로 모델이 훈련에 사용되는 데이터와 상당히 다른 데이터에 대해 추론을 실행하기 때문에 발생한다. 데이터 드리프트는 즉시 감지되기 어렵고, 예측이 부정확해 지기 때문에 예측을 기반으로 내린 비즈니스 결정에 어려움을 겪을 수 있다. 드리프트를 관리하기 위해 필요한 작업은 드리프트의 유형이나 범위 및 성격에 따라 달라진다. 머신러닝 모델은 변화하는 상황을 인식하지 못하기 때문에, 드리프트 문제가 발생하는 것은 단순히 데이터 뿐만 아니라 모델 자체의 문제일 수도 있다. 적절한 조치를 취하려면 드리프트 식별 뿐만 아니라 데이터 품질 관리 및 평가와 함께 드리프트 비율에 대한 임계값 설정 및 사전 경고 구성을 위한 반복 가능한 절차를 확립하는 것이 중요하다.

본 논문에서는 머신러닝 프로젝트에서 발생하는 드리프트 문제를 데이터의 품질평가 메트릭을 통해 해결해 보고자 하였다. 이를 위해 2단계 데이터 품질평가 프레임워크를 제안하고, 드리프트 탐지를 위한 평가 메트릭스와 평가 절차를 정의한다. 본 연구에서 제안하는 데이터 품질 평가의 첫 번째 단계는 머신러닝 모델 구축 목표에 맞는 목적 기반 메트릭(G-DQM)을 활용하여 데이터의 품질을 평가한다. 두 번째 단계는 머신러닝 모델 학습 이후 시간이 지나면서 머신러닝 모델의 성능의 저하되는 드리프트 현상을 발견하고 드리프트 기반 메트릭(D-DQM)을 활용하여 데이터의 품질을 평가하고 드리프트 유형을 확인한다. 두 번째 단계의 데이터 품질평가는 주기적으로 수행하고 드리프트 유형에 따라 G-DQM 선정부터 다시 할지 첫 번째 단계를 반복할지 결정

하게 된다.

본 논문에서 정의한 2단계 데이터 품질 평가 프레임워크는 데이터 드리프트의 문제를 데이터 품질관리 및 평가의 차원에서 반복가능한 절차를 통해 접근해 보았다. 현재까지 데이터 드리프트 문제에 대한 모범사례(best-practice)가 없는 상황에서, 지속적인 모니터링 체계 확립을 통해 이를 해결해 보고자 하였다.

향후 제안된 프레임워크 구현을 통해, 실제 데이터셋의 품질 관리 사례를 마련하는 것이 필요하며, 기존 연구들과의 비교 평가를 통한 유효성 검증이 필요하다.

이 논문은 정부(과학기술정보통신부)의
재원으로 정보통신기획평가원의 지원을 받아
수행된 지역지능화혁신인재양성사업임
(IITP-2024- RS-2022-00156334)

참 고 문 헌

[1] M. Ali, “Understanding Data Drift and Model Drift : Drift Detection in Python”, Founder&Creator of PyCaret, Jan. 2023, https://www.datacamp.com/tutorial/understanding-data-drift-model-drift

[2] A. Bifet, R. Gavaldà, “Learning from time-changing data with adaptive windowing”, In Proceedings of the SIAM International Conference on Data Mining (ICDM), pp. 443-448, 2007, DOI: https://doi.org/10.1137/1.9781611972771.42

[3] J. G. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N. V. Chawla, F. Herrera, “A unifying view on dataset shift in classification”, Pattern recognition, 45(1), pp. 521-530, 2012, DOI: https://doi.org/10.1016/j.patcog.2011.06.019

[4] A. Suprem, J. Arulraj, C. Pu, J. Ferreira, “ODIN: Automated drift detection and recovery in video analytics”, In Proceedings of the VLDB Endowent, 13(12), pp. 2453-2465, July 2020, DOI: https://doi.org/10.48550/arXiv.2009.05440

[5] A. Tahmasbi, E. Jothimurugesan, S. Tirthapura, P. B. Gibbons, “Driftsurf: A risk-competitive learning algorithm under concept drift”, ArXiv journal, 2020, DOI : https://doi.org/10.48550/arXiv.2003.06508

[6] G. Widmer, M. Kubat, “Learning in the presence of concept drift and hidden contexts”, Machine learning, 23(1), pp. 69-101, 1996, DOI: https://doi.org/10.1007/BF00116900

[7] A. Pesaranghader, H. L. Viktor, E. Paquet, “Mcdiarmid drift detection methods for evolving data streams”, In Proceedings of the International Joint Conference on Neural Networks, pp.1-9, 2018, DOI: https://doi.org/10.1109/IJCNN.2018.8489260

[8] D. Brzezinski, J. Stefanowski, “Reacting to different types of concept drift: The accuracy updated ensemble algorithm”, IEEE Transactions on Neural Networks and Learning Systems, 25(1), pp. 81-94, 2014, https://doi.org/10.1109/TNNLS.2013.2251352

[9] A. Acharya, “How to Detect Data Drift on Datasets”, August 9, 2023, available at https://encord.com/blog/detect-data-drift/

[10] S. Ashok, S. Ezhumalai, T. Patwa, “Remediating data drifts and re-establishing ML models”, In Proceedings of International Conference on Machine Learning and Data Engineering, 218, pp. 799-809, 2023, DOI: https://doi.org/10.1016/j.procs.2023.01.060

[11] Y. Konno, M. Nakano, M. Oguchi, “Efficient Data Selection Indicators for Updating Models under Data Drifted Environment”, In Proceedings of International Conference on Big Data, pp. 6724-6726, 2022, DOI: https://doi.org/

10.1109/BigData55660.2022.10020630
[12] R. S. Barros, S. Garrido T. Carvalho Santos, “A large-scale comparison of concept drift detectors”, Information Science, vol.451-452, pp.348-370, 2018, DOI: https://doi.org/10.1016/j.ins.2018.04.014
[13] Y. Gong, G. Liu, Y. Xue, R. Li, L. Meng, “A Survey on dataset quality in machine learning”, Infromation and software technology, 162(107268), 2023, DOI: https://doi.org/10.1016/j.infsof.2023.107268
[14] A. Mallick, K. Hsieh, B. Arzani, G. Joshi, “Matchmaker: Data Drift mitigation in machine learning for large-scale systems”, In Proceedings of the Machine Learning and Systems, pp. 77-94, 2022,
[15] J. Pan, V. Pham, M. Dorairaj, H. Chen, J. Lee, “Adversarial validation approach to concept drift problem in automated machine learning systems”, journal of ArXiv, 2020, https://doi.org/10.48550/arXiv.2004.03045
[16] J. Gama , P. Medas , G. Castillo , P. Rodrigues, “Learning with drift detection”, LNAI, vol. 3171, pp. 286-295, 2004, DOI: https://doi.org/10.1007/978-3-540-28645-5_29
[17] M. Baena-Garcia, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavald, R. Morales-Bueno, “Early drift detection method”, In Proceedings of the International Workshop on Knowledge Discovery from Data Streams, pp. 77-86, 2006,
[18] K. Nishida, K. Yamauchi, “Detecting concept drift using statistical testing”, LNCS, vol. 4755, pp. 264-269, 2007, DOI: https://doi.org/10.1007/978-3-540-75488-6_27
[19] A. Bifet , R. Gavaldà, “Learning from time-changing data with adaptive windowing”, In Proceedings of SIAM International Conference on Data Mining, pp. 443-448, 2007, DOI: https://doi.org/10.1137/1.9781611972771.42
[20] J. Gama, “Knowledge Discovery from Data Streams”, Chapman & Hall/CRC Data mining and knowledge discovery series, 2010
[21] O. Choi, Y. Kim, “A Survey of Data Quality Assessment Methods for Big Data”, Journal of Software Assessment and Valuation, 19(4), pp. 89-98, 2023, DOI : http://dx.doi.org/10.29056/jsav.2023.12.09

저 자 소 개

최옥주(Okjoo Choi)

2008.2 숙명여자대학교 컴퓨터과학과 박사
1990.8-1996.3 LG생산기술원 주임원구원
1996.7-2009.8 한국오라클 수석컨설턴트
2009.9-2022.12 한국과학기술원 연구교수
2023.10-2024.2 강원대학교 산학협력중점교수
2024.03-현재 배재대학교 조교수
<주관심분야> 빅데이터 분석, 데이터마이닝, 데이터 품질, 소프트웨어 품질, 프로젝트 관리

김유경(Yukyong Kim)

2001.8 숙명여자대학교 컴퓨터과학과 박사
2005.9-2006.8 UC Davis, Post-doc.
2006.9-2013.9 한양대학교 컴퓨터공학과 연구교수
2018.3-현재 숙명여자대학교 기초공학부 교수
<주관심분야> 웹서비스 QoS 평가, SOA 기반 IoT 신뢰 평가, 소프트웨어 품질평가

데이터 드리프트 검출 및 추적 시스템의 설계 및 개발

박종빈, 김경원, 정종진
한국전자기술연구원
jpark@keti.re.kr

Design and Development of Data Drift Detection and Tracking System

Jongbin Park, Kyung-Won Kim, Jong-Jin Jung
Korea Electronics Technology Institute

요 약

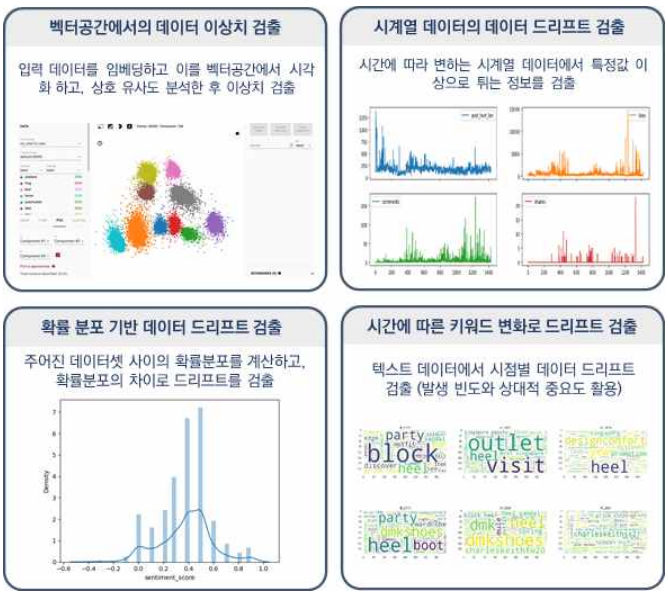
본 논문은 기계학습이나 인공지능 모델을 개발하고 운용 시 발생하는 데이터 드리프트 현상을 검출하고 추적하는 시스템을 설계하고 개발한 내용을 소개한다. 통상적으로 데이터 드리프트 문제는 데이터 간 통계 분포의 불일치, 외부 환경 및 요구사항의 변화 등이 분석 모델의 성능 저하로 이어진다. 구체적인 데이터 드리프트 발생 원인은 다양하며, 서비스나 데이터 특성에 따라 다양한 감시 및 추적 기술이 요구된다. 이러한 데이터 드리프트 문제를 조기에 검출하고, 지속적으로 추적하며 상황에 맞는 해결 방안을 제시하는 것은 날로 중요해지고 있다. 따라서 본 논문에서는 데이터 드리프트 검출 및 추적을 위한 벡터공간에서의 데이터 이상치 검출, 시계열 데이터에서의 드리프트 검출, 확률 분포 함수의 차이를 사용한 드리프트 검출, 텍스트 데이터에서의 시간에 따른 키워드 변화 추적을 통한 드리프트 검출 기술을 설계하고 최소 기능 구현 제품으로 구현한다.

Ⅰ. 서 론

데이터 드리프트 문제는 분석을 위해 사용하는 소정의 모델이 실증 및 현장의 서비스 단계에서 성능이 감소하는 현상으로 해석할 수 있다[1]. 이를 해결하기 위해서 데이터 드리프트 현상이 언제, 어떻게, 얼마나 발생했는지를 파악하는 기술이 중요하며, 여러 가지 데이터 종류(예: 텍스트, 이미지, 비디오)에 대해서도 유기적으로 대응할 수 있어야 한다. 이런 문제를 해결하고자 본 논문에서는 다양한 데이터 종류에 대응하는 데이터 드리프트 현상 검출 및 추적 기술의 구현 내용을 소개한다.

Ⅱ. 본론

그림 1은 개발한 데이터 드리프트 검출 및 추적 기술 관련 요소기술을 나타낸다. 이들은 마이크로 서비스 아키텍처[2] 방식으로 구현하며 통합 프레임워크를 구성한다. 그림 1의 왼쪽 상단은 Cifar10 이미지 데이터를 비전 트랜스포머(ViT, Vision Transformer)로 임베딩 후에 고차원 벡터 정보를 3차원 영역에 사영(projection)한 결과이다. 군집에서 벗어난 데이터를 이상치로 판단하거나 새롭게 입력된 이미지가 기존 군집과 얼마나 떨어져 있는지를 계산할 수도 있다. 그림 1의 오른쪽 상단은 시계열 데이터에서 데이터 변화를 검출하며 푸리에 변환, 웨이블릿 변환 등을 사용하여 데이터 주기성과 이에 따른 급격한 데이터 변화(Spike) 사건을 찾는 예시이다. 그림 1의 왼쪽 하단은 주어진 데이터에서 확률 분포 함수를 계산 후 Earth Mover distance나 Kullback-Leibler divergence를 계산하여 데이터셋 사이의 드리프트를 정량적으로 계산하는 예시이다. 그림 1의 오른쪽 하단은 키워드 분석을 통해 시간에 따른 드리프트 현상을 검출하고 추적하는 기술을 보여준다.



<그림 1> 데이터 드리프트 검출 및 추적 기술 예시

Ⅲ. 결론

본 논문에서는 기계학습 및 인공지능 영역에서 활용할 수 있는 데이터 드리프트 검출 및 추적 기술을 최소 기능 제품으로 구현한 내용을 소개했다.

ACKNOWLEDGMENT

이 논문은 2024년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No. RS-2024-00337489, 분석 모델의 성능저하 극복을 위한 데이터 드리프트 관리 기술 개발).

참 고 문 헌

[1] A. Mallick, K. Hsieh, B. Arzani, G. Joshi, "Matchmaker: Data drift mitigation in machine learning for large-scale systems," Proceedings of Machine Learning and Systems, vol. 4, pp. 77-94, 2022.

[2] N. Alshuqayran, N. Ali, and R. Evans, "A systematic mapping study in microservice architecture," IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA), pp. 44-51, 2016.

Learning under Concept Drift: A Review

Jie Lu, *Fellow, IEEE,* Anjin Liu, *Member, IEEE,* Fan Dong, Feng Gu, João Gama, and Guangquan Zhang

Abstract—Concept drift describes unforeseeable changes in the underlying distribution of streaming data over time. Concept drift research involves the development of methodologies and techniques for drift detection, understanding and adaptation. Data analysis has revealed that machine learning in a concept drift environment will result in poor learning results if the drift is not addressed. To help researchers identify which research topics are significant and how to apply related techniques in data analysis tasks, it is necessary that a high quality, instructive review of current research developments and trends in the concept drift field is conducted. In addition, due to the rapid development of concept drift in recent years, the methodologies of learning under concept drift have become noticeably systematic, unveiling a framework which has not been mentioned in literature. This paper reviews over 130 high quality publications in concept drift related research areas, analyzes up-to-date developments in methodologies and techniques, and establishes a framework of learning under concept drift including three main components: concept drift detection, concept drift understanding, and concept drift adaptation. This paper lists and discusses 10 popular synthetic datasets and 14 publicly available benchmark datasets used for evaluating the performance of learning algorithms aiming at handling concept drift. Also, concept drift related research directions are covered and discussed. By providing state-of-the-art knowledge, this survey will directly support researchers in their understanding of research developments in the field of learning under concept drift.

Index Terms—concept drift, change detection, adaptive learning, data streams

1 Introduction

Governments and companies are generating huge amounts of streaming data and urgently need efficient data analytics and machine learning techniques to support them making predictions and decisions. However, the rapidly changing environment of new products, new markets and new customer behaviors inevitably results in the appearance of concept drift problem. Concept drift means that the statistical properties of the target variable, which the model is trying to predict, change over time in unforeseen ways [1]. If the concept drift occurs, the induced pattern of past data may not be relevant to the new data, leading to poor predictions and decision outcomes. The phenomenon of concept drift has been recognized as the root cause of decreased effectiveness in many data-driven information systems such as data-driven early warning systems and data-driven decision support systems. In an ever-changing and big data environment, how to provide more reliable data-driven predictions and decision facilities has become a crucial issue.

Concept drift problem exists in many real-world situations. An example can be seen in the changes of behavior in mobile phone usage, as shown in Fig. 1. From the bars in this figure, the time percentage distribution of the mobile phone usage pattern has changed from "Audio Call" to "Camera" and then to "Mobile Internet" over the past two decades.

Recent attractive research in the field of concept drift targets more challenging problems, i.e., how to accurately detect concept drift in unstructured and noisy datasets [2], [3], how to quantitatively understand concept drift in a explainable way [4], [5], and how to effectively react to drift by adapting related knowledge [6], [7].

Solving these challenges endows prediction and decision-making with the adaptability in an uncertain environment. Conventional research related to machine learning has been significantly improved by introducing concept drift techniques in data science and artificial intelligence in general, and in pattern recognition and data stream mining in particular. These new studies enhance the effectiveness of analogical and knowledge reasoning in an ever-changing environment. A new topic is formed during this development: adaptive data-driven prediction/decision systems. In particular, concept drift is a highly prominent and significant issue in the context of the big data era because the uncertainty of data types and data distribution is an inherent nature of big data.

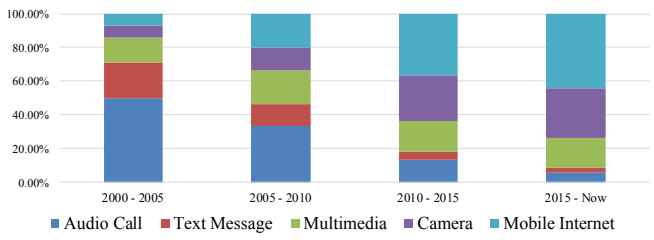


Fig. 1. Concept drift in mobile phone usage (data used in figure are for demonstration only)

Conventional machine learning has two main components: training/learning and prediction. Research on learning under concept drift presents three new components: drift detection (whether or not drift occurs), drift understanding (when, how, where it occurs) and drift adaptation (reaction to the existence of drift) as shown in Fig. 2. These will be discussed in Section 3-5.

In literature, a detailed concept drift survey paper [8] was published in 2014 but intentionally left certain subproblems of concept drift to other publications, such as the details of the data distribution change ($P(X)$) as mentioned in their Section 2.1. In 2015, another comprehensive survey paper [9] was published, which surveys and gives tutorial of both the established and the state-of-the-art approaches. It provides a hybrid-view about concept drift from two

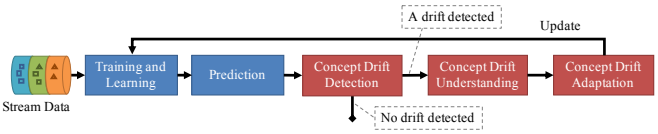


Fig. 2. Framework for handling concept drift in machine learning. Please note that some methods can do concept drift detection and concept drift understanding simultaneously.

primary perspectives, active and passive. Both survey papers are comprehensive and can be a good introduction to concept drift researching. However, many new publications have become available in the last three years, even a new category of drift detection methods has arisen, named multiple hypothesis tests drift detection. It is necessary to review the past research focuses and give the most recent research trends about concept drift, which is one of the main contribution of this survey paper.

Besides these two publications, four related survey papers [6], [7], [10], [11] have also provided valuable insights into how to address concept drift, but their specific research focus is only on data stream learning, rather than analyzing concept drift adaptation algorithms and understanding concept drift. Specifically, paper [7] focuses on data reduction for stream learning incorporating concept drift, while [6] only focuses on investigating the development in learning ensembles for data stream learning in a dynamic environment. [11] concerns the evolution of data stream clustering, and [10] focuses on investigating the current and future trends of data stream learning. There is therefore a gap in the current literature that requires a fuller picture of established and the new emerged research on concept drift; a comprehensive review of the three major aspects of concept drift: concept drift detection, understanding and adaptation, as shown in Fig. 2; and a discussion about the new trend of concept drift research.

The selection of references in this survey paper was performed according to the following steps:

Step 1. Publication database: Science Direct, ACM Digital Library, IEEE Xplore and SpringerLink.

Step 2. Preliminary screening of articles: The first search was based on keywords. The articles were then selected as references if they 1) present new theory, algorithm or methodology in the area of concept drift, or 2) report a concept drift application.

Step 3. Result filtering for articles: The articles selected in Step 2 were divided into three groups: concept drift detection, understanding, and adaptation. The references in each group were filtered again, based on 1) Time: published mainly within the last 10 years, or 2) Impact: published in high quality journals/conferences or having high citations.

Step 4. Dataset selection: To help readers test their research results, this paper lists popular datasets and their characteristics, the dataset providers, and how each dataset can be used.

On completion of this process, 137 research articles, 10 widely used synthetic datasets for evaluating the performance of learning algorithms dealing with concept drift, and 14 publicly available and widely used real-world datasets were listed for discussion.

The main contributions of this paper are:

1) It perceptively summarizes concept drift research achievements and clusters the research into three categories: concept drift detection, understanding and adaptation, providing a clear framework for concept drift research development (Fig. 2);
2) It proposes a new component, concept drift understanding, for retrieving information about the status of concept drift in aspects of when, how, and where. This also creates a connection between drift detection and drift adaptation;
3) It uncovers several very new concept drift techniques, such as active learning under concept drift and fuzzy competence model-based drift detection, and identifies related research involving concept drift;
4) It systematically examines two sets of concept drift datasets, Synthetic datasets and Real-world datasets, through multiple dimensions: dataset description, availability, suitability for type of drift, and existing applications;
5) It suggests several emerging research topics and potential research directions in this area.

The remainder of this paper is structured as follows. In Section 2, the definitions of concept drift are given and discussed. Section 3 presents research methods and algorithms in concept drift detection. Section 4 discusses research developments in concept drift understanding. Research results on drift adaptation (concept drift reaction) are reported in Section 5. Section 6 presents evaluation systems and related datasets used to test concept drift algorithms. Section 7 summaries related research concerning the concept drift problem. Section 8 presents a comprehensive analysis of main findings and future research directions.

2 Problem Description

This section first gives the formal definition and the sources of concept drift in Section 2.1. Then, in Section 2.2, the commonly defined types of concept drift are introduced.

2.1 Concept drift definition and the sources

Concept drift is a phenomenon in which the statistical properties of a target domain change over time in an arbitrary way [3]. It was first proposed by [12] who aimed to point out that noise data may turn to non-noise information at different time. These changes might be caused by changes in hidden variables which cannot be measured directly [4]. Formally, concept drift is defined as follows:

Given a time period $[0, t]$, a set of samples, denoted as $S_{0,t} = \{d_0, \ldots, d_t\}$, where $d_i = (X_i, y_i)$ is one observation (or a data instance), X_i is the feature vector, y_i is the label, and $S_{0,t}$ follows a certain distribution $F_{0,t}(X, y)$. Concept drift occurs at timestamp $t+1$, if $F_{0,t}(X, y) \neq F_{t+1,\infty}(X, y)$, denoted as $\exists t\colon P_t(X, y) \neq P_{t+1}(X, y)$ [2], [8], [13], [14].

Concept drift has also been defined by various authors using alternative names, such as dataset shift [15] or concept shift [1]. Other related terminologies were introduced in [16]'s work, the authors proposed that concept drift or shift is only one subcategory of dataset shift and the dataset shift is consists of covariate shift, prior probability shift and

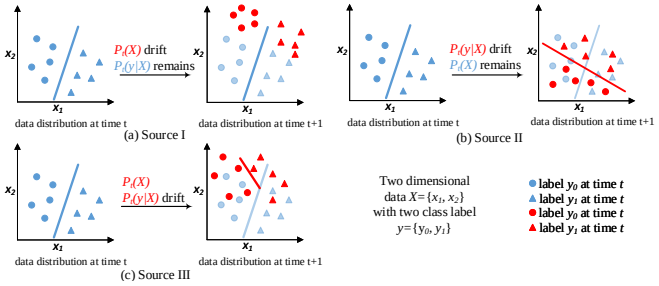


Fig. 3. Three sources of concept drift

concept shift. These definitions clearly stated the research scope of each research topics. However, since concept drift is usually associated with covariate shift and prior probability shift, and an increasing number of publications [2], [8], [13], [14] refer to the term "concept drift" as the problem in which $\exists t$: $P_t(X, y) \neq P_{t+1}(X, y)$. Therefore, we apply the same definition of concept drift in this survey. Accordingly, concept drift at time t can be defined as the change of joint probability of X and y at time t. Since the joint probability $P_t(X, y)$ can be decomposed into two parts as $P_t(X, y) = P_t(X) \times P_t(y|X)$, concept drift can be triggered by three sources:

- **Source I**: $P_t(X) \neq P_{t+1}(X)$ while $P_t(y|X) = P_{t+1}(y|X)$, that is, the research focus is the drift in $P_t(X)$ while $P_t(y|X)$ remains unchanged. Since $P_t(X)$ drift does not affect the decision boundary, it has also been considered as virtual drift [7], Fig. 3(a).
- **Source II**: $P_t(y|X) \neq P_{t+1}(y|X)$ while $P_t(X) = P_{t+1}(X)$ while $P_t(X)$ remains unchanged. This drift will cause decision boundary change and lead to learning accuracy decreasing, which is also called actual drift, Fig. 3(b).
- **Source III**: mixture of Source I and Source II, namely $P_t(X) \neq P_{t+1}(X)$ and $P_t(y|X) \neq P_{t+1}(y|X)$. Concept drift focus on the drift of both $P_t(y|X)$ and $P_t(X)$, since both changes convey important information about learning environment Fig. 3(c).

Fig. 3 demonstrates how these sources differ from each other in a two-dimensional feature space. Source I is feature space drift, and Source II is decision boundary drift. In many real-world applications, Source I and Source II occur together, which creates Source III.

2.2 The types of concept drift

Commonly, concept drift can be distinguished as four types [8] as shown in Fig. 4:

Research into concept drift adaptation in Types 1-3 focuses on how to minimize the drop in accuracy and achieve the fastest recovery rate during the concept transformation process. In contrast, the study of Type 4 drift emphasizes the use of historical concepts, that is, how to find the best matched historical concepts with the shortest time. The new concept may suddenly, incrementally, or gradually reoccur.

To better demonstrate the differences between these types, the term "intermediate concept" was introduced by [8] to describe the transformation between concepts. As mentioned by [4], a concept drift may not only take place

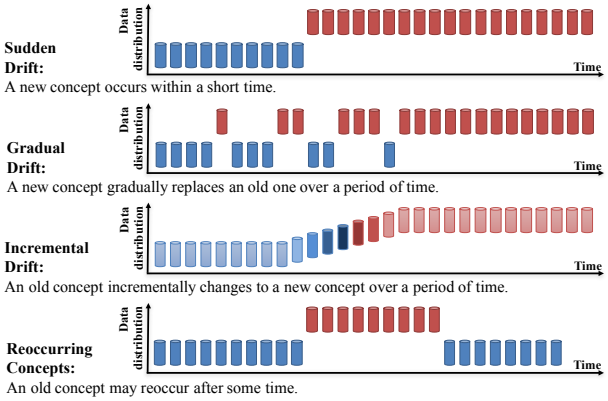


Fig. 4. An example of concept drift types.

at an exact timestamp, but may also last for a long period. As a result, intermediate concepts may appear during the transformation as one concept (starting concept) changes to another (ending concept). An intermediate concept can be a mixture of the starting concept and the ending concept, like the incremental drift, or one of the starting or ending concept, such as the gradual drift.

3 Concept Drift Detection

This section focuses on summarizing concept drift detection algorithms. Section 3.1 introduces a typical drift detection framework. Then, Section 3.2 systematically reviews and categorizes drift detection algorithms according to their implementation details for each component in the framework. At last, Section 3.3 lists the state-of-the-art drift detection algorithms with comparisons of their implementation details.

3.1 A general framework for drift detection

Drift detection refers to the techniques and mechanisms that characterize and quantify concept drift via identifying change points or change time intervals [17]. A general framework for drift detection contains four stages, as shown in Fig. 5.

Stage 1 (Data Retrieval) aims to retrieve data chunks from data streams. Since a single data instance cannot carry enough information to infer the overall distribution [2], knowing how to organize data chunks to form a meaningful pattern or knowledge is important in data stream analysis tasks [7].

Stage 2 (Data Modeling) aims to abstract the retrieved data and extract the key features containing sensitive information, that is, the features of the data that most impact a system if they drift. This stage is optional, because it mainly concerns dimensionality reduction, or sample size reduction, to meet storage and online speed requirements [4].

Stage 3 (Test Statistics Calculation) is the measurement of dissimilarity, or distance estimation. It quantifies the severity of the drift and forms test statistics for the hypothesis test. It is considered to be the most challenging aspect of concept drift detection. The problem of how to define an accurate and robust dissimilarity measurement is still an open question. A dissimilarity measurement can also be

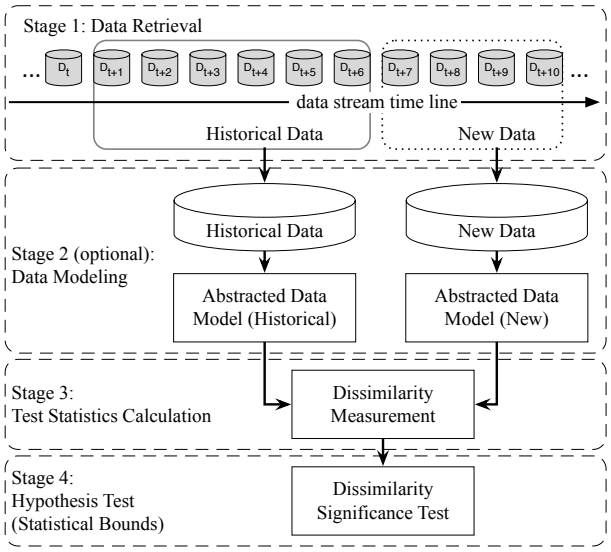


Fig. 5. An overall framework for concept drift detection

used in clustering evaluation [11], and to determine the dissimilarity between sample sets [18].

Stage 4 (Hypothesis Test) uses a specific hypothesis test to evaluate the statistical significance of the change observed in Stage 3, or the p-value. They are used to determine drift detection accuracy by proving the statistical bounds of the test statistics proposed in Stage 3. Without Stage 4, the test statistics acquired in Stage 3 are meaningless for drift detection, because they cannot determine the drift confidence interval, that is, how likely it is that the change is caused by concept drift and not noise or random sample selection bias [3]. The most commonly used hypothesis tests are: estimating the distribution of the test statistics [19], [20], bootstrapping [21], [22], the permutation test [3], and Hoeffding's inequality-based bound identification [23].

It is also worth to mention that, without Stage 1, the concept drift detection problem can be considered as a two-sample test problem which examines whether the population of two given sample sets are from the same distribution [18]. In other words, any multivariate two-sample test is an option that can be adopted in Stages 2-4 to detect concept drift [18]. However, in some cases, the distribution drift may not be included in the target features, therefore the selection of the target feature will affect the overall performance of a learning system and is a critical problem in concept drift detection [24].

3.2 Concept drift detection algorithms

This section surveys drift detection methods and algorithms, which are classified into three categories in terms of the test statistics they apply.

3.2.1 Error rate-based drift detection

PLearner error rate-based drift detection algorithms form the largest category of algorithms. These algorithms focus on tracking changes in the online error rate of base classifiers. If an increase or decrease of the error rate is proven to be statistically significant, an upgrade process (drift alarm) will be triggered.

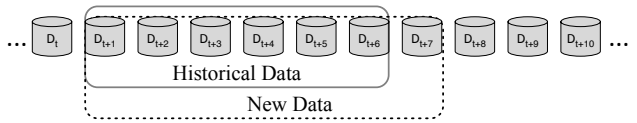


Fig. 6. Landmark time window for drift detection. The starting point of the window is fixed, while the end point of the window will be extended after a new data instance has been received.

One of the most-referenced concept drift detection algorithms is the Drift Detection Method (DDM) [20]. It was the first algorithm to define the warning level and drift level for concept drift detection. In this algorithm, Stage 1 is implemented by a landmark time window, as shown in Fig. 6. When a new data instance become available for evaluation, DDM detects whether the overall online error rate within the time window has increased significantly. If the confidence level of the observed error rate change reaches the warning level, DDM starts to build a new learner while using the old learner for predictions. If the change reached the drift level, the old learner will be replaced by the new learner for further prediction tasks. To acquire the online error rate, DDM needs a classifier to make the predictions. This process converts training data to a learning model, which is considered as the Stage 2 (Data Modeling). The test statistics in Stage 3 constitute the online error rate. The hypothesis test, Stage 4, is conducted by estimating the distribution of the online error rate and calculating the warning level and drift threshold.

Similar implementations have been adopted and applied in the Learning with Local Drift Detection (LLDD) [25], Early Drift Detection Method (EDDM) [26], Heoffding's inequality based Drift Detection Method (HDDM) [23], Fuzzy Windowing Drift Detection Method (FW-DDM) [5], Dynamic Extreme Learning Machine (DELM) [27]. LLDD modifies Stages 3 and 4, dividing the overall drift detection problem into a set of decision tree node-based drift detection problems; EDDM improves Stage 3 of DDM using the distance between two correct classifications to improve the sensitivity of drift detection; HDDM modifies Stage 4 using Hoeffding's inequality to identify the critical region of a drift; FW-DDM improves Stage 1 of DDM using a fuzzy time window instead of a conventional time window to address the gradual drift problem; DEML does not change the DDM detection algorithm but uses a novel base learner, which is a single hidden layer feedback neural network called Extreme Learning Machine (ELM) [28] to improve the adaptation process after a drift has been confirmed. EWMA for Concept Drift Detection (ECDD) [29] takes advantage of the error rate to detect concept drift. ECDD employs the EWMA chart to track changes in the error rate. The implementation of Stages 1-3 of ECDD is the same as for DDM, while Stage 4 is different. ECDD modifies the conventional EWMA chart using a dynamic mean $\hat{p}_{0,t}$ instead of the conventional static mean p_0, where $\hat{p}_{0,t}$ is the estimated online error rate within time $[0, t]$, and p_0 implies the theoretical error rate when the learner was initially built. Accordingly, the dynamic variance can be calculated by $\sigma^2_{Z_t} = \hat{p}_{0,t}(1 - \hat{p}_{0,t})\sqrt{\frac{\lambda}{2-\lambda}(1 - (1 - \lambda)^{2t})}$ where λ controls how much weight is given to more recent data as opposed to older data, and $\lambda = 0.2$ is recommended by the authors.

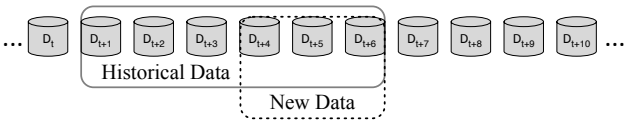


Fig. 7. Two time windows for concept drift detection. The New Data window has to be defined by the user.

Also, when the test statistic of the conventional EWMA chart is $Z_t > \hat{p}_{0,t} + 0.5L\sigma_{Z_t}$, ECDD will report a concept drift warning; when $Z_t > \hat{p}_{0,t} + L\sigma_{Z_t}$, ECDD will report a concept drift. The control limits L is given by the authors through experimental evaluation.

In contrast to DDM and other similar algorithms, Statistical Test of Equal Proportions Detection (STEPD) [30] detects error rate change by comparing the most recent time window with the overall time window, and for each timestamp, there are two time windows in the system, as shown in Fig. 7. The size of the new window must be defined by the user. According to [30], the test statistic θ_{STEPD} conforms to standard normal distribution, denoted as $\theta_{\text{STEPD}} \sim \mathcal{N}(0,1)$. The significance level of the warning level and the drift level were suggested as $\alpha_w = 0.05$ and $\alpha_d = 0.003$ respectively. As a result, the warning threshold and drift threshold can be easily calculated.

Another popular two-time window-based drift detection algorithm is **AD**aptive **WIN**dowing (ADWIN) [31]. Unlike STEPD, ADWIN does not require users to define the size of the compared windows in advance; it only needs to specify the total size n of a "sufficiently large" window W. It then examines all possible cuts of W and computes optimal sub-window sizes n_{hist} and n_{new} according to the rate of change between the two sub-windows w_{hist} and w_{new}. The test statistic is the difference of the two sample means $\theta_{\text{ADWIN}} = |\hat{\mu}_{\text{hist}} - \hat{\mu}_{\text{new}}|$. An optimal cut is found when the difference exceeds a threshold with a predefined confidence interval δ. The author proved that both the false positive rate and false negative rate are bounded by δ. It is worth noting that many concept drift adaptation methods/algorithms in the literature are derived from or combined with ADWIN, such as [32]–[35]. Since their drift detection methods are implemented with almost the same strategy, we will not discuss them in detail.

3.2.2 Data Distribution-based Drift Detection

The second largest category of drift detection algorithms is data distribution-based drift detection. Algorithms of this category use a distance function/metric to quantify the dissimilarity between the distribution of historical data and the new data. If the dissimilarity is proven to be statistically significantly different, the system will trigger a learning model upgradation process. These algorithms address concept drift from the root sources, which is the distribution drift. Not only can they accurately identify the time of drift, they can also provide location information about the drift. However, these algorithms are usually reported as incurring higher computational cost than the algorithms mentioned in Section 3.2.1 [2]. In addition, these algorithms usually require users to predefine the historical time window and new data window. The commonly used strategy is two sliding windows with the historical time window fixed

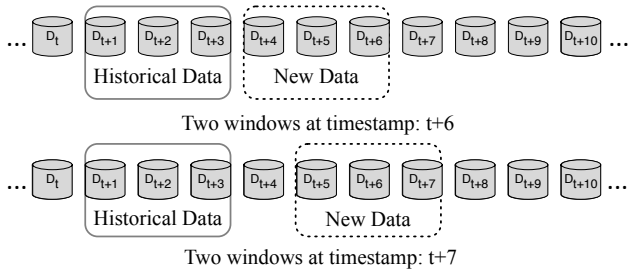


Fig. 8. Two sliding time windows, of fixed size. The Historical Data window will be fixed while the New Data window will keep moving.

while sliding the new data window [3], [22], [36], as shown in Fig. 8.

According to the literature, the first formal treatment of change detection in data streams was proposed by [37]. In their study, the authors point out that the most natural notion of distance between distributions is total variation, as defined by: $TV(P_1, P_2) = 2sup_{E\in\varepsilon}|P_1(E) - P_2(E)|$ or equivalently, when the distribution has the density functions f_1 and f_2, $dist_{L^1} = \int |f_1(x) - f_2(x)|\mathrm{d}x$. This provides practical guidance on the design of a distance function for distribution discrepancy analysis. Accordingly, [37] proposed a family of distances, called Relativized Discrepancy (RD). The authors also present the significance level of the distance according to the number of data instances. The bounds on the probabilities of missed detections and false alarms are theoretically proven, using Chernoff bounds and the Vapnik-Chervonenkis dimension. The authors of [37] do not propose novel high-dimensional friendly data models for Stage 2 (data modeling); instead, they stress that a suitable model choice is an open question.

Another typical density-based drift detection algorithm is the Information-Theoretic Approach (ITA) [22]. The intuitive idea underlying this algorithm is to use kdqTree to partition the historical and new data (multi-dimensional) into a set of bins, denoted as $\mathcal{A}$,and then use Kullback-Leibler divergence to quantify the difference of the density θ_{ITA} in each bin. The hypothesis test applied by ITA is bootstrapping by merging W_{hist}, W_{new} as W_{all} and resampling as W'_{hist}, W'_{new} to recompute the θ^*_{ITA}. Once the estimated probability $P(\theta^*_{\text{ITA}} \geq \theta_{\text{ITA}}) < 1 - \alpha$, concept drift is confirmed, where α is the significant level controlling the sensitivity of drift detection.

Similar distribution-based drift detection methods/algorithms are: Statistical Change Detection for multi-dimensional data (SCD) [38], Competence Model-based drift detection (CM) [2], a prototype-based classification model for evolving data streams called SyncStream [36], PCA-based change detection framework (PCA-CD) [39], Equal Density Estimation (EDE) [40], Least Squares Density Difference-based Change Detection Test (LSDD-CDT) [21], Incremental version of LSDD-CDT (LSDD-INC) [41] and Local Drift Degree-based Density Synchronized Drift Adaptation (LDD-DSDA) [4].

3.2.3 Multiple Hypothesis Test Drift Detection

Multiple hypothesis test drift detection algorithms apply similar techniques to those mentioned in the previous two categories. The novelty of these algorithms is that they use multiple hypothesis tests to detect concept drift in different

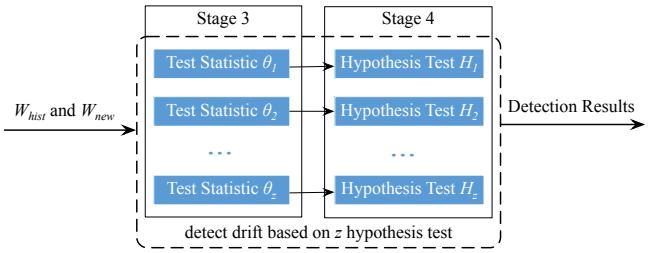


Fig. 9. Parallel multiple hypothesis test drift detection.

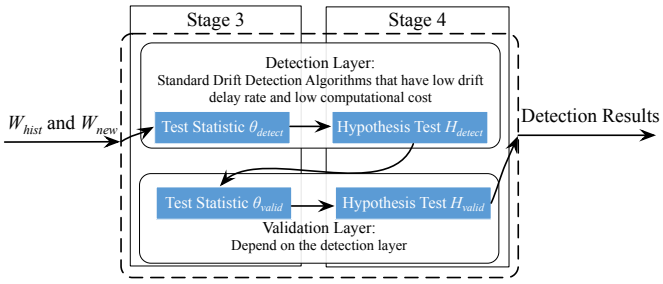


Fig. 10. Hierarchical multiple hypothesis test drift detection.

ways. These algorithms can be divided into two groups: 1) parallel multiple hypothesis tests; and 2) hierarchical multiple hypothesis tests.

The idea of parallel multiple hypothesis drift detection algorithm is demonstrated in Fig. 9. According to the literature, Just-In-Time adaptive classifiers (JIT) [19] is the first algorithm that set multiple drift detection hypothesis in this way. The core idea of JIT is to extend the CUSUM chart, known as the Computational Intelligence-based **CUSUM** test (CI-CUSUM), to detect the mean change in the features interested by learning systems. The authors of [19], gave the following four configurations for the drift detection target. Config1: the features extracted by Principal Component Analysis (PCA), which removes eigenvalues whose sum is below a threshold, e.g. 0.001. Config2: PCA extracted features plus one generic component of the original features x_i; Config3: detects the drift in each x_i individually. Config4: detects drift in all possible combinations of the feature space x_i. The authors stated that Config2 is the preferred setting for most situations, according to their experimentation, and also mentioned that Config1 may have a high missing rate, Config3 suffers from a high false alarm rate, and Config4 has exponential computational complexity. The same drift detection strategy has also been applied in [42]–[45] for concept drift adaptation.

Similar implementations have been applied in Linear Four Rate drift detection (LFR) [46], which maintains and tracks the changes in True Positive rate (TP), True Negative rate (TN), False Positive rate (FP) and False Negative rate (FN) in an online manner. The drift detection process also includes warning and drift levels.

Another parallel multiple hypothesis drift detection algorithm is three-layer drift detection, based on Information Value and Jaccard similarity (IV-Jac) [47]. IV-Jac aims to individually address the label drift $P_t(y)$ Layer I, feature space drift $P_t(X)$ Layer II, and decision boundary drift $P_t(y|X)$ Layer III. It extracts the Weight of Evidence (WoE) and Information Value (IV) from the available data and then detects whether a significant change exists between the WoE and IV extracted from W_{hist} and W_{new} by measuring the contribution to the label for a feature value. The hypothesis test thresholds are predefined parameters $\theta_{P_t(y)} = \theta_{P_t(X)} = \theta_{P_t(X|y)} = 0.5$ by default, which are chosen empirically.

Ensemble of Detectors (e-Detector) [48] proposed to detect concept drift via ensemble of heterogeneous drift detector. The authors consider two drift detectors are homogeneous as if they are equivalent in finding concept drifts, otherwise they are heterogeneous. e-Detector groups homogeneous drift detectors via a diversity measurement, named diversity vector. For each group, it select the one with the smallest coefficient of failure as the base detector to form the ensemble. e-Detector reports concept drift following the early-find-early-report rule, which means no matter which base detector detect a drift, the e-Detector reports a drift. Similar strategy has been applied in drift detection ensemble (DDE) [49].

Hierarchical drift detection is an emerging drift detection category that has a multiple verification schema. The algorithms in this category usually detect drift using an existing method, called the detection layer, and then apply an extra hypothesis test, called the validation layer, to obtain a second validation of the detected drift in a hierarchical way. The overall workflow is shown in Fig. 10.

According to the claim made by [50], Hierarchical Change-Detection Tests (HCDTs) is the first attempt to address concept drift using a hierarchical architecture. The detection layer can be any existing drift detection method that has a low drift delay rate and low computational burden. The validation layer will be activated and deactivated based on the results returned by the detection layer. The authors recommend two strategies for designing the validation layer: 1) estimating the distribution of the test statistics by maximizing the likelihood; 2) adapting an existing hypothesis test, such as the Kolmogorov-Smirnov test or the Cramer-Von Mises test.

Hierarchical Linear Four Rate (HLFR) [51] is another recently proposed hierarchical drift detection algorithm. It applies the drift detection algorithm LFR as the detection layer. Once a drift is confirmed by the detection layer, the validation layer will be triggered. The validation layer of HLFR is simply a zero-one loss, denoted as E, over the ordered train-test split. If the estimated zero-one loss exceeds a predefined threshold, $\eta = 0.01$, the validation layer will confirm the drift and report to the learning system to trigger a model upgradation process.

Two-Stage Multivariate Shift-Detection based on **EWMA** (TSMSD-EWMA) [52] has a very similar implementation, however, the authors do not claim that their method is a hierarchy-based algorithm.

Hierarchical Hypothesis Testing with Classification Uncertainty (HHT-CU) and Hierarchical Hypothesis Testing with Attribute-wise "Goodness-of-fit" (HHT-AG) are two drift detection algorithms based on request and reverify strategy [53]. For HHT-CU, the detection layer is a hypotheses test based on Heoffding's inequality that monitoring the change of the classification uncertainty measurement. The validation layer is a permutation test that evaluates the change of the zero-one loss of the learner. For HHT-AG, the

detection layer is conducted based on Kolmogorov-Smirnov (KS) test for each feature distribution. Then HHT-AG validate the potential drift points by requiring true labels of data that come from w_{new}, and performing d independent two-dimensional (2D) KS test with each feature-label bivariate distribution. Compare to other drift detection algorithms, HHT-AG can handle concept drift with less true labels, which makes it more powerful when dealing with high verification latency.

3.3 Summary of concept drift detection methods/algorithms

TABLE 1 lists the most popular concept drift detection methods/algorithms against the general framework summarized in Section 3.1 (Fig. 5). A comparative study on eight popular drift detection methods can be found in [54].

4 Concept Drift understanding

Drift understanding refers to retrieving concept drift information about "When" (the time at which the concept drift occurs and how long the drift lasts), "How" (the severity /degree of concept drift), and "Where" (the drift regions of concept drift). This status information is the output of the drift detection algorithms, and is used as input for drift adaptation.

4.1 The time of concept drift occurs (When)

The most basic function of drift detection is to identify the timestamp when a drift occurs. Recalling the definition of concept drift $\exists t\colon P_t(X,y) \neq P_{t+1}(X,y)$, the variable t represents the time at which a concept drift occurs. In drift detection methods/algorithms, an alarm signal is used to indicate whether the concept drift has or has not occurred or not at the current timestamp. It is also a signal for a learning system to adapt to a new concept. Accurately identifying the time a drift occurs is critical to the adaptation process of a learning system; a delay or a false alarm will lead to failure of the learning system to track new concepts.

A drift alarm usually has a statistical guarantee with a predefined false alarm rate. Error rate-based drift detection algorithms monitor the performance of the learning system, based on statistical process control. For example, DDM [20] sends a drift signal when the learning accuracy of the learner drops below a predefined threshold, which is chosen by the three-sigma rule [55]. ECCD [29] reports a drift when the online error rate exceeds the control limit of EWMA. Most data distribution-based drift detection algorithms report a drift alarm when two data samples have a statistically significant difference. PCA-based drift detection [36] outputs a drift signal when the p-value of the generalized Wilcoxon test statistic W^l_{BF} is significantly large. The method in [3] confirms that a drift has occurred by verifying whether the empirical competence-based distance is significantly large through permuataion test.

Taking into account the various drift types, concept drift understanding needs to explore the start time point, the change period, and the end time point of concept drift. And these time information could be useful input for the adaptation process of the learning system. However the drift timestamp alert in existing drift detection algorithms is delayed compared to the actual drifting timestamp, since most drift detectors require a minimum number of new data to evaluate the status of the drift, as shown in Fig. 11. The emergence time of the new concept is therefore still vague. Some concept drift detection algorithms such as DDM [20], EDDM [26], STEPD [30], and HDDM [23], trigger a warning level to indicate a drift may have occurred. The threshold used to trigger warning level is a relaxed condition of the threshold used for the drift level; for example, the warning level is set p-value to 95% or 2σ, and the drift level is set p-value to 99% or 3σ. The data accumulated between the warning level and the drift level are used as the training set for updating a learning model.

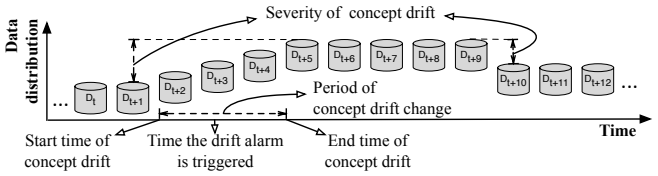


Fig. 11. An example of the occurrence time and the severity of concept drift. One incremental drift starts to change at $t+1$ and ends at $t+5$. The other sudden drift occurs between $t+9$ and $t+10$. The severity of these two concept drifts (D_{t+1}-D_{t+5} and D_{t+9}-D_{t+10}) is marked with brackets.

4.2 The severity of concept drift (How)

The severity of concept drift refers to using a quantified value to measure the similarity between the new concept and the previous concept, as shown in Fig. 11. Formally, the severity of concept drift can be represented as $\Delta = \delta(P_t(X,y), P_{t+1}(X,y))$, where δ is a function to measure the discrepancy of two data distributions, and t is the timestamp when the concept drift occurred. Δ usually is a non-negative value indicating the severity of concept drift. The greater the value of Δ, the larger the severity of the concept drift is.

In general, error rate-based drift detection cannot directly measure the severity of concept drift, because it mainly focuses on monitoring the performance of the learning system, not the changes in the concept itself. However, the degree of decrease in learning accuracy can be used as an indirect measurement to indicate the severity of concept drift. If learning accuracy has dropped significantly when drift is observed, this indicates that the new concept is different from the previous one. For example, the severity of concept drift could be reflected by the difference between p_i and $p_{\min}$ in [20], [27], denoted as $\Delta \sim p_i - p_{\min}$; the difference between overall accuracy $\hat{p}_{\text{hist}}$ and recent accuracy $\hat{p}_{\text{new}}$ in [30], expressed as $\Delta \sim \hat{p}_{\text{new}} - \hat{p}_{\text{hist}}$; and the difference between test statistics in the former window $E[\hat{X}_{\text{cut}}]$ and test statistics in the later window $E[\hat{Y}_{i-\text{cut}}]$ [23], denoted as $\Delta \sim E[\hat{Y}_{i-\text{cut}}] - E[\hat{X}_{\text{cut}}]$. However, the meaning of these differences is not discussed in existing publications. The ability of error rate-based drift detection to output the severity of concept drift is still vague.

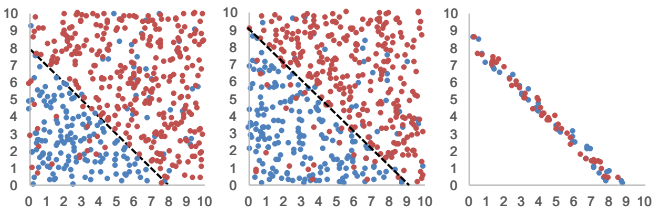
Data distribution-based drift detection methods can directly quantify the severity of concept drift since the measurement used to compare two data samples already reflects the difference. For example, [37] employed *a relaxation of the total variation distance* $d_{\mathcal{A}}(S_1, S_2)$ to measure the difference

TABLE 1
Summary of drift detection algorithms

Category	Algorithms	Stage 1	Stage 2	Stage 3	Stage 4
Error rate-based	DDM [20]	Landmark	Learner	Online error rate	Distribution estimation
	EDDM [26]	Landmark	Learner	Online error rate	Distribution estimation
	FW-DDM [5]	Landmark	Learner	Online error rate	Distribution estimation
	DEML [27]	Landmark	Learner	Online error rate	Distribution estimation
	STEPD [30]	Predefined w_{hist}, w_{new}	Learner	Error rate difference	Distribution estimation
	ADWIN [31]	Auto cut w_{hist}, w_{new}	Learner	Error rate difference	Hoeffding's Bound
	ECDD [29]	Landmark	Learner	Online error rate	EWMA Chart
	HDDM [23]	Landmark	Learner	Online error rate	Hoeffding's Bound
	LLDD [25]	Landmark, or sliding w_{hist}, w_{new}	Decision trees	Tree node error rate	Hoeffding's Bound
Data distribution-based	kdqTree [22]	Fixed w_{hist}, Sliding w_{new}	kdqTree	KL divergence	Bootstrapping
	CM [2], [3]	Fixed w_{hist}, Sliding w_{new}	Competence model	Competence distance	Permutation test
	RD [37]	Fixed w_{hist}, Sliding w_{new}	KS structure	Relativized Discrepancy	VC-Dimension
	SCD [38]	Fixed w_{hist}, Sliding w_{new}	kernel density estimator	log-likelihood	Distribution estimation
	EDE [40]	Fixed w_{hist}, Sliding w_{new}	Nearest neighbor	Density scale	Permutation test
	SyncStream [36]	Fixed w_{hist}, Sliding w_{new}	PCA	P-Tree	Wilcoxon test
	PCA-CD [39]	Fixed w_{hist}, Sliding w_{new}	PCA	Change-Score	Page-Hinkley test
	LSDD-CDT [21]	Fixed w_{hist}, Sliding w_{new}	Learner	Relative difference	Distribution estimation
	LSDD-INC [41]	Fixed w_{hist}, Sliding w_{new}	Learner	Relative difference	Distribution estimation
	LDD-DSDA [4]	Fixed w_{hist}, Sliding w_{new}	k-nearest neighbor	Local drift degree	Distribution estimation
Multiple Hypothesis Tests	JIT [19]	Landmark	Selected features	4 configurations	Distribution estimation
	LFR [46]	Landmark	Learner	TP, TN, FP, FN	Distribution estimation
	Three-layer [47]	Sliding both w_{hist}, w_{new}	Learner	$P(y)$, $P(X)$, $P(X\|y)$	Distribution estimation
	e-Detector [48]	depends on base detector	depends	depends	depends
	DDE [49]	depends on base detector	depends	depends	depends
	TSMSD-EWMA [52]	Landmark	Learner	Online error rate	EWMA Chart
	HCDTs [50]	Landmark	Depending on layers	Depending on layers	Depending on layer
	HLFR [51]	Landmark	Learner	TP, TN, FP, FN	Distribution estimation
	HHT-CU [53]	Landmark	Learner	Classification uncertainty	Layer-I Hoeffding's Bound, Layer-II Permutation Test
	HHT-AG [53]	Fixed w_{hist}, Sliding w_{new}	N/A	KS statistic on each attribute	Layer-I KS test, Layer -II 2D KS test

between two data distributions. [3] proposed *a competence-based empirical distance* $d^{CB}(S_1, S2)$ to show the difference between two data samples. Other drift detection methods have used information-theoretic distance; for example, *Kullback-Leibler divergence* $D(W_1||W_2)$, also called relative entropy, was used in the study reported in [22]. The range of these distances is $[0, 1]$. The greater the distance, the larger the severity of the concept drift is. The distance "1" means that a new concept is different from the pervious one, while the distance "0" means that two data concepts are identical. The *test statistic* δ used in [38] gives an extremely small negative value if the new concept is quite different from the previous concept. The degree of concept drift in [36] is measured by the resulting p-value of the test statistic W^l_{BF} and the defined $Angle(D_t, D_{t+1})$ of two datasets D_t and D_{t+1}.

The severity of concept drift can be used as a guideline for choosing drift adaptation strategies. For example, if the severity of drift in a classification task is low, the decision boundary may not move much in the new concept. Thus, adjusting the current learner by incremental learning will be adequate. In contrast, if the severity of the concept drift is high, the decision boundary could change significantly, therefore discarding the old learner and retraining a new one could be better than incrementally updating the old learner. We would like to mention that, even though some researches have promoted the ability to describe and quantify the severity of the detected drift, this information is not yet widely utilized in drift adaptation.



(a) Data at time t (b) Data at time $t+1$ (c) Data in drift regions

Fig. 12. An example of the drift regions of concept drift.

4.3 The drift regions of concept drift (Where)

The drift regions of concept drift are the regions of conflict between a new concept and the previous concept. Drift regions are located by finding regions in data feature space X where $P_t(X, y)$ and $P_{t+1}(X, y)$ have significant difference. To illustrate this, we give an example of a classification task in Fig. 12. The data used in this task are uniformly sampled in the range of $[0, 10]^2$. The left sub-figure is the data sample accumulated at time t, where the decision boundary is $x_1 + x_2 = 8$. The middle sub-figure is the data accumulated at time $t + 1$, where the decision boundary is $x_1 + x_2 = 9$. Intuitively, data located in regions $8 \leq x_1 + x_2 < 9$ have different classes in time t and time $t + 1$, since the decision boundary has changed. The right sub-figure shows the data located in the drift regions.

The techniques to identify drift regions are highly dependent on the data model used in the drift detection

methods/algorithms. Paper [25] detected drift data in local regions of the instance space by using online error-rate inside the inner-nodes of a decision tree. The whole data feature space is partitioned by decision tree. Each leaf of this decision tree corresponds to a hyper-rectangle in the data feature space. All leaf nodes contain a drift detector. When the leaf nodes are alerted to a drift, the corresponding hyper-rectangles indicate the regions of drift in the data feature space. Similar to [25], [22] utilized the nodes of the *kdq-tree* with Kulldorff's spatial scan statistic to identify the regions in which data had changed the most. Once a drift has been reported, Kulldorff's statistic measures how the two datasets differ only with respect to the region associated with the leaf node of the *kdq-tree*. The leaf nodes with the greater statistical value of show the drift regions. [2] highlighted the most severe regions of concept drift through top-p-competence areas. Utilizing the *RelatedSets* of the competence model, the data feature space is partitioned by a set of overlapping hyperspheres. The *RelatedSet*-based empirical distance defines the distance between two datasets on a particular hypersphere. The drift regions are identified by the corresponding hyperspheres with large empirical distance at top p% level. [4] identified drift regions by monitoring the discrepancy in the regional density of data, which is called the local drift degree. A local region with a corresponding increase or decrease in density will be highlighted as a critical drift region.

Locating concept drift regions benefits drift adaptation. Paper [56] pointed out that even if the concept of the entire dataset drifts, there are regions of the feature space that will remain stable significantly longer than other regions. In an ensemble scenario, the old learning models of stable regions could still be useful for predicting data instances located within stable regions, or data instances located in drift regions could be used to build a more updated current model. The authors of [3] also pointed out that identifying drift regions can help in recognizing obsolete data that conflict with current concepts and distinguish noise data from novel data. In their later research [2], they utilized top-p-competence areas to edit cases in a case-based reasoning system for fast new concept switching. One step in their drift adaptation is to remove conflict instances. To preserve as many instances of a new concept as possible, they only remove obsolete conflict instances which are outside the drift regions. LDD-DSDA [4] utilized drift regions as an instance selection strategy to construct a training set that continually tracked a new concept.

4.4 Summary of drift understanding

We summarize concept drift detection algorithms according to their ability to identify when, how, and where concept drift occurs, as shown in TABLE 2. All drift detection algorithms can identify the occurrence time of concept drift (when), and most data distribution-based drift detection algorithms can also measure the severity of concept drift (how) through the test statistics, but only a few algorithms have the ability to locate drift regions (where).

TABLE 2
Summary of drift understanding for drift detection algorithms

Category	Algorithms	When	How	Where
Error rate-based	DDM [20]	√		
	EDDM [26]	√		
	FW-DDM [5]	√		
	DEML [27]	√		
	STEPD [30]	√		
	ADWIN [31]	√		
	ECDD [29]	√		
	HDDM [23]	√		
	LLDD [25]	√		√
Data distribution-based	kdqTree [22]	√	√	√
	CM [2], [3]	√	√	√
	RD [37]	√	√	
	SCD [38]	√	√	
	EDE [40]	√		
	SyncStream [36]	√	√	
	PCA-CD [39]	√	√	
	LSDD-CDT [21]	√		
	LSDD-INC [41]	√		
	LDD-DSDA [4]	√	√	√
Multiple hypothesis tests	JIT [19]	√		
	LFR [46]	√		
	Three-layer drift detection [47]	√		
	e-Detector [48]	√		
	DDE [49]	√		
	EWMA [52]	√		
	HCDTs [50]	√		
	HLFR [51]	√		
	HHT-CU [53]	√		
	HHT-AG [53]	√		

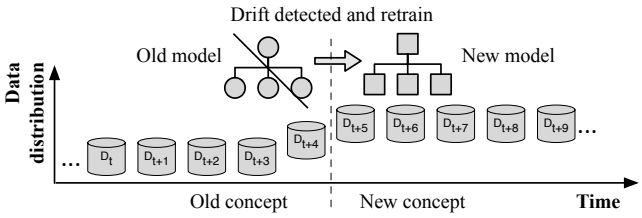


Fig. 13. A new model is trained with latest data to replace the old model when a concept drift is detected.

5 Drift Adaptation

This section focuses on strategies for updating existing learning models according to the drift, which is known as drift adaptation or reaction. There are three main groups of drift adaptation methods, namely simple retraining, ensemble retraining and model adjusting, that aim to handle different types of drift.

5.1 Training new models for global drift

Perhaps the most straightforward way of reacting to concept drift is to retrain a new model with the latest data to replace the obsolete model, as shown in Fig. 13. An explicit concept drift detector is required to decide when to retrain the model (see Section 3 on drift detection). A window strategy is often adopted in this method to preserve the most recent data for retraining and/or old data for distribution change test. *Paired Learners* [57] follows this strategy and uses two learners: the *stable learner* and the *reactive learner*. If the stable learner frequently misclassifies instances that the reactive learner correctly classifies, a new concept is detected and the stable learner will be replaced with the reactive learner. This method is simple to understand and easy to implement, and can be applied at any point in the data stream.

When adopting a window-based strategy, a trade-off must be made in order to decide an appropriate window size. A small window can better reflect the latest data distribution, but a large window provides more data for training a new model. A popular window scheme algorithm that aims to mitigate this problem is ADWIN [31]. Unlike most earlier works, it does not require the user to guess a fixed size of the windows being compared in advance; instead, it examines all possible cuts of the window and computes optimal sub-window sizes according to the rate of change between the two sub-windows. After the optimal window cut has been found, the window containing old data is dropped and a new model can be trained with the latest window data.

Instead of directly retraining the model, researchers have also attempted to integrate the drift detection process with the retraining process for specific machine learning algorithms. DELM [27] extends the traditional ELM algorithm with the ability to handle concept drift by adaptively adjusting the number of hidden layer nodes. When the classification error rate increases — which could indicate the emergence of a concept drift — more nodes are added to the network layers to improve its approximation capability. Similarly, FP-ELM [58] is an ELM-extended method that adapts to drift by introducing a forgetting parameter to the ELM model. A parallel version of ELM-based method [59] has also been developed for high-speed classification tasks under concept drift. OS-ELM [60] is another online learning ensemble of repressor models that integrates ELM using an ordered aggregation (OA) technique, which overcomes the problem of defining the optimal ensemble size.

Instance-based lazy learners for handling concept drift have also been studied intensively. The *Just-in-Time* adaptive classifier [19], [42] is such a method which follows the "detect and update model" strategy. For drift detection, it extends the traditional CUSUM test [61] to a pdf-free form. This detection method is then integrated with a kNN classifier [42]. When a concept drift is detected, old instances (more than the last T samples) are removed from the case base. In later work, the authors of [11], [45] extended this algorithm to handle recurrent concepts by computing and comparing current concept to previously stored concepts. NEFCS [2] is another kNN-based adaptive model. A competence model-based drift detection algorithm [3] was used to locate drift instances in the case base and distinguish them from noise instances and a redundancy removal algorithm, Stepwise Redundancy Removal (SRR), was developed to remove redundant instances in a uniform way, guaranteeing that the reduced case base would still preserve enough information for future drift detection.

5.2 Model ensemble for recurring drift

In the case of recurring concept drift, preserving and reusing old models can save significant effort to retrain a new model for recurring concepts. This is the core idea of using ensemble methods to handle concept drift. Ensemble methods have received much attention in stream data mining research community in recent years. Ensemble methods comprise a set of base classifiers that may have different types or different parameters. The output of each base classifier is combined using certain voting rules to predict the newly arrived data. Many adaptive ensemble methods have been developed that aim to handle concept drift by extending classical ensemble methods or by creating specific adaptive voting rules. An example is shown in Fig. 14, where new base classifier is added to the ensemble when concept drift occurs.

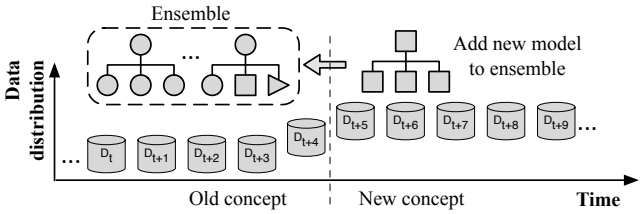


Fig. 14. A new base classifier is added to the ensemble when a concept drift occurs.

Bagging, Boosting and Random Forests are classical ensemble methods used to improve the performance of single classifiers. They have all been extended for handling streaming data with concept drift. An online version of the bagging method was first proposed in [62] which uses each instance only once to simulate the batch mode bagging. In a later study [63], this method was combined with the ADWIN drift detection algorithm [31] to handle concept drift. When a concept drift is reported, the newly proposed method, called Leveraging Bagging, trains a new classifier on the latest data to replace the existing classifier with the worst performance. Similarly, an adaptive boosting method was developed in [64] which handles concept drift by monitoring prediction accuracy using a hypothesis test, assuming that classification errors on non-drifting data should follow Gaussian distribution. In a recent work [35], the Adaptive Random Forest (ARF) algorithm was proposed, which extends the random forest tree algorithm with a concept drift detection method, such as ADWIN [31], to decide when to replace an obsolete tree with a new one. A similar work can be found in [65], which uses Hoeffding bound to distinguish concept drift from noise within decision trees.

Besides extending classical methods, many new ensemble methods have been developed to handle concept drift using novel voting techniques. **D**ynamic **W**eighted **M**ajority (DWM) [66] is such an ensemble method that is capable of adapting to drifts with a simple set of weighted voting rules. It manages base classifiers according to the performance of both the individual classifiers and the global ensemble. If the ensemble misclassifies an instance, DWM will train a new base classifier and add it to ensemble. If a base classifier misclassifies an instance, DWM reduces its weight by a factor. When the weight of a base classifier drops below a user defined threshold, DWM removes it from the ensemble. The drawback of this method is that the adding classifier process may be triggered too frequently, introducing performance issues on some occasions, such as when gradual drift occurs. A well-known ensemble method, Learn++NSE [67], mitigates this issue by weighting base classifiers according to their prediction error rate on the latest batch of data. If the error rate of the youngest classifier exceeds 50%, a new classifier will be trained based on the latest data. This method has several other benefits: it

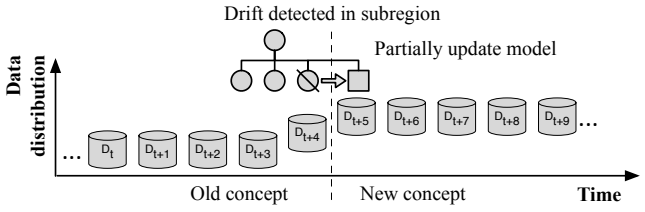


Fig. 15. A decision tree node is replaced with a new one as its performance deteriorates when a concept drift occurs in a subregion.

can easily adopt almost any base classifier algorithm; it does not store history data, only the latest batch of data, which it only uses once to train a new classifier; and it can handle sudden drift, gradual drift, and recurrent drift because underperforming classifiers can be reactivated or deactivated as needed by adjusting their weights. Other voting strategies than standard weighted voting have also been applied to handle concept drift. Examples include hierarchical ensemble structure [68], [69], short term and long term memory [13], [70] and dynamic ensemble sizes [71], [72].

A number of research efforts have been made that focus on developing ensemble methods for handling concept drift of certain types. **A**ccuracy **U**pdate **E**nsemble (AUE2) [73] was proposed with an emphasis on handling both sudden drift and gradual drift equally well. It is a batch mode weighted voting ensemble method based on incremental base classifiers. By doing re-weighting, the ensemble is able react quickly to sudden drift. All classifiers are also incrementally trained with the latest data, which ensures that the ensemble evolves with gradual drift. The Optimal Weights Adjustment (OWA) method [74] achieves the same goal by building ensembles using both weighted instances and weighted classifiers for different concept drift types. The authors of [75] considered a special case of concept drift — class evolution — the phenomenon of class emergence and disappearance. Recurring concepts are handled in [76], [77], which monitor concept information to decide when to reactivate previously stored obsolete models. [78] is another method that handles recurring concepts by refining the concept pool to avoid redundancy.

5.3 Adjusting existing models for regional drift

An alternative to retraining an entire model is to develop a model that adaptively learns from the changing data. Such models have the ability to partially update themselves when the underlying data distribution changes, as shown in Fig. 15. This approach is arguably more efficient than retraining when the drift only occurs in local regions. Many methods in this category are based on the decision tree algorithm because trees have the ability to examine and adapt to each sub-region separately.

In a foundational work [79], an online decision tree algorithm, called **V**ery **F**ast **D**ecision **T**ree classifier (VFDT) was proposed, which is especially tailored for high speed data streams. It uses Hoeffding bound to limit the number of instances required for node splitting. This method has become very popular because of its several distinct advantages: 1) it only needs to process each instance once and does not store instances in memory or disk; 2) the tree itself only consumes a small amount of space and does not grow with the number of instances it processes unless there is new information in the data; 3) the cost of tree maintenance is very low. An extended version, called CVFDT [80], was later proposed to handle concept drift. In CVFDT, a sliding window is maintained to hold the latest data. An alternative sub-tree is trained based on the window and its performance is monitored. If the alternative sub-tree outperforms its original counterpart, it will be used for future prediction and the original obsolete sub-tree will be pruned. VFDTc [81] is another attempt to make improvements to VFDT with several enhancements: the ability to handle numerical attributes; the application of naive Bayes classifiers in tree leaves and the ability to detect and adapt to concept drift. Two node-level drift detection methods were proposed based on monitoring differences between a node and its sub-nodes. The first method uses classification error rate and the second directly checks distribution difference. When a drift is detected on a node, the node becomes a leaf and its descending sub-tree is removed. Later work [82], [83] further extended VFDTc using an adaptive leaf strategy that chooses the best classifier from three options: majority voting, Naive Bayes and Weighted Naive Bayes.

Despite the success of VFDT, recent studies [84], [85] have shown that its foundation, the Hoeffding bound, may not be appropriate for the node splitting calculation because the variables it computes, the information gain, are not independent. A new online decision tree model [86] was developed based on an alternative impurity measure. The paper shows that this measure also reflects concept drift and can be used as a replacement measure in CVFDT. In the same spirit, another decision tree algorithm (IADEM-3) [87] aims to rectify the use of Hoeffding bound by computing the sum of independent random variables, called *relative frequencies*. The error rate of sub-trees are monitored to detect drift and are used for tree pruning.

6 Evaluation, Datasets and Benchmarks

Section 6.1 discusses the evaluation systems used for learning algorithms handling concept drift. Section 6.2 introduces synthetic datasets, which used to simulate specific and controllable types of concept drift. Section 6.3 describes real-world datasets, which used to test the overall performance in a real-life scenario.

6.1 Evaluation Systems

The evaluation systems is an important part for learning algorithms. Some evaluation methodologies used in learning under concept drift have been mentioned in [8]. We enrich this previous research by reviewing the evaluation systems from three aspects: 1) validation methodology, 2) evaluation metrics, and 3) statistical significance, and each evaluation is followed by its computation equation and usage introduction.

Validation methodology refers to the procedure for a learning algorithm to determine which data instances are used as the training set and which are used as the testing set. There are three procedures peculiar to the evaluation for learning algorithms capable of handling concept drift:

1) *holdout*, 2) *prequential*, and 3) *controlled permutation*. In the scenario of a dataset involving concept drift, *holdout* should follow the rule: when testing a learning algorithm at time t, the holdout set represents exactly the same concept at that time t. Unfortunately, it is only applied on synthetic datasets with predefined concept drift times. *Prequential* is a popular evaluation scheme used in streaming data. Each data instance is first used to test the learning algorithm, and then to train the learning algorithm. This scheme has the advantage that there is no need to know the drift time of concepts, and it makes maximum use of the available data. The prequential error is computed based on an accumulated sum of a loss function between the prediction and observed label: $S = \sum_{t=1}^{n} f(\hat{y}_t, y_t)$. There are three prequential error rate estimates: a landmark window (interleaved-test-then-train), a sliding window, and a forgetting mechanism [88]. *Controlled permutation* [89] runs multiple test datasets in which the data order has been permutated in a controlled way to preserve the local distribution, which means that data instances that were originally close to one another in time need to remain close after a permutation. Controlled permutation reduces the risk that their prequential evaluation may produce biased results for the fixed order of data in a sequence.

Evaluation metrics for datasets involving concept drift could be selected from traditional accuracy measures, such as precision/recall in retrieval tasks, mean absolute scaled error in regression, or root mean square error in recommender systems. In addition to that, the following measures should be examined: 1) *RAM-hours* [90] for the computation cost of the mining process; 2) *Kappa statistic* $\kappa = \frac{p-p_{\text{ran}}}{1-p_{\text{ran}}}$ [91] for classification taking into account class imbalance, where p is the accuracy of the classifier under consideration (reference classifier) and p_{ran} is the accuracy of the random classifier; 3) *Kappa-Temporal statistic* $\kappa_{per} = \frac{p-p_{\text{per}}}{1-p_{\text{per}}}$ [92] for the classification of streaming data with temporal dependence, where p_{per} is the accuracy of the persistent classifier (a classifier that predicts the same label as previously observed); 4) *Combined Kappa statistic* $\kappa^{+} = \sqrt{\max(0,\kappa)\max(0,\kappa_{\text{per}})}$ [92], which combines the κ and κ_{per} by taking the geometric average; 5) *Prequential AUC* [93]; and 6) the Averaged Normalized Area Under the Curve (NAUC) values for Precision-Range curve and Recall-Range curve [53], for the classification of streaming data involving concept drift. Apart from evaluating the performance of learning algorithms, the accuracy of the concept drift detection method/algorithm can be accessed according to the following criteria: 1) *true detection rate*, 2) *false detection rate*, 3) *miss detection rate*, and 4) delay of detection [22].

Statistical significance is used to compare learning algorithms on achieved error rates. The three most frequently used statistical tests for comparing two learning algorithms [94], [95] are: 1) McNemar test [96]: denote the number of data instances misclassified by the first classifier and correctly classified by the second classifier by a, and denote b in the opposite way. The McNemar statistic is computed as $M = \text{sign}(a-b) \times (a-b)^2/(a+b)$ to test whether two classifiers perform equally well. The test follows the χ^2 distribution; 2) Sign test: for N data instances, denote the number of data instances misclassified by the first classifier and correctly classified by the second classifier by B and the number of ties by T. Conduct one-sided sign test by computing $p = \sum_{k=B}^{N-T} \binom{N-T}{k} 0.5^k \times 0.5^{N-T-k}$. If p less than a significant level, then the second classifier is better than the first classifier. and 3) Wilcoxon's sign-rank test: For testing two classifiers on N datasets, let $x_{i,1}$ and $x_{i,2}$ $(i = 1, \ldots, N)$ denote the measurements. The number of ties is T and $N_r = N - T$. The test statistic $W = \sum_{i=1}^{N_r} (\text{sign}(x_{i,1} - x_{i,2}) \times R_i)$ where R_i is the rank ordered by $|x_{i,1} - x_{i,2}|$ increasingly. Two classifiers perform equally is rejected if $|W| > W_{\text{critical},N_r}$ (two-sided), where W_{critical,N_r} can be acquired from the statistical table. All three tests are non-parametric. The Nemenyi test [97] is used to compare more than two learning algorithms. It is an appropriate test for comparing all learning algorithms with multiple datasets, based on the average rank of algorithms over all datasets. The Nemenyi test consists of the following: two classifiers are performing differently if the corresponding average ranks differ by at least the critical difference CD $= q_\alpha \sqrt{k(k+1)/6N}$, where k is the number of learners, N is the number of datasets, and critical values q_α are based on the Studentized range statistic divided by $\sqrt{2}$. Other tests can be used to compare learning algorithms with a control algorithm [97].

6.2 Synthetic datasets

We list several widely used synthetic datasets for evaluating the performance of learning algorithms dealing with concept drift. Since data instances are generated by predefined rules and specific parameters, a synthetic dataset is a good option for evaluating the performance of learning algorithms in different concept drift scenarios. The dataset provider, the number of instances (#Insts.), the number of attributes (#Attrs.), the number of classes (#Cls.), types of drift (Types), sources of drift (Sources), and used by references, are listed in TABLE 3.

6.3 Real-world datasets

In this section, we collect several publicly available real-world datasets, including real-world datasets with synthetic drifts. The dataset provider, the number of instances (#Insts.), the number of attributes (#Attrs.), the number of classes (#Cls.), and used by references, are shown in TABLE 4.

Most of these datasets contain temporal concept drift spanning over different period range - e.g. daily (Sensor [108]), seasonally (Electricity [109]) or yearly (Airlines [104], NOAA weather [67]). Others include geographical (Covertype [106]) or categorical (Poker-Hand [106]) concept drift. Certain datesets, mainly text based, are targeting at specific drift types, such as sudden drift (Email_data [110]), gradural drift (Spam assassin corpus [111]), recurrent drift (Usenet [112]) or novel class (KDDCup'99 [106], ECUE drift dataset 2 [113])

These datasets provide realistic benchmark for evaluating differnent concept drift handling methods. There are, however, two limitations of real world data sets: 1) the groud truth of precise start and end time of drifts is unknown; 2) some real datasets may include mixed drift types. These limitations make it difficult to evaluate methods for

TABLE 3
List of synthetic datasets for performance evaluation of learning under concept drift.

	Dataset	#Insts.	#Attrs.	#Cls.	Types	Sources	Used by references
1	STAGGER [1]	Custom	3	2	Sudden	II	[20], [23], [27], [30], [41], [57], [65], [72], [87], [98], [99]
2	SEA [100]	Custom	3	2	Sudden	II	[2], [5], [13], [20], [27], [32], [35], [51], [57], [58], [63], [65], [67], [73], [76], [99]–[102]
3	Rotating hyperplane [80], [103]	Custom	10	2	Gradual; Incremental	II	[2], [13], [21], [27], [30], [32], [35], [36], [41], [51], [58], [59], [63], [71]–[73], [78], [80], [83], [87], [102]
4	Random RBF [104]	Custom	Custom	Custom	Sudden; Gradual; Incremental	III	[13], [20], [21], [26], [27], [29], [30], [35], [41], [50], [63], [67], [72]–[74], [87], [102], [105]
5	Random Tree [79], [104]	Custom	Custom	Custom	Sudden; Reoccurring	II	[27], [35], [73], [82], [84]–[87]
6	LED [106]	Custom	24	10	Sudden	II	[23], [27], [35], [63], [73], [81], [82], [87], [99], [102]
7	Waveform [106]	Custom	40	3	Sudden	II	[18], [27], [78], [81]–[83], [87], [102]
8	Sine [20]	Custom	2	2	Sudden	II	[20], [21], [26], [29], [72], [107]
9	Circle [20]	Custom	2	2	Gradual	III	[20], [21], [26], [30], [41], [72], [101], [107]
10	Rotating chessboard [67]	Custom	2	2	Gradual	II	[13], [45], [51], [67], [107]

understanding the drift, and could introduce bias when comparing different machine learning models.

7 The Concept Drift Problem in Other Research Areas

We have observed that handling the concept drift problem is not a standalone research subject but has a large number of indirect usage scenarios. In this section, we adopt this new perspective to review recent developments in other research areas that benefit from handling the concept drift problem.

7.1 Class imbalance

Class imbalance is a common problem in stream data mining in addition to concept drift. Research effort has been made to develop effective learning algorithms to tackle both problems at same time. [117] presented two ensemble methods for learning under concept drift with imbalanced class. The first method, Learn++.CDS, is extended from Learn++.NSE and combined with the Synthetic Minority class Oversampling Technique (SMOTE). The second algorithm, Learn++.NIE, improves on the previous method by employing a different penalty constraint to prevent prediction accuracy bias and replacing SMOTE with bagging to avoid oversampling. ESOS-ELM [118] is another ensemble method which uses Online Sequential Extreme Learning Machine (OS-ELM) as a basic classifier to improve performance with class imbalanced data. A concept drift detector is integrated to retrain the classifier when drift occurs. The author then developed another algorithm [119], which is able to tackle multi-class imbalanced data with concept drift. [120] proposed two learning algorithms OOB and UOB, which build an ensemble model to overcome the class imbalance in real time through resampling and time-decayed metrics. [121] developed an ensemble method which handles concept drift and class imbalance with additional true label data limitation.

7.2 Big data mining

Data mining in big data environments faces similar challenges to stream data mining [122]: data is generated at a fast rate (Velocity) and distribution uncertainty always exists in the data, which means that handling concept drift is also crucial in big data applications. Additionally, scalability is an important consideration because in big data environments, a data stream may come in very large and potentially unpredictable quantities (Volume) and cannot be processed in a single computer server. An attempt to handle concept drift in a distributed computing environment was made by [123] in which an Online Map-Reduce Drift Detection Method (OMR-DDM) was proposed, using the combined online error rate of the parallel classification algorithms to identify the changes in a big data stream. A recent study [124] proposed another scalable stream data mining algorithm, called Micro-Cluster Nearest Neighbor (MC-NN), based on nearest neighbor classifier. This method extends the original Micro-Cluster algorithm [125] to adapt to concept drift by monitoring classification error. This micro-cluster algorithm was further extended to a parallel version using the map-reduce technique in [126] and applied to solve the label-drift classification problem where class labels are not known in advance [127].

7.3 Active learning and semi-supervised learning

Active learning is based on the assumption that there is a large amount of unlabeled data but only a fraction of them can be labeled by human effort. This is a common situation in stream data applications, which are often also subject to the concept drift problem. [115] presented a general framework that combines active learning and concept drift adaptation. It first compares different instance-sampling strategies for labeling to guarantee that the labeling cost will be under budget, and that distribution bias will be prevented. A drift adaptation mechanism is then adopted, based on the DDM detection method [20]. In [128], the authors proposed a new active learning algorithm that primarily aims to avoid bias in the sampling process of choosing instances for labeling. They also introduced a memory loss factor to the model, enabling it to adapt to concept drift.

Semi-supervised learning concerns how to use limited true label data more efficiently by leveraging unsupervised techniques. In this scenario, additional design effort is required to handle concept drift. For example, in [129], the authors applied a Gaussian Mixture model to both labeled and unlabeled data, and assigned labels, which has the ability to adapt to gradual drift. Similarly, [99], [130], [131] are all cluster-based semi-supervised ensemble methods that aim to adapt to drift with limited true label data. The latter

TABLE 4
List of real-world datasets for performance evaluation of learning under concept drift.

	Dataset	#Insts.	#Attrs.	#Cls.	Used by references
1	Airlines [104]	539384	7	2	[4], [5], [35], [73], [102], [114], [115]
2	Covertype [106]	581012	54	7	[13], [23], [35], [36], [59], [63], [73], [81]–[83], [86], [87], [102], [115]
3	Electricity [109]	45312	8	2	[4], [5], [13], [20], [23], [26], [29], [31], [35], [36], [57], [63], [72], [73], [78], [86], [87], [101], [102], [115]
4	Poker-Hand [106]	1025010	10	10	[13], [32], [63], [73], [102]
5	NOAA weather [67]	18159	8	2	[2], [4], [13], [67], [68], [78], [105]
6	Sensor [108]	2219803	5	54	[36], [78]
7	KDDCup'99 [106]	494021	41	23	[35], [47], [65], [69], [74], [84], [86], [99], [102]
8	Usenet1 [112]	1500	99	2	[23], [51], [87]
9	Usenet2 [112]	1500	99	2	[23], [87]
10	Email_data [110]	1500	913	2	[45], [76], [77]
11	Spam_data [110]	9324	499	2	[4], [5], [23], [36], [102], [116]
12	Spam assassin corpus [111]	9324	39916	2	[4], [35], [76], [87]
13	ECUE drift dataset 1 [113]	10983	287034	2	[2], [3]
14	ECUE drift dataset 2 [113]	11905	166047	2	[2], [3]

are also able to recognize recurring concepts. In [132], the author adopted a new perspective on the true label scarcity problem by considering the true labeled data and unlabeled data as two independent non-stationary data generating processes. Concept drift is handled asynchronously on these two streams. The SAND algorithm [133], [134] is another semi-supervised adaptive method which detects concept drift on cluster boundaries. There are also studies [90, 91] that focus on adapting to concept drift in circumstances where true label data is completely unavailable.

7.4 Decision Rules

Data-driven decision support systems need to be able to adapt to concept drift in order to make accurate decisions and decision rules is the main technique for this purpose. [102] proposed a decision rule induction algorithm, Very Fast Decision Rules (VFDR), to effectively process stream data. An extended version, Adaptive VFDR, was developed to handle concept drift by dynamically adding and removing decision rules according to their error rate which is monitored by drift detector. Instead of inducing rules from decision trees, [135] proposed another decision rule algorithm based on PRISM [136] to directly induce rules from data. This algorithm is also able to adapt to drift by monitoring the performance of each rule on a sliding window of latest data. [137] also developed an adaptive decision making algorithm based on fuzzy rules. The algorithm includes a rule pruning procedure, which removes obsolete rules to adapt to changes, and a rule recal procedure to adapt to recurring concepts.

This section by no means attempts to cover every research field in which concept drift handling is used. There are many other studies that also consider concept drift as a dual problem. For example, [138] is a dimension reduction algorithm to separate classes based on least squares linear discovery analysis (LSLDA), which is then extended to adapt to drift; [139] considered the concept drift problem in time series and developed an online explicit drift detection method by monitoring time series features; and [140] developed an incremental scaffolding classification algorithm for complex tasks that also involve concept drift.

8 Conclusions: Findings and Future Directions

We summarize the recent developments of concept drift research, and the following important findings can be extracted:

1) Error rate-based and data distribution-based drift detection methods are still playing a dominant role in concept drift detection research, while multiple hypothesis test methods emerge in recent years;
2) Regarding to concept drift understanding, all drift detection methods can answer "When", but very few methods have the ability to answer "How" and "Where";
3) Adaptive models and ensemble techniques have played an increasingly important role in recent concept drift adaptation developments. In contrast, research of retraining models with explicit drift detection has slowed;
4) Most existing drift detection and adaptation algorithms assume the ground true label is available after classification/prediction, or extreme verification latency. Very few research has been conducted to address unsupervised or semi-supervised drift detection and adaptation.
5) Some computational intelligence techniques, such as fuzzy logic, competence model, have been applied in concept drift;
6) There is no comprehensive analysis on real-world data streams from the concept drift aspect, such as the drift occurrence time, the severity of drift, and the drift regions.
7) An increasing number of other research areas have recognized the importance of handling concept drift, especially in big data community.

Based on these findings, we suggest four new directions in future concept drift research:

1) Drift detection research should not only focus on identifying drift occurrence time accurately, but also need to provide the information of drift severity and regions. These information could be utilized for better concept drift adaptation.
2) In the real-world scenario, the cost to acquire true label could be expensive, that is, unsupervised or semi-su-

pervised drift detection and adaptation could still be promising in the future.
3) A framework for selecting real-world data streams should be established for evaluating learning algorithms handling concept drift.
4) Research on effectively integrating concept drift handling techniques with machine learning methodologies for data-driven applications is highly desired.

We hope this paper can provide researchers with state-of-the-art knowledge on concept drift research developments and provide guidelines about how to apply concept drift techniques in different domains to support users in various prediction and decision activities.

ACKNOWLEDGMENTS

The work presented in this paper was supported by the Australian Research Council (ARC) under discovery grant DP150101645. We significantly thank Yiliao Song for her help in preparation of datasets and applications shown in Sections 6.

REFERENCES

[1] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Machine Learning*, vol. 23, no. 1, pp. 69–101, 1996.
[2] N. Lu, J. Lu, G. Zhang, and R. Lopez de Mantaras, "A concept drift-tolerant case-base editing technique," *Artif. Intell.*, vol. 230, pp. 108–133, 2016.
[3] N. Lu, G. Zhang, and J. Lu, "Concept drift detection via competence models," *Artif. Intell.*, vol. 209, pp. 11–28, 2014.
[4] A. Liu, Y. Song, G. Zhang, and J. Lu, "Regional concept drift detection and density synchronized drift adaptation," in *Proc. 26th Int. Joint Conf. Artificial Intelligence*. Accept, 2017, Conference Proceedings.
[5] A. Liu, G. Zhang, and J. Lu, "Fuzzy time windowing for gradual concept drift adaptation," in *Proc. 26th IEEE Int. Conf. Fuzzy Systems*. IEEE, 2017, Conference Proceedings.
[6] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woniak, "Ensemble learning for data stream analysis: A survey," *Information Fusion*, vol. 37, pp. 132–156, 2017.
[7] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, and F. Herrera, "A survey on data preprocessing for data stream mining: Current status and future directions," *Neurocomputing*, vol. 239, pp. 39–57, 2017.
[8] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 1–37, 2014.
[9] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in nonstationary environments: a survey," *IEEE Comput. Intell. Mag.*, vol. 10, no. 4, pp. 12–25, 2015.
[10] J. Gama, "A survey on learning from data streams: current and future trends," *Progress in Artificial Intelligence*, vol. 1, no. 1, pp. 45–55, 2012.
[11] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. P. L. F. d. Carvalho, and J. Gama, "Data stream clustering: A survey," *ACM Comput. Surv.*, vol. 46, no. 1, pp. 1–31, 2013.
[12] J. C. Schlimmer and R. H. Granger Jr, "Incremental learning from noisy data," *Machine learning*, vol. 1, no. 3, pp. 317–354, 1986.
[13] V. Losing, B. Hammer, and H. Wersing, "Knn classifier with self adjusting memory for heterogeneous concept drift," in *Proc. 16th Int. Conf. Data Mining*, 2016, Conference Proceedings, pp. 291–300.
[14] I. Žliobaitė and J. Hollmén, "Optimizing regression models for data streams with missing values," *Machine Learning*, vol. 99, no. 1, pp. 47–73, 2014.
[15] S. Amos, "When training and test sets are different: characterizing learning transfer," *Dataset Shift in Machine Learning*, pp. 3–28, 2009.
[16] J. G. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N. V. Chawla, and F. Herrera, "A unifying view on dataset shift in classification," *Pattern Recognit.*, vol. 45, no. 1, pp. 521–530, 2012.
[17] M. Basseville and I. V. Nikiforov, *Detection of abrupt changes: theory and application*. Prentice Hall Englewood Cliffs, 1993, vol. 104.
[18] A. Dries and U. Rückert, "Adaptive concept drift detection," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 2, no. 5–6, pp. 311–327, 2009.
[19] C. Alippi and M. Roveri, "Just-in-time adaptive classifiers part i: Detecting nonstationary changes," *IEEE Trans. Neural Networks*, vol. 19, no. 7, pp. 1145–1153, 2008.
[20] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Proc. 17th Brazilian Symp. Artificial Intelligence*, ser. Lecture Notes in Computer Science. Springer, 2004, Book Section, pp. 286–295.
[21] L. Bu, C. Alippi, and D. Zhao, "A pdf-free change detection test based on density difference estimation," *IEEE Trans. Neural Networks Learn. Syst.*, vol. PP, no. 99, pp. 1–11, 2016.
[22] T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi, "An information-theoretic approach to detecting changes in multi-dimensional data streams," in *Proc. Symp. the Interface of Statistics, Computing Science, and Applications*. Citeseer, 2006, Conference Proceedings, pp. 1–24.
[23] I. Frias-Blanco, J. d. Campo-Avila, G. Ramos-Jimenez, R. Morales-Bueno, A. Ortiz-Diaz, and Y. Caballero-Mota, "Online and non-parametric drift detection methods based on hoeffding's bounds," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 3, pp. 810–823, 2015.
[24] M. Yamada, A. Kimura, F. Naya, and H. Sawada, "Change-point detection with feature selection in high-dimensional time-series data," in *Proc. 23rd Int. Joint Conf. Artificial Intelligence*, 2013, Conference Proceedings, pp. 1827–1833.
[25] J. Gama and G. Castillo, "Learning with local drift detection," in *Proc. 2nd Int. Conf. Advanced Data Mining and Applications*. Springer, 2006, Conference Proceedings, pp. 42–55.
[26] M. Baena-García, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, and R. Morales-Bueno, "Early drift detection method," in *Proc. 4th Int. Workshop Knowledge Discovery from Data Streams*, 2006, Conference Paper.
[27] S. Xu and J. Wang, "Dynamic extreme learning machine for data stream classification," *Neurocomputing*, vol. 238, pp. 433–449, 2017.
[28] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.
[29] G. J. Ross, N. M. Adams, D. K. Tasoulis, and D. J. Hand, "Exponentially weighted moving average charts for detecting concept drift," *Pattern Recognit. Lett.*, vol. 33, no. 2, pp. 191–198, 2012.
[30] K. Nishida and K. Yamauchi, "Detecting concept drift using statistical testing," in *Proc. 10th Int. Conf. Discovery Science*, V. Corruble, M. Takeda, and E. Suzuki, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, Conference Proceedings, pp. 264–269.
[31] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," in *Proc. 2007 SIAM Int. Conf. Data Mining*, vol. 7. SIAM, 2007, Conference Proceedings, p. 2007.
[32] ——, "Adaptive learning from evolving data streams," in *Proc. 8th Int. Symp. Intelligent Data Analysis*. Springer, 2009, Conference Proceedings, pp. 249–260.
[33] A. Bifet, G. Holmes, B. Pfahringer, and R. Gavaldà, "Improving adaptive bagging methods for evolving data streams," in *Proc. 1st Asian Conf. Machine Learning*, ser. Lecture Notes in Computer Science, Z.-H. Zhou and T. Washio, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, Book Section, pp. 23–37.
[34] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà, "New ensemble methods for evolving data streams," in *Proc. 15th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*. ACM, 2009, Conference Proceedings, pp. 139–148.
[35] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, and T. Abdessalem, "Adaptive random forests for evolving data stream classification," *Machine Learning*, 2017.
[36] J. Shao, Z. Ahmadi, and S. Kramer, "Prototype-based learning on concept-drifting data streams," in *Proc. 20th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*. 2623609: ACM, 2014, Conference Proceedings, pp. 412–421.

[37] D. Kifer, S. Ben-David, and J. Gehrke, "Detecting change in data streams," in *Proc. 30th Int. Conf. Very Large Databases*, vol. 30. VLDB Endowment, 2004, Conference Proceedings, pp. 180–191.

[38] X. Song, M. Wu, C. Jermaine, and S. Ranka, "Statistical change detection for multi-dimensional data," in *Proc. 13th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*. San Jose, California, USA: ACM, 2007, Conference Paper, pp. 667–676.

[39] A. A. Qahtan, B. Alharbi, S. Wang, and X. Zhang, "A pca-based change detection framework for multidimensional data streams," in *Proc. 21th Int. Conf. on Knowledge Discovery and Data Mining*. ACM, 2015, Conference Proceedings, pp. 935–944.

[40] F. Gu, G. Zhang, J. Lu, and C.-T. Lin, "Concept drift detection based on equal density estimation," in *Proc. 2016 Int. Joint Conf. Neural Networks*. IEEE, 2016, Conference Proceedings, pp. 24–30.

[41] L. Bu, D. Zhao, and C. Alippi, "An incremental change detection test based on density difference estimation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. PP, no. 99, pp. 1–13, 2017.

[42] C. Alippi and M. Roveri, "Just-in-time adaptive classifiers part ii: designing the classifier," *IEEE Trans. Neural Networks*, vol. 19, no. 12, pp. 2053–2064, 2008.

[43] C. Alippi, G. Boracchi, and M. Roveri, "A just-in-time adaptive classification system based on the intersection of confidence intervals rule," *Neural Networks*, vol. 24, no. 8, pp. 791–800, 2011.

[44] ——, "Just-in-time ensemble of classifiers," in *Proc. 2012 Int. Joint Conf. Neural Networks*. IEEE, 2012, Conference Proceedings, pp. 1–8.

[45] ——, "Just-in-time classifiers for recurrent concepts," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 24, no. 4, pp. 620–634, 2013.

[46] W. Heng and Z. Abraham, "Concept drift detection for streaming data," in *Proc. 2015 Int. Joint Conf. Neural Networks*, 2015, Conference Proceedings, pp. 1–9.

[47] Y. Zhang, G. Chu, P. Li, X. Hu, and X. Wu, "Three-layer concept drifting detection in text data streams," *Neurocomputing*, vol. 260, pp. 393–403, 2017.

[48] L. Du, Q. Song, L. Zhu, and X. Zhu, "A selective detector ensemble for concept drift detection," *The Computer Journal*, vol. 58, no. 3, pp. 457–471, 2014.

[49] B. I. F. Maciel, S. G. T. C. Santos, and R. S. M. Barros, "A lightweight concept drift detection ensemble," in *Proc. 27th IEEE Int. Conf. on Tools with Artificial Intelligence*. IEEE, 2015, pp. 1061–1068.

[50] C. Alippi, G. Boracchi, and M. Roveri, "Hierarchical change-detection tests," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 28, no. 2, pp. 246–258, 2017.

[51] S. Yu and Z. Abraham, "Concept drift detection with hierarchical hypothesis testing," in *Proc. 2017 SIAM Int. Conf. Data Mining*. SIAM, 2017, Conference Proceedings, pp. 768–776.

[52] H. Raza, G. Prasad, and Y. Li, "Ewma model based shift-detection methods for detecting covariate shifts in non-stationary environments," *Pattern Recognit.*, vol. 48, no. 3, pp. 659–669, 2015.

[53] S. Yu, X. Wang, and J. C. Principe, "Request-and-reverify: Hierarchical hypothesis testing for concept drift detection with expensive labels," *arXiv preprint arXiv:1806.10131*, 2018.

[54] P. M. Gonçalves Jr, S. G. de Carvalho Santos, R. S. Barros, and D. C. Vieira, "A comparative study on concept drift detectors," *Expert Systems with Applications*, vol. 41, no. 18, pp. 8144–8156, 2014.

[55] F. Pukelsheim, "The three sigma rule," *The American Statistician*, vol. 48, no. 2, pp. 88–91, 1994.

[56] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen, "Dynamic integration of classifiers for handling concept drift," *Information Fusion*, vol. 9, no. 1, pp. 56–68, 2008.

[57] S. H. Bach and M. Maloof, "Paired learners for concept drift," in *Proc. 8th Int. Conf. Data Mining*, 2008, Conference Proceedings, pp. 23–32.

[58] D. Liu, Y. Wu, and H. Jiang, "Fp-elm: An online sequential learning algorithm for dealing with concept drift," *Neurocomputing*, vol. 207, pp. 322–334, 2016.

[59] D. Han, C. Giraud-Carrier, and S. Li, "Efficient mining of high-speed uncertain data streams," *Applied Intelligence*, vol. 43, no. 4, pp. 773–785, 2015.

[60] S. G. Soares and R. Araújo, "An adaptive ensemble of on-line extreme learning machines with variable forgetting factor for dynamic system prediction," *Neurocomputing*, vol. 171, pp. 693–707, 2016.

[61] B. F. J. Manly and D. Mackenzie, "A cumulative sum type of method for environmental monitoring," *Environmetrics*, vol. 11, no. 2, pp. 151–166, 2000.

[62] N. C. Oza and S. Russell, "Experimental comparisons of online and batch versions of bagging and boosting," in *Proc. 7th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*. 502565: ACM, 2001, Conference Proceedings, pp. 359–364.

[63] A. Bifet, G. Holmes, and B. Pfahringer, "Leveraging bagging for evolving data streams," in *Proc. 2010 Joint European Conf. Machine Learning and Knowledge Discovery in Databases*. Springer, 2010, Conference Proceedings, pp. 135–150.

[64] F. Chu and C. Zaniolo, "Fast and light boosting for adaptive mining of data streams," in *Proc. 8th Pacific-Asia Conf. Knowledge Discovery and Data Mining*, H. Dai, R. Srikant, and C. Zhang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, Book Section, pp. 282–292.

[65] P. Li, X. Wu, X. Hu, and H. Wang, "Learning concept-drifting data streams with random ensemble decision trees," *Neurocomputing*, vol. 166, pp. 68–83, 2015.

[66] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *Journal of Machine Learning Research*, 2007.

[67] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Trans. Neural Networks*, vol. 22, no. 10, pp. 1517–31, 2011.

[68] X.-C. Yin, K. Huang, and H.-W. Hao, "De2: Dynamic ensemble of ensembles for learning nonstationary data," *Neurocomputing*, vol. 165, pp. 14–22, 2015.

[69] P. Zhang, J. Li, P. Wang, B. J. Gao, X. Zhu, and L. Guo, "Enabling fast prediction for ensemble models on data streams," in *Proc. 17th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*. San Diego, California, USA: ACM, 2011, Conference Paper, pp. 177–185.

[70] Y. Xu, R. Xu, W. Yan, and P. Ardis, "Concept drift learning with alternating learners," in *Proc. 2017 Int. Joint Conf. Neural Networks*, 2017, Conference Proceedings, pp. 2104–2111.

[71] L. Pietruczuk, L. Rutkowski, M. Jaworski, and P. Duda, "A method for automatic adjustment of ensemble size in stream data mining," in *Proc. 2016 Int. Joint Conf. Neural Networks*, 2016, Conference Proceedings, pp. 9–15.

[72] S.-C. You and H.-T. Lin, "A simple unlearning framework for online learning under concept drifts," in *Proc. 20th Pacific-Asia Conf. Knowledge Discovery and Data Mining*. Springer, 2016, Conference Proceedings, pp. 115–126.

[73] D. Brzezinski and J. Stefanowski, "Reacting to different types of concept drift: The accuracy updated ensemble algorithm," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 25, no. 1, pp. 81–94, 2014.

[74] P. Zhang, X. Zhu, and Y. Shi, "Categorizing and mining concept drifting data streams," in *Proc. 14th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*. Las Vegas, Nevada, USA: ACM, 2008, Conference Paper, pp. 812–820.

[75] Y. Sun, K. Tang, L. L. Minku, S. Wang, and X. Yao, "Online ensemble learning of data streams with gradually evolved classes," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 6, pp. 1532–1545, 2016.

[76] J. Gama and P. Kosina, "Recurrent concepts in data streams classification," *Knowledge and Information Systems*, vol. 40, no. 3, pp. 489–507, 2013.

[77] J. B. Gomes, M. M. Gaber, P. A. Sousa, and E. Menasalvas, "Mining recurring concepts in a dynamic feature space," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 25, no. 1, pp. 95–110, 2014.

[78] Z. Ahmadi and S. Kramer, "Modeling recurring concepts in data streams: a graph-based framework," *Knowledge and Information Systems*, 2017.

[79] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proc. 6th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*. ACM, 2000, Conference Proceedings, pp. 71–80.

[80] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proc. 7th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*. San Francisco, California: ACM, 2001, Conference Paper, pp. 97–106.

[81] J. Gama, R. Rocha, and P. Medas, "Accurate decision trees for mining high-speed data streams," in *Proc. 9th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*. ACM, 2003, Conference Proceedings, pp. 523–528.

[82] H. Yang and S. Fong, "Incrementally optimized decision tree for noisy big data," in *Proc. 1st Int. Workshop Big Data, Streams and Heterogeneous Source Mining Algorithms, Systems, Programming*

Models and Applications. Beijing, China: ACM, 2012, Conference Paper, pp. 36–44.

[83] ——, "Countering the concept-drift problems in big data by an incrementally optimized stream mining model," *Journal of Systems and Software*, vol. 102, pp. 158–166, 2015.

[84] L. Rutkowski, M. Jaworski, L. Pietruczuk, and P. Duda, "Decision trees for mining data streams based on the gaussian approximation," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 1, pp. 108–119, 2014.

[85] L. Rutkowski, L. Pietruczuk, P. Duda, and M. Jaworski, "Decision trees for mining data streams based on the mcdiarmid's bound," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 6, pp. 1272–1279, 2013.

[86] L. Rutkowski, M. Jaworski, L. Pietruczuk, and P. Duda, "A new method for data stream mining based on the misclassification error," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 26, no. 5, pp. 1048–1059, 2015.

[87] I. Frías-Blanco, J. d. Campo-Ávila, G. Ramos-Jiménez, A. C. P. L. F. Carvalho, A. Ortiz-Díaz, and R. Morales-Bueno, "Online adaptive decision trees based on concentration inequalities," *Knowledge-Based Systems*, vol. 104, pp. 179–194, 2016.

[88] J. Gama, R. Sebastião, and P. P. Rodrigues, "On evaluating stream learning algorithms," *Machine Learning*, vol. 90, no. 3, pp. 317–346, 2012.

[89] I. Žliobaitė, "Controlled permutations for testing adaptive learning models," *Knowledge and Information Systems*, vol. 39, no. 3, pp. 565–578, 2014.

[90] A. Bifet, G. Holmes, B. Pfahringer, and E. Frank, "Fast perceptron decision tree learning from evolving data streams," in *Proc. 14th Pacific-Asia Conf. Knowledge Discovery and Data Mining*, M. J. Zaki, J. X. Yu, B. Ravindran, and V. Pudi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, Book Section, pp. 299–310.

[91] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, 1960.

[92] I. Žliobaitė, A. Bifet, J. Read, B. Pfahringer, and G. Holmes, "Evaluation methods and decision theory for classification of streaming data with temporal dependence," *Machine Learning*, vol. 98, no. 3, pp. 455–482, 2015.

[93] D. Brzezinski and J. Stefanowski, "Prequential auc for classifier evaluation and drift detection in evolving data streams," in *Proc. 3rd Int. Workshop New Frontiers in Mining Complex Patterns*, A. Appice, M. Ceci, C. Loglisci, G. Manco, E. Masciari, and Z. W. Ras, Eds. Cham: Springer International Publishing, 2014, Book Section, pp. 87–101.

[94] A. Bifet, G. d. F. Morales, J. Read, G. Holmes, and B. Pfahringer, "Efficient online evaluation of big data stream classifiers," in *Proc. 21th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*. Sydney, NSW, Australia: ACM, 2015, Conference Paper, pp. 59–68.

[95] N. Japkowicz and M. Shah, *Evaluating learning algorithms: a classification perspective*. Cambridge University Press, 2011.

[96] Q. McNemar, "Note on the sampling error of the difference between correlated proportions or percentages," *Psychometrika*, vol. 12, no. 2, pp. 153–157, 1947.

[97] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, no. Jan, pp. 1–30, 2006.

[98] J. Z. Kolter and M. A. Maloof, "Using additive expert ensembles to cope with concept drift," in *Proc. 22nd Int. Conf. Machine Learning*. Bonn, Germany: ACM, 2005, Conference Paper, pp. 449–456.

[99] X. Wu, P. Li, and X. Hu, "Learning from concept drifting data streams with unlabeled data," *Neurocomputing*, vol. 92, pp. 145–155, 2012.

[100] W. N. Street and Y. Kim, "A streaming ensemble algorithm (sea) for large-scale classification," in *Proc. Seventh ACM Int. Conf. Knowledge Discovery and Data Mining*. 502568: ACM, 2001, Conference Proceedings, pp. 377–382.

[101] R. Fok, A. An, and X. Wang, "Mining evolving data streams with particle filters," *Comput. Intell.*, vol. 33, no. 2, pp. 147–180, 2017.

[102] P. Kosina and J. Gama, "Very fast decision rules for classification in data streams," *Data Mining and Knowledge Discovery*, vol. 29, no. 1, pp. 168–202, 2015.

[103] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proc. 9th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*. Washington, D.C.: ACM, 2003, Conference Paper, pp. 226–235.

[104] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "Moa: Massive online analysis," *Journal of Machine Learning Research*, vol. 99, pp. 1601–1604, 2010.

[105] V. M. Souza, D. F. Silva, J. Gama, and G. E. Batista, "Data stream classification guided by clustering on nonstationary environments and extreme verification latency," in *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM, 2015, pp. 873–881.

[106] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[107] M. Harel, S. Mannor, R. El-Yaniv, and K. Crammer, "Concept drift detection through resampling," in *Proc. 31st Int. Conf. Machine Learning*, 2014, Conference Proceedings, pp. 1009–1017.

[108] X. Zhu, "Stream data mining repository," 2010. [Online]. Available: http://www.cse.fau.edu/~xqzhu/stream.html

[109] M. Harries and N. S. Wales, "Splice-2 comparative evaluation: Electricity pricing," 1999.

[110] I. Katakis, G. Tsoumakas, and I. Vlahavas, "Tracking recurring contexts using ensemble classifiers: an application to email filtering," *Knowledge and Information Systems*, vol. 22, no. 3, pp. 371–391, 2009.

[111] I. Katakis, G. Tsoumakas, E. Banos, N. Bassiliades, and I. Vlahavas, "An adaptive personalized news dissemination system," *Journal of Intelligent Information Systems*, vol. 32, no. 2, pp. 191–212, 2008.

[112] I. Katakis, G. Tsoumakas, and I. P. Vlahavas, "An ensemble of classifiers for coping with recurring contexts in data streams," in *18th European Conf. Artificial Intelligence*, 2008, Conference Proceedings, pp. 763–764.

[113] S. J. Delany, P. Cunningham, A. Tsymbal, and L. Coyle, "A case-based technique for tracking concept drift in spam filtering," *Knowledge-Based Systems*, vol. 18, no. 4–5, pp. 187–195, 2005.

[114] L.-Y. Wang, C. Park, K. Yeon, and H. Choi, "Tracking concept drift using a constrained penalized regression combiner," *Comput. Stat. Data Anal.*, vol. 108, pp. 52–69, 2017.

[115] I. Zliobaite, A. Bifet, B. Pfahringer, and G. Holmes, "Active learning with drifting streaming data," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 25, no. 1, pp. 27–39, 2014.

[116] G. Song, Y. Ye, H. Zhang, X. Xu, R. Y. K. Lau, and F. Liu, "Dynamic clustering forest: An ensemble framework to efficiently classify textual data stream with concept drift," *Information Sciences*, vol. 357, pp. 125–143, 2016.

[117] G. Ditzler and R. Polikar, "Incremental learning of concept drift from streaming imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 10, pp. 2283–2301, 2013.

[118] B. Mirza, Z. Lin, and N. Liu, "Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift," *Neurocomputing*, vol. 149, pp. 316–329, 2015.

[119] B. Mirza and Z. Lin, "Meta-cognitive online sequential extreme learning machine for imbalanced and concept-drifting data classification," *Neural Networks*, vol. 80, pp. 79–94, 2016.

[120] S. Wang, L. L. Minku, and X. Yao, "Resampling-based ensemble methods for online class imbalance learning," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 5, pp. 1356–1368, 2015.

[121] E. Arabmakki and M. Kantardzic, "Som-based partial labeling of imbalanced data stream," *Neurocomputing*, vol. 262, pp. 120–133, 2017.

[122] A. Katal, M. Wazid, and R. H. Goudar, "Big data: Issues, challenges, tools and good practices," in *Proc. 6th Int. Conf. Contemporary Computing (IC3)*, 2013, Conference Proceedings, pp. 404–409.

[123] A. Andrzejak and J. B. Gomes, "Parallel concept drift detection with online map-reduce," in *Proc. 12th Int. Conf. Data Mining Workshops*, 2012, Conference Proceedings, pp. 402–407.

[124] M. Tennant, F. Stahl, O. Rana, and J. B. Gomes, "Scalable real-time classification of data streams with concept drift," *Future Generation Computer Systems*, vol. 75, pp. 187–199, 2017.

[125] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *Proc. 29th Int. Conf. Very Large Databases*, vol. 29. VLDB Endowment, 2003, Conference Proceedings, pp. 81–92.

[126] X. Song, H. He, S. Niu, and J. Gao, "A data streams analysis strategy based on hoeffding tree with concept drift on hadoop system," in *Proc. 4th Int. Conf. Advanced Cloud and Big Data*, 2016, Conference Proceedings, pp. 45–48.

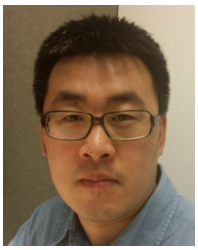
[127] V. Nguyen, T. D. Nguyen, T. Le, S. Venkatesh, and D. Phung, "One-pass logistic regression for label-drift and large-scale clas-

sification on distributed systems," in *Proc. 16th Int. Conf. Data Mining*, 2016, Conference Proceedings, pp. 1113–1118.
[128] W. Chu, M. Zinkevich, L. Li, A. Thomas, and B. Tseng, "Unbiased online active learning in data streams," in *Proc. 17th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*. San Diego, California, USA: ACM, 2011, Conference Paper, pp. 195–203.
[129] G. Ditzler and R. Polikar, "Semi-supervised learning in non-stationary environments," in *Proc. 2011 Int. Joint Conf. Neural Networks*, 2011, Conference Proceedings, pp. 2741–2748.
[130] M. J. Hosseini, A. Gholipour, and H. Beigy, "An ensemble of cluster-based classifiers for semi-supervised classification of non-stationary data streams," *Knowledge and Information Systems*, vol. 46, no. 3, pp. 567–597, 2015.
[131] P. Zhang, X. Zhu, J. Tan, and L. Guo, "Classifier and cluster ensembles for mining concept drifting data streams," in *Proc. 10th Int. Conf. Data Mining*, 2010, Conference Proceedings, pp. 1175–1180.
[132] S. Chandra, A. Haque, L. Khan, and C. Aggarwal, "An adaptive framework for multistream classification," in *Proc. 25th ACM Int. on Conf. Information and Knowledge Management*. Indianapolis, Indiana, USA: ACM, 2016, Conference Paper, pp. 1181–1190.
[133] A. Haque, L. Khan, M. Baron, B. Thuraisingham, and C. Aggarwal, "Efficient handling of concept drift and concept evolution over stream data," in *Proc. 32nd Int. Conf. Data Engineering*, 2003, Conference Proceedings, pp. 481–492.
[134] A. Haque, L. Khan, and M. Baron, "Sand: Semi-supervised adaptive novel class detection and classification over data stream," in *30th AAAI Conf. Artificial Intelligence*, 2016, Conference Proceedings, pp. 1652–1658.
[135] T. Le, F. Stahl, M. M. Gaber, J. B. Gomes, and G. D. Fatta, "On expressiveness and uncertainty awareness in rule-based classification for data streams," *Neurocomputing*, vol. 265, pp. 127–141, 2017.
[136] J. Cendrowska, "Prism: An algorithm for inducing modular rules," *Int. J. Man Mach. Stud.*, vol. 27, no. 4, pp. 349–370, 1987.
[137] M. Pratama, S. G. Anavatti, M. Joo, and E. D. Lughofer, "pclass: An effective classifier for streaming examples," *IEEE Trans. Fuzzy Syst.*, vol. 23, no. 2, pp. 369–386, 2015.
[138] Y.-R. Yeh and Y.-C. F. Wang, "A rank-one update method for least squares linear discriminant analysis with concept drift," *Pattern Recognit.*, vol. 46, no. 5, pp. 1267–1276, 2013.
[139] R. C. Cavalcante, L. L. Minku, and A. L. I. Oliveira, "Fedd: Feature extraction for explicit concept drift detection in time series," in *Proc. 2016 Int. Joint Conf. Neural Networks*, 2016, Conference Proceedings, pp. 740–747.
[140] M. Pratama, J. Lu, E. Lughofer, G. Zhang, and S. Anavatti, "Scaffolding type-2 classifier for incremental learning under concept drifts," *Neurocomputing*, vol. 191, pp. 304–329, 2016.

Jie Lu is a Distinguished Professor, Director of Centre for Artificial Intelligence, and Associate Dean Research with in the Faculty of Engineering and Information Technology at the University of Technology Sydney. Her research interests lie in the area of decision support systems, concept drift, fuzzy transfer learning, and recommender systems. She has published 10 research books and 400 papers, won 8 Australian Research Council discovery grants and 20 other grants. She serves as Editor-In-Chief for KBS and IJCIS, and delivered 16 keynotes in international conferences.

Anjin Liu is a Postdoctoral Research Associate in the A/DRsch Centre for Artificial Intelligence, Faculty of Engineering and Information Technology, University of Technology Sydney. He received the BIT degree (Honour) at the University of Sydney in 2012. His research interests include concept drift detection, adaptive data stream learning, multi-stream learning, machine learning and big data analytics

Fan Dong is Research Fellow of Centre for Artificial Intelligence, University of Technology Sydney. He received the dual Ph.D. degree from University of Technology Sydney and Beijing Institute of Technology in 2018. His research interests include concept drift detection, adaptive learning under concept drift and data stream mining.



Feng Gu is a Ph.D. candidate at the Faculty of Engineering and Information Technology, the University of Technology Sydney, NSW, Australia. He received bachelors degree of software engineering at Zhejiang University, China, in 2012. His research interests include stream data mining, adaptive learning under concept drift and evolving data.

João Gama is an Associate Professor at the University of Porto, Portugal. He is also a senior researcher and member of the board of directors of the Laboratory of Artificial Intelligence and Decision Support (LIAAD), a group belonging to INESC Porto. He serves as the member of the Editorial Board of Machine Learning Journal, Data Mining and Knowledge Discovery, Intelligent Data Analysis and New Generation Computing. His main research interest is in knowledge discovery from data streams and evolving data. He has published more than 200 papers and a recent book on Knowledge Discovery from Data Streams. He has extensive publications in the area of data stream learning.

Guangquan Zhang is an Associate Professor, and the Director of Decision System and e-Service Intelligence (DeSI) lab with in the Centre for Artificial Intelligence, in the Faculty of Engineering and Information Technology at the University of Technology Sydney. His main research interests lie in the area of uncertain information processing, fuzzy decision making, concept drift and fuzzy transfer learning. He has published 4 monographs and over 400 papers in refereed journals, conference proceedings and book chapters. He has won 7 Australian Research Council discovery grants and guest edited many special issues for international journals.

Data Drift vs. Concept Drift

Philip Tannor | October 06, 2021 | 8 mins

Table of Contents

Introduction

Putting Machine Learning (ML) models into production is a great achievement, but your work does not stop there. The performance of your models may degrade over time due to a concept called “model drift.” Your model in production is constantly receiving new data to make predictions upon. However, this data might have a different probability distribution than the one you trained the model on. Using the original model with the new data distribution will cause a drop in model performance. To avoid performance degradation, you need to monitor the changes in your model’s performance.

ML model drift is a situation where a model’s performance degrades over time, causing the model to start giving poor predictions. ML model drift can be categorized into two broad categories: concept drift and data drift.

This article explains the core ideas behind data drift vs. concept drift. It covers what they are, the reasons behind them, their differences, and how to detect and handle them in an ML project.

A Note About Terminology

There is confusion about the terminology when you read about concept and data drift for many reasons.

Machine Learning is a new and dynamically growing area in the software engineering discipline, with novel ideas coming up every day in different research and business domains.

The definitions differ because of the different research, textbook, and production environments people work with. For example, “concept drift” is used as an umbrella term in online learning. However, batch learning papers refer to the same thing as “dataset drift” (e.g., here and here).

This blog post uses the terms “concept drift” and “data drift”, following widely accepted Machine Learning engineering conventions, and denotes alternatives to clarify their relationship with each other.

Concept Drift in Machine Learning

To understand what concept drift is, we need to define "Concept" within the context. Concept stands for the joint probability distribution of a Machine Learning model's inputs (X) and outputs (Y). We can express their relationship in the following form:

$P(X, Y) = P(Y)\ P(X|Y) = P(X)\ P(Y|X)$

Concept drift can originate from any of the concept components. The most important source is the posterior class probability P(Y|X), as it shows how well our model understands the relationship between inputs and outputs. For this reason, people use the term "concept drift" or "real concept drift" for this specific type.

Concept shift/drift happens when posterior probabilities of X and Y, that is the probability of Y as output given X as input changes.

$P_{t1}(Y|X) \neq P_{t2}(Y|X)$

Where:
t1 = initial time
t2 = final time

(Real) concept drift is the situation when the functional relationship between the model inputs and outputs changes. The context has changed, but the model doesn't know about the change. Its learned patterns do not hold anymore.

Other terms for concept shift are class drift, real concept drift, or posterior probability shift.

The cause of the relationship change is usually some kind of external event/process or change in the real world. For example, if we try to predict life expectancy using geographic regions as input. As the region's development level increases (or decreases), the effect of the region on life expectancy changes, causing the model's prediction to not hold true anymore.

This mechanism is also behind the original understanding of "concept drift," the change in the "meaning" of predicted labels. A common example is the shifting view of what emailing behavior we consider "normal" or "spam." Sending emails frequently and in mass was a clear sign of spamming behavior a few years ago. Today, this is much less so. Models using these attributes to identify spam experienced concept drift and had to be retrained.

Other examples of concept change:

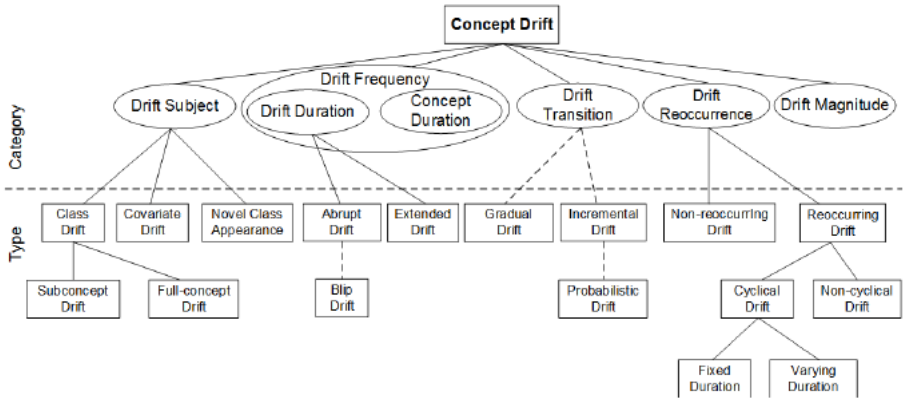
- The effect of the tax law change on predicting tax compliance.
- Changing consumer preferences when predicting products bought.
- Predicting company profits after a financial crisis.

It is also useful to look at the other statistical components as they can affect model performance or help predict the presence of "real concept drift." For this reason, we can distinguish the following additional sources of drift:

- "Data drift": Covariates P(X)
- "Label drift": Prior probabilities P(Y)
- Conditional covariates: P(X|Y)

However, it is important to note that some of these probabilities affect each other because of their relationship (e.g., for P(Y) to change, P(X) or P(Y|X) also has to change).

There are other ways to categorize concept drift, like frequency, transition speed, magnitude, recurrence, etc. This graph gives a good overview. Some of the terms in the graph are discussed later in this article.



Source

Testing. CI/CD. Monitoring.

Because ML systems are more fragile than you think. All based on our open-source core.

Install Open SourceBook a Demo

Data Drift in Machine Learning

Data drift is the situation where the model's input distribution changes.

$P_{t1}(X) \neq P_{t2}(X)$

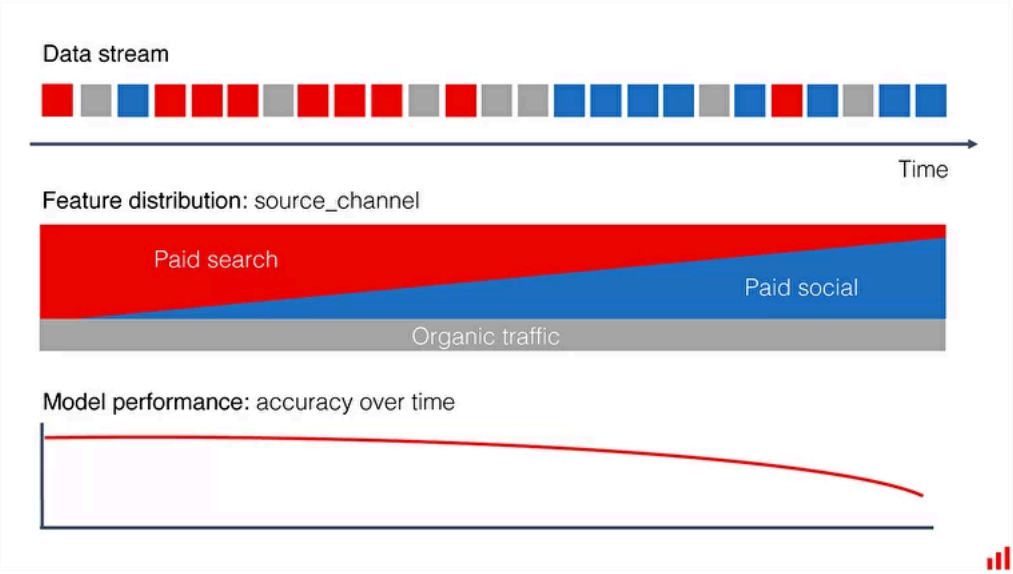
People also call data drift covariate shift, virtual drift, or virtual concept drift, depending on their definition of “concept’.”Other terms are feature drift or population drift.

So what does data drift mean? A helpful way to think about this is to consider feature segments. A segment here refers to a specific categorical value or a continuous value range in our input features. Example segments are an age group, a geographical origin, or customers from particular marketing channels.

Let’s say our model works well on the entire dataset but does not produce good results on a specific segment (e.g., because of little exposure). Low segment-level performance is not a problem if the segment’s proportion is small, and we aim only for aggregate results.

However, if our overall performance drops when our model receives new data with a high proportion of the poorly predicted segments. The input distribution shift makes the model less capable of predicting labels.

In the diagram below, we see that at the start time, the data stream consists of user traffic from paid search and organic traffic. Over time, the user traffic changes (maybe due to a social media marketing campaign) and we see more paid social media traffic and less paid search. If the model was being used to predict the number of users expected on an app, its performance would drop because it was not trained on a data set with a feature showing a lot of paid social media traffic.



Source

Data drift does not mean a change in the relationship between the input variables and the output. The model’s performance weakens because it receives data on which it hasn’t been trained enough. It can also occur if the data received by the model at inference time contains features that were not present during training.

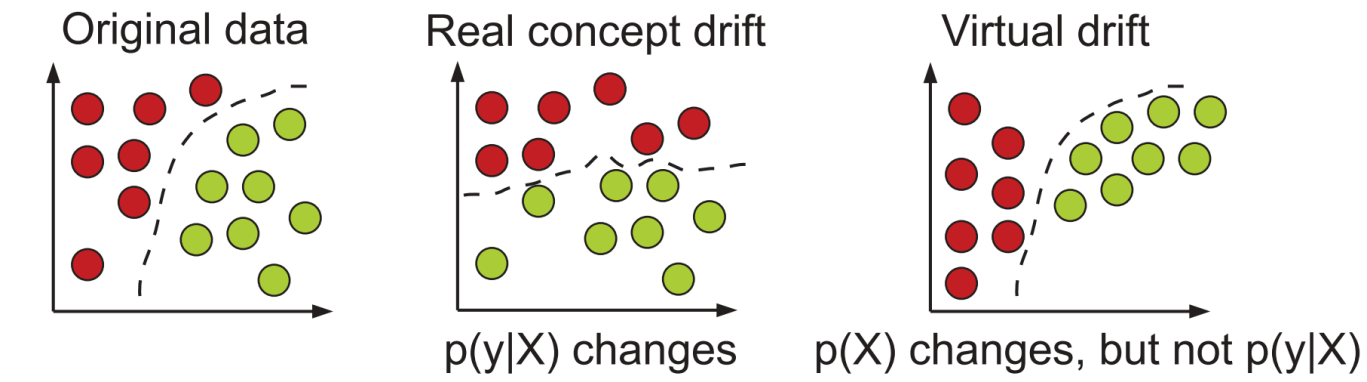
Performance degrades in proportion to the importance of that particular feature. For example, if we try to predict cancer and train our model on non-smokers, introducing a significant smoking subpopulation can alter the results.

The main cause of data drift is the increased relevance of “under-learned” segments. This can occur via different mechanisms:

- Sample selection bias: We have a systematic flaw in data collection, labelling, or introducing a sample selection.
- Non-stationary environment: The training and testing environments differ (e.g., temporal change or using the model on new geography).
- Upstream data transformation: Upstream data processing changes affect feature value distributions. (Many identify this as a different drift type, depending on their definition).

The Difference Between Data Drift And Real Concept Drift

1. In (Real) concept drift, the decision boundary P(Y|X) changes while, in the case of data drift (or virtual drift), the boundary remains the same even though P(X) has changed.



Source

2. Another difference is that in data drift, the cause is somewhat internal to the process of collecting and processing data and training our model on it. In the case of concept drift, the reason is usually an external event.
3. With data drift only the features are affected, while with concept drift, either the labels or the features or both are affected.

How to Detect Concept Drift and Data Drift

One way to detect model drift is through user feedback. However, in high-value models, it may not be ideal for the model performance depreciation to be experienced by the users before it is detected. When an ML model is deployed to production, it needs to be monitored. The following methods can be used to detect drift:

- Performance monitoring: When your data contains labels, you can use this method. To accomplish this, simply monitor model performance metrics such as accuracy, precision, and a variety of statistical measures. Deepchecks provides out-of-the-box methods for quickly obtaining reports on your model's performance and the presence of concept or data drift. You can also develop your own custom metrics based on the requirements of your model. This paper on concept drift adaptation discusses some approaches that you can try.

- Data monitoring: If your data has no labels, you can detect data drift by monitoring and comparing the statistical properties of both training and production data. These properties might include distributions, robustness, completeness, etc. Metrics are oftentimes set by the data science team to enable the monitoring tool to alert them when anything changes beyond a threshold. Some metrics include Population Stability Index (PSI), Jensen-Shannon (JS), among others.

How to Handle Concept Drift and Data Drift

To handle model drift in Machine Learning, you can use one or more of these strategies, depending on the cause of the drift:

- Retrain or adapt your Machine Learning model: If the drift is a result of changes in the distribution of your data, you can either decide to retrain your model or adapt your model by adjusting model parameters like the training weights to account for changes in the information being carried by the data features in the model.
- Update your training data with new data that carry current information about the relationship between the input and output data and retrain your model.
- Scheduled model management if it is a seasonal drift.
- Update your data pipeline: There may be data discrepancies due to some problems with the Data Engineering, for example, a change in the data schema from the API serving your ML model. This demands that you make the required changes to ensure that your model gets the data in the structure and format it was trained on.
- Stay up to date on changes to the data schema and always update model accordingly. Let's say you work with data on the volumes of substances produced at your organization. If the unit of measurement changes from litres to cubic metres, you should be aware in other to make appropriate changes (like retraining your model) as the expected values of your data change.
- Online training: The advantage of online training is that your model stays updated irrespective of any changes that occur. If you observe that your model is regularly drifting, you can decide to adopt online training where you train your model as new data enters the system. This is advisable if it is cost-effective to train the model continuously; it also depends on the data latency.
- When the drift is a result of sample selection bias, if possible, collect data points that are representative of the necessary patterns in the data.

Monitor Data Drift and Concept Drift in Your Machine Learning Workflow

To maintain the performance of your models, you need to prevent data and concept drift. To do that, you need to monitor your model to identify them in advance.

Since there are different drift types, you need to implement monitoring functions for each. You can do this manually, or you can use a Machine Learning monitoring framework like Deepchecks for that.

Deepchecks provides a framework for monitoring ml models in production. It helps you detect AI model drift so you can implement drift handles before they occur and degrade the performance of your models. Deepchecks gives a full report showing graphs (and their explanations) that visualize the presence of drift in your feature and label data.

Would you like to learn more? Check out our case studies.
Testing. CI/CD. Monitoring.

Because ML systems are more fragile than you think. All based on our open-source core.

Install Open SourceBook a Demo

Share

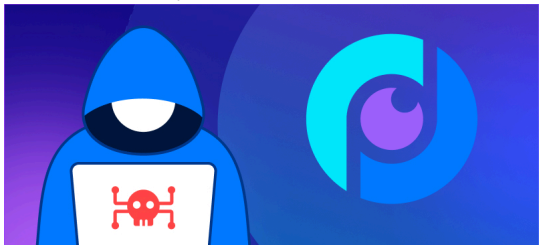
Recent Blog Posts

, 2025 10 mins
ating LLM Evaluations into CI/CD Pipelines

Deepchecks Community Blog

June 11, 2025 3.5 mins
Deepchecks Integrates with NVIDIA Enterprise AI Factory Validated Design for Iterative LLM Evaluation

Philip Tannor



June 05, 2025 8.5 mins
Red Teaming LLMs: The Ultimate Step-by-Step Guide to Securing AI Systems

Deepchecks Community Blog

- Products
- LLM Evaluation
- ML Monitoring
- Open-Source Testing
- Pricing
- Book a Demo

- Solutions
- Agents
- RAG
- Generation
- Summarization
- Testing
- Monitoring

- Company
- About Us
- Careers
- Contact Us

- Resources
- Docs
- Blog
- Checks Demo 🔗
- LLM Tools
- Glossary
- Events
- FAQs

- Privacy Policy
- Cookies Policy
- Terms & Conditions

FREE TRIAL FOR DEEPCHECKS' LLM EVALUATION SOLUTION

Fill Out Your Details Here

×

Full Name*

Work Email*

A few words about your use case and areas of interest

Submit

Deepchecks is Now Available Natively Within AWS Sagemaker

Want to learn more?

×

Full Name*

Work Email*

A few words about your use case and areas of interest

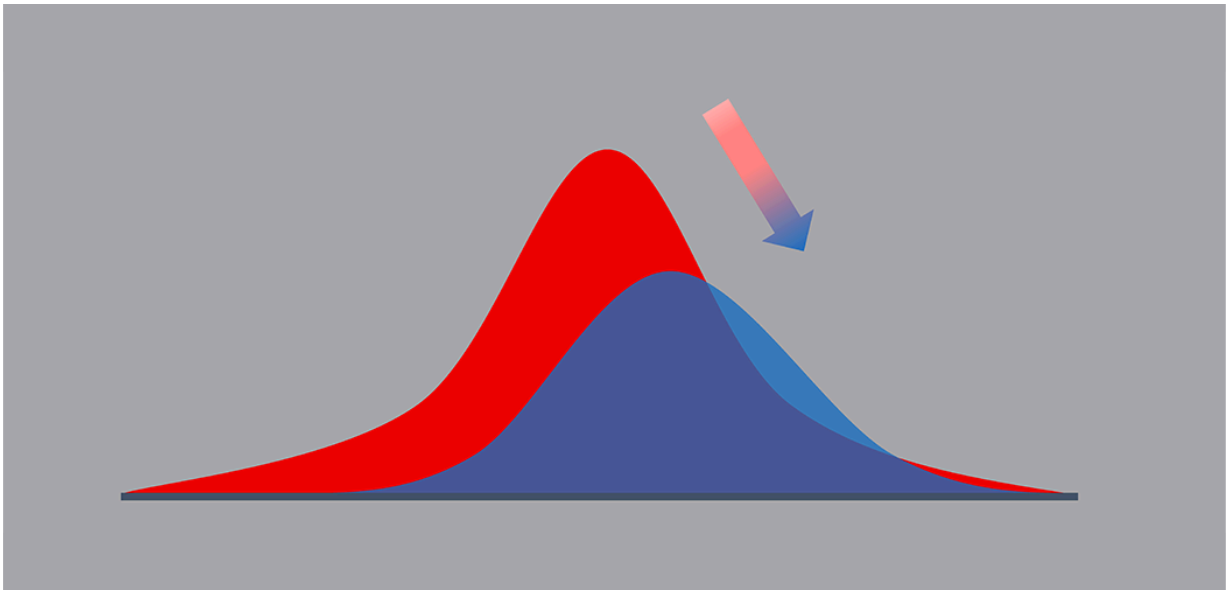
Submit

ML MONITORING

Machine Learning Monitoring, Part 5: Why You Should Care About Data and Concept Drift

Last updated: January 9, 2025

Published: November 12, 2020



This blog is a part of the Machine Learning Monitoring series. Earlier we:

- *introduced the concept of model performance monitoring;*
- *discussed who should care about it and the gap that exists;*
- *looked at the possible data quality and integrity issues, and*
- *proposed how to track data quality.*

Data quality issues account for a major share of failures in production.

But let's say it is covered. The data engineering team does a great job, data owners and producers do no harm, and no system breaks. Does this mean our model is safe?

Sadly, this is never a given. While the data can be right, the model itself can start degrading.

A few terms are used in this context. Let's dive in.

Model decay

"Past performance is no guarantee of future results." It's a small print for most financial products out there.

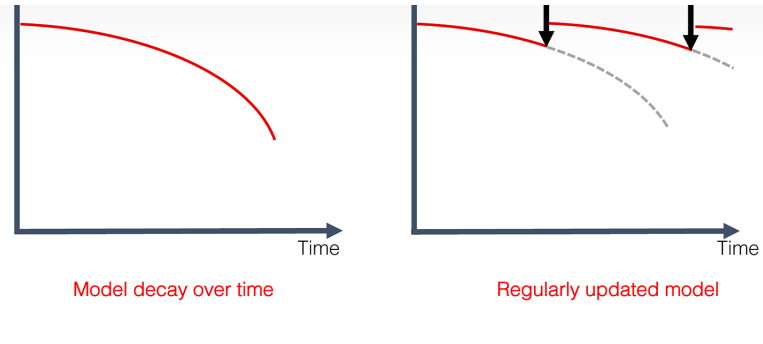
People call it **model drift**, **decay**, or **staleness**.

Things change, and model performance degrades over time. The ultimate measure is the model quality metric. It can be accuracy, mean error rate, or some downstream business KPI, such as click-through rate.

No model lives forever, but the speed of decay varies.

Some models can last years without an update. For example, certain computer vision or language models. Or any decision system in an isolated, stable environment.

Others might need daily retraining on the fresh data.



Model decay is a term that literally says that the model got worse. But of course, it happens for a reason.

When data quality is fine, there are two usual suspects: data drift or concept drift. Or both at the same time.

Bear with us. We'll explain it now.

Data drift

Data drift, **feature drift**, **population**, or **covariate shift**. Quite a few names to describe essentially the same thing.

Which is: the input data has changed. The distribution of the variables is meaningfully different. As a result, the trained model is not relevant for this new data.

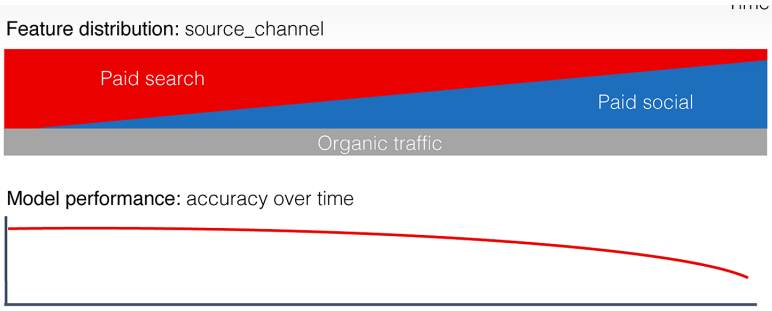
It would still perform well on the data that is similar to the "old" one! The model is fine, as much as the model "in a vacuum" can be. But in practical terms, it became dramatically less useful since we are dealing with a new feature space.

Here is an example of propensity modeling.

An online marketplace attracts users through advertising. Soon after newcomers sign up, we want to predict how likely they will make a purchase in order to send them personalized offers.

Previously most user acquisition happened through paid search. After a new ad campaign, a lot of users come from Facebook. In training, our machine learning model did not do well in this segment. It had limited examples to learn from.

Still, the overall model quality was sufficient since this sub-population was small. Now that it grew, the performance predictably dropped.

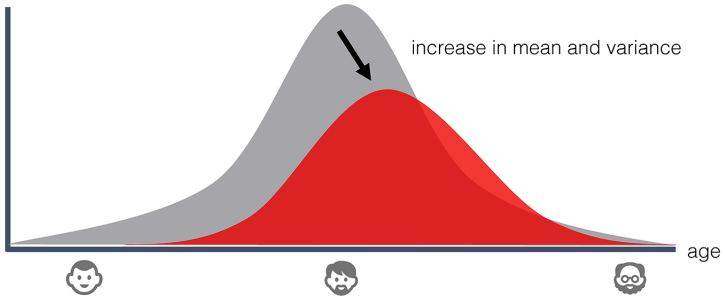


As more users come from social channels, the model performance declines.

When debugging, we would see the change in class frequencies in the "source_channel" and distributions of other related variables, such as "device," "region," and so on.

Or, with the monitoring in place, we'd notice this drift early on!

Similar things happen when applying a model in a new geographical area. Or, if the growing audience starts to include a new demographic segment, and so on.



Change in the distribution of the "age" feature.

In these examples, the real-world patterns can remain the same. But the model performance drops as its quality varies for different data regions.

The decay can range from minor to dramatic—when the drift affects the most important features.

To address it, we need to train the model on the new data or rebuild it for the new segment.

Training-serving skew

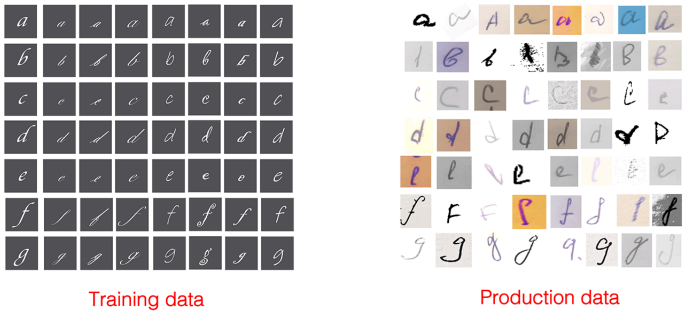
This one deserves an honorable mention.

It is often mixed with data drift or used interchangeably. The way we

production use of the model. Training-serving skew is more of a mismatch. It reveals at the first attempt to apply the model to the real data.

It often happens when you train a model on an artificially constructed or cleaned dataset. This data does not necessarily represent the real world, or does this incompletely.

For example, an invoice classification model is trained on a limited set of crowdsourced images. It does well on the test set. But once put in production, it is surprised by the diversity of how people fill in the invoice data. Or the low quality of scanned images compared to model training.



Training-serving skew.

Recently, the Google Health team faced a similar issue. They developed a computer vision model to detect signs of retinopathy from eye scan images. In practice, those were often taken in poor lighting conditions. The model struggled to perform as well in real life as it did in the lab.

In most cases of training-serving skew, the model development has to continue. If you are lucky, the non-successful trial run might instead generate enough data to train a new model or adapt the existing one. Otherwise, you might need first to collect and label a new dataset.

Concept drift

Concept drift occurs when the patterns the model learned no longer hold.

In contrast to the data drift, the distributions (such as user demographics, frequency of words, etc.) might even remain the same. Instead, the relationships between the model inputs and outputs change.

In essence, the very meaning of what we are trying to predict evolves. Depending on the scale, this will make the model less accurate or even obsolete.

Concept drift comes in different flavors.

Gradual or **incremental** drift is the one we expect.

The world changes and the model grows old. Its quality decline follows the gradual changes in external factors.

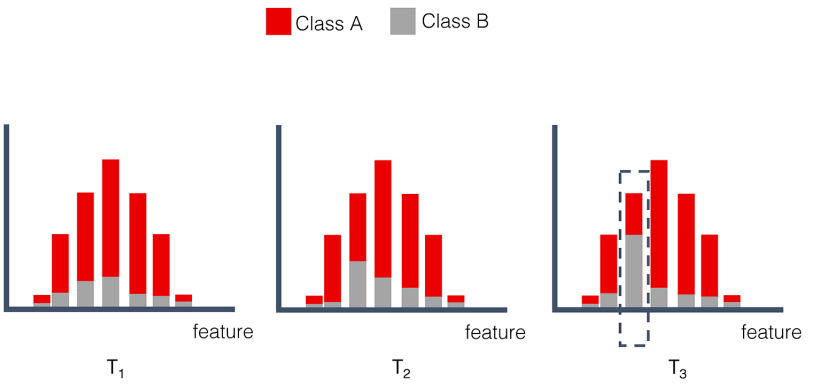
Some examples:

- **Competitors launch new products.** Consumers have more choices, and their behavior changes. As should sales forecasting models.
- **Macroeconomic conditions evolve.** As some borrowers default on their loans, the credit risk is redefined. Scoring models need to learn it.
- **Mechanical wear of equipment.** Under the same process parameters, the patterns are now slightly different. It affects quality prediction models in manufacturing.

No individual change is dramatic. Each might affect only a tiny segment. But eventually, they add up.

Sometimes it is possible to observe the shift at a level of individual features.

This is how it might look in a binary classification task, like churn prediction. The distribution of a specific feature is stable. But the share of the target class at a particular value range grows over time. A new predictive pattern appears. But this is just a single feature: the initial effect on the model performance is mild.



A change in the relationship between a given feature and the prediction target.

How fast do models age?

It depends, of course. But it is often possible to get an estimate ahead of time.

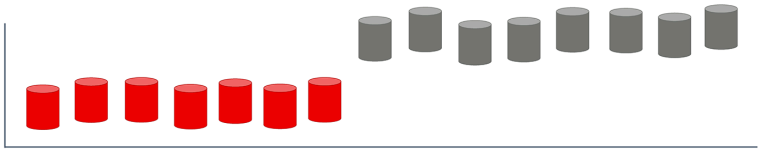
it impacts model quality.

Such a test gives us a good proxy on regular model retraining needs. Does each new week's data improve the performance? Or a 3-month old model still performs as good as new?

We can then schedule model updates on freshly accumulated data at a chosen interval.

We describe more checks one can run to decide on the retraining schedule in a separate blog.

Sudden concept drift

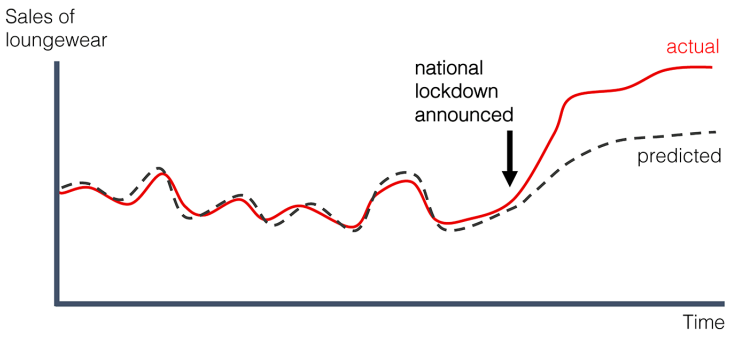


External changes might be more **sudden** or **drastic**. These are hard to miss. As we shared in our recent post, the COVID-19 pandemic is a perfect example.

Almost overnight, the mobility and shopping patterns shifted. It affected all sorts of models, even otherwise "stable" ones.

Demand forecasting models would not guess that sales of yoga pants surge 350% (like it happened to Stitch Fix) or that most flights will be canceled as borders close.

Were you working to recognize pneumonia on X-ray images? You instantly got a new label.



Consumer demand changes suddenly due to stay-at-home policy.

- **Change in the interest rate by the central bank.** All financial and investment behavior is affected, and models fail to adapt to unseen patterns.
- **Technical revamp of the production line.** Predictive maintenance becomes obsolete since modified equipment has new failure modes (or lack of those).
- **Major update in the app interface.** Past data on clicks and conversion becomes irrelevant since the user journey is a new one.

Dealing with drift

Image credit: Unsplash.

In the case of radical shifts, models break. In the meantime, the not-so-relevant model might need human help. You might want to pause it until more data is collected. As a fallback strategy, you can use expert rules or heuristics.

To get back on track, we need to retrain the model. Approaches vary:

- Retrain the model using all available data, both before and after the change.
- Use everything, but assign higher weights to the new data so that model gives priority to the recent patterns.
- If enough new data is collected, we can simply drop the past.

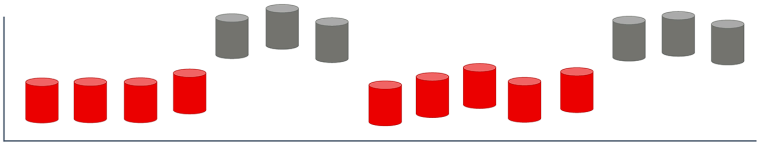
"Naive" retraining is not always enough. If the problem has evolved, we might need to tune the model, not feed the latest data into the existing one.

Options include domain adaptation strategies, building a composition of models that use both old and new data, adding new data sources, or trying entirely new architectures.

Sometimes it's best to modify the model scope or the business process. It can be shortening the prediction horizon or running the model more frequently. For example, retailers facing shifts in shopping habits might

model maintainers. An early warning will help you prepare the model for a new cold start.

Recurring concept drift



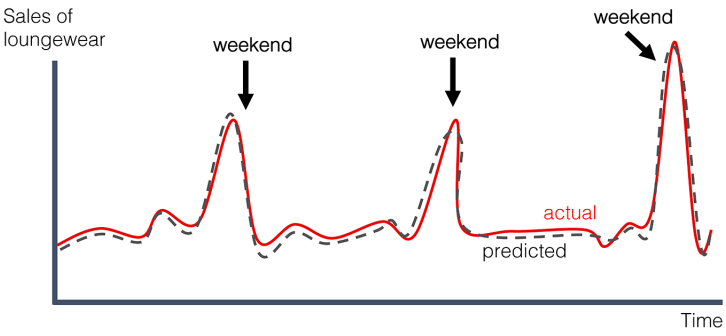
One more honorable mention.

Some researchers use the term "**recurring** drift" to describe repeating changes. We feel here things get mixed up a bit.

Seasonality is a known modeling concept. It indeed looks like (and it is) a temporary change in the target function.

The same people that shop modestly during the year show unusual patterns on Black Fridays. Bank holidays affect everything, from retail sales to production failures. Mobility on the weekend is different from business days. And so on.

If we have a sound model, it should react to these patterns. Every Black Friday resembles the one a year before. We can factor cyclic changes and special events in the system design or build ensemble models. "Recurring drift" will be expected and will not lead to quality decline.



Consumers shop more on the weekends. The pattern is known, and the model forecast shows it.

From the point of model monitoring, this "drift" has no importance. Weekends happen every week, and we don't need an alert. Unless we see a new pattern, of course.

If it first sees some special events or seasons in production, you can use other similar events as examples. For instance, if a new bank holiday is introduced, you can assume a similarity with a known one.

If needed, domain experts can help to add manual post-processing rules or corrective coefficients on top of the model output.

Drift recap

Let's sum up.

All models degrade. Sometimes, the performance drop is due to low data quality, broken pipelines, or technical bugs.

If not, you have two more bets:

1. **Data drift: change in data distributions.** The model performs worse on unknown data regions.
2. **Concept drift: change in relationships.** The world has changed, and the model needs an update. It can be gradual (expected), sudden (you get it), and recurring (seasonal).

In practice, the semantic distinction makes little difference. More often than not, the drift will be combined and subtle.

What matters is how it impacts the model performance, if retraining is justified, and how to catch this on time.

How to analyze this? Stay tuned for the next blog, and check out our open-source tools for model analytics on GitHub.

WRITTEN BY

Emeli Dral
Co-founder and CTO
Evidently AI

Elena Samuylova
Co-founder and CEO
Evidently AI

SHARE ON

#data-drift #concept-drift

You might also like

TUTORIALS

How to break a model in 20 days. A tutorial on production model analytics

What can go wrong with ML model in production? Here is a story of how we trained a model, simulated deployment, and analyzed its gradual decay.

ML MONITORING

"My data drifted. What's next?" How to handle ML model drift in production.

What can you do once you detect data drift for a production ML model? Here is an introductory overview of the possible steps.

⌨️

05. Data Drift in Text data

날짜	@2025년 2월 3일 → 2025년 2월 7일
태그	docs
상태	시작 전

> 왜 텍스트 데이터에서 드리프트 탐지를 위해 임베딩을 적용하여 사용해야 하는가?

텍스트 데이터는 이미지, 오디오 등과 함께 비정형 데이터(Unstructured Data)에 속하며, 전통적인 방식으로는 직접적인 분포 비교가 어렵다. 따라서, NLP(자연어 처리) 모델에서는 **임베딩(Embedding)** 기법을 활용하여 텍스트 데이터를 고차원의 벡터 공간으로 변환한 후, 데이터 드리프트를 탐지하는 것이 일반적이다.

본 문서에서는 임베딩을 활용한 텍스트 데이터 드리프트 탐지 방법과 각 접근 방식의 특성을 비교한다.

1. 임베딩 벡터 변환

1.1 비정형 데이터 특성

- 텍스트 데이터는 단순한 수치형 데이터와 달리 직접적인 비교가 어려움
- 즉, "사과"와 "바나나"라는 단어는 개별 문자로 비교하면 유사성이 전혀 없지만, 의미적으로는 둘 다 과일이라는 공통점이 있음
- 이러한 의미 기반의 비교를 위해 임베딩을 활용

1.2 임베딩 벡터 변환

- 입력 데이터가 저차원의 수작업으로 설계된 벡터 공간임을 가정하기에 NLP 어플리케이션에 적용하기에는 어려움
- NLP. 고차원의 latent document embeddings을 사용하는 복잡한 언어 모델을 사용하기 때문

임베딩을 활용하면 텍스트 데이터를 다차원 공간의 벡터로 변환할 수 있음

대표적인 방법으로는 다음과 같은 사전 학습된 언어 모델:

- **BERT (Bidirectional Encoder Representations from Transformers)**
- **FastText**
- **Word2Vec**
- **GloVe**

이러한 모델을 활용하면 텍스트를 고차원 벡터로 변환할 수 있으며, 이 벡터를 기반으로 데이터 드리프트를 탐지

1.3 텍스트 데이터에서 발생할 수 있는 드리프트 예시

- **새로운 인기 단어나 밈 등장**
  - ex. "챗GPT" 같은 신조어 증가
- **새로운 클래스 등장 또는 클래스 비율 변화**
  - ex. 제품 리뷰에서 "친환경" 관련 키워드 급증
- **스팸성 콘텐츠 증가**
  - ex. 광고성 문구 증가
- **감정 변화 (긍정 → 부정)**
  - ex. 특정 브랜드에 대한 소비자 인식 변화
- **새로운 언어 등장**
  - ex. 기존 영어 리뷰에 스페인어 리뷰 증가

2. 텍스트 데이터 드리프트 탐지 방법 비교

드리프트 탐지 방법 비교는 다음과 같다.

- **유클리드 거리 (Euclidean Distance)**
  - 두 데이터셋의 평균 임베딩 간 거리를 계산
  - 값이 클수록 두 데이터셋 간 차이가 크다는 의미
- **코사인 거리 (Cosine Distance)**
  - 두 벡터 간 각도를 계산
  - 동일한 벡터는 0, 완전히 다른 벡터는 2
- **모델 기반 탐지 (Classifier)**
  - 바이너리 분류기를 학습해 참조와 현재 데이터를 구별
  - ROC AUC(0.5~1)로 드리프트 여부 판단
- **임베딩 구성 요소 비율 (Share of Drifted Components)**
  - 각 임베딩의 수치 구성 요소 간 변화 비율 측정
  - 변화된 구성 요소 비율이 일정 임계값을 넘으면 드리프트로 판단
- **최대 평균 차이 (Maximum Mean Discrepancy, MMD)**
  - 두 분포 간 평균 차이 측정
  - 값이 0에 가까우면 동일, 값이 클수록 차이가 큼

References

[1] https://www.evidentlyai.com/blog/embedding-drift-detection

[2] Drift Detection in Text Data with Document Embeddings