기계학습에 사용되는 테스트 케이스 생성에 관한 고찰

정재호, 노종선
서울대학교 전기정보공학부 뉴미디어통신공동연구소
jaehoj@ccl.snu.ac.kr, jsno@snu.ac.kr

A Study on Test Case Generation for Machine Learning Systems

Jaeho Jeong and Jong-Seon No
INMC, Department of ECE, Seoul National University

요 약

본 논문은 기계학습에 관한 연구 중에서 데이터를 학습, 검증, 시험하기 위해 필요한 테스트 케이스를 잘 설계하고 적용하는 방법에 대한 최신 연구들을 살펴보았다. 현재 진행되고 있는 관련 연구 분야로는 테스트 케이스 생성, 테스트 케이스 감축, 테스트 케이스 우선 순위화 등이 있으며 최신 연구들이 컴퓨터 공학 분야에서 어떤 방법으로 테스트 케이스 생성과 관련된 성능을 발전시켰는지 알아본다.

Ⅰ. 서 론

일반적으로 기계학습(Machine Learning)에 관한 연구라고 하면 기계학습 그 자체의 성능을 높이기 위한 알고리즘 혹은 기계학습을 적용시킬 수 있는 새로운 분야에 관한 연구가 진행되기 마련이다. 하지만 이러한 연구를 진행하기 위해 데이터를 학습(training), 검증(validation), 시험(test)하는 데는 필요한 테스트 케이스(Test Case)를 적절하게 잘 만들고 적용하는 것이 매우 중요하다. 일반적으로는 기계학습에 MNIST, CIFAR-10, CIFAR-100, ImageNet 등 이미지 파일들로 구성된 데이터 세트를 많이 사용하지만, 다른 분야에 적용되는 기계학습의 경우에는 학습을 위한 데이터를 어떻게 설계하느냐에 따라 학습이 잘 안 될 수도 있고, 데이터 세트 전처리를 어떻게 하느냐에 따라 성능 차이가 크게 벌어질 수 있다. 그렇기 때문에 테스트 케이스 그 자체에 대한 연구도 활발히 진행되어야 하는데, 현재 진행되고 있는 관련 연구 분야로는 테스트 케이스 생성(Test Case Generation), 테스트 케이스 감축(Test Case Reduction), 테스트 케이스 우선 순위화(Test Case Prioritization) 등이 있다. 특히 이러한 기계학습용 테스트 케이스를 잘 선별해 내는 데에도 기계학습 그 자체가 다시 사용될 수 있다.

본 논문에서는 테스트 케이스 생성 기법이 적용될 수 있는 컴퓨터 공학 분야들에 대해 먼저 알아본다. 이후 이와 관련된 최신 연구들에서 각각의 분야들에 어떤 방법으로 테스트 케이스 생성과 관련된 성능을 발전시켰는지 분석한다.

Ⅱ. 본 론

총 3종류의 최신 연구를 바탕으로 각각의 연구가 진행된 분야에 대한 소개와 함께 이를 통해 테스트 케이스 생성, 감축, 우선 순위화 방식을 어떻게 적용시켰는지 자세히 알아보도록 한다.

- 소프트웨어 테스트(Software Testing)

소프트웨어 테스트란, 소프트웨어가 제대로 작동하는지, 버그가 발생할 수 있다면 어떤 부분에서 발생할 수 있는지 검증하는 작업이다. 개발이 완료된 소프트웨어를 출시하기 전에 이러한 테스트를 미리 많이 해봐야 소프트웨어의 보안이나 안정성이 보장되어 공격을 당해 악용되거나 제대로 작동을 못 하는 것을 방지할 수 있다. 이러한 소프트웨어 테스트를 진행할 때, 개발자들이 입력값을 직접 만들어서 테스트를 진행하게 된다면 속도도 느릴뿐더러 미처 생각해내지 못한 부분에 대한 테스트는 진행하지 못할 수도 있다. 따라서, 테스트 케이스 생성을 자동화시키는 방법에 대한 연구를 연구[1]에서 진행하였고, 그 과정에서 Focal Method라는 방식이 적용되었다. Focal Method란 그림 1에서 보이는 것처럼 전체 소스 코드들을 클래스(class), 메소드(method), 생성자(constructor) 등 여러 context로 적당히 쪼개는 방법을 의미한다.

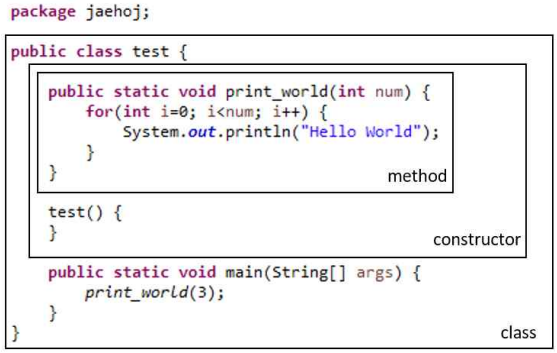

그림 1. Focal Method를 통해 자바 소스 코드를 구분한 예시

연구[1]에서는 오픈소스(open-source) 자바(Java) 소스 코드들을 긁어모아 코드 자체를 영어 문장들로 보거나 focal method를 이용하여 적당히 쪼개어 구분해 놓았다. 이후 이를 자연어 처리 기계학습 모델의 일종인 sequence-to-sequence transformer에 적용하였고, 이렇게 학습시켜 자동

으로 생성된 소스 코드들을 살펴본 결과, 기존 개발자들이 작성한 테스트 케이스들과 매우 유사하면서도 이해하기 쉽고 정확한 데이터 세트들을 작성할 수 있었고, 이를 온라인으로 공개하였다.

- CI (Continuous Integration)

CI란 동시에 여러 개발자들이 소스 코드를 수정하다 보면 발생할 수 있는 프로그램 버전 혹은 호환성 등과 같은 문제를 해결하기 위한 연구 분야이다. 전체 커다란 프로그램이 있을 때 일부 작은 부분의 코드만 수정이 되었다고 전체 프로그램을 다시 돌려보기에는 자원 낭비가 심할 수 있고, 코드가 일부 수정되어 샘플 입력을 돌려보는 과정에서 다른 한쪽의 코드가 수정된 것은 반영이 안 되어 있을 수도 있는 등, 여러 개발자들이 참여하는 프로젝트에서는 위와 같은 CI 이슈가 발생할 수 있다.

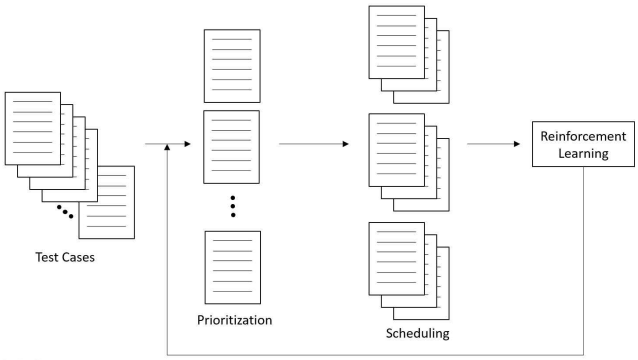

그림2. 테스트 케이스 우선 순위화 과정과 스케쥴링 과정을 학습하는 과정

연구[2]에서는 이러한 CI 과정에서 테스트 케이스를 자동으로 선택하고 우선 순위화하는 데 있어서 최초로 강화학습을 적용시켰고, 현재까지 나와있는 다른 테스트 케이스 자동생성 프로그램들과 비교했을 때 더 적은 cycle 수로 모든 테스트 케이스 검사를 진행할 수 있도록 우선 순위화하여 스케쥴링(scheduling) 방식으로 동시에 순차적으로 프로그램을 검증시키는 데에 성공하였다(그림 2). 이 과정에서 학습에 필요한 문제를 state, action, agent, policy, reward function 등의 notation으로 구분하여 실제 현업에서 사용되는 데이터도 강화학습에 적용될 수 있다는 점을 보일 수 있었다.

- 기호 실행(Symbolic Execution)

기호 실행이란 어떤 프로그램의 입력값으로 구체적인 값(concrete value)이 아닌 정해지지 않은 기호 값(symbolic value)이 들어가는 것을 의미한다. 변수에 들어갈 수 있는 모든 값이 입력으로 가능하기에 특정 분기 조건(branch condition)에서 어떤 방향으로 진행하는지 볼 수 있으며, 프로그램의 일정 부분으로 가기 위해서는 어떤 입력값을 넣어야 하는지 알 수 있다. 프로그램 진행 도중 분기 조건을 마주쳤을 때 포크(fork)를 통해 양쪽 경로를 모두 다 탐험할 수 있어 프로그램 내 모든 경로를 훑을 수 있기에 소프트웨어 테스트나 소프트웨어 보안 분야에 많이 사용된다.

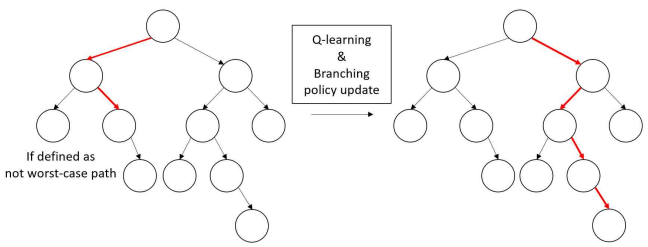

그림 3. 기호 실행 도중 다음 분기로 넘어가는 과정

연구[3]에서는 기호 실행을 통해 최악의 경로(worst-case path)값을 찾는 데 활용하였다. 그림 3에서와같이 특정 분기 정책(branching policy)을 통해 경로를 찾다가 Q-learning을 통해 최악의 경로값이 아니라고 판단이 되는 패턴이 등장한다면 해당 경로 탐험을 중단하고 이전 분기의 다른 경로로 탐험을 진행하게 된다. 이 과정에서 해당 분기 정책을 업데이트하고 이를 다시 학습하면서 기호 실행을 지속하게 된다. 일반적으로 분기 조건마다 탐험해야 하는 경로는 최소 2배씩 늘어나므로 복잡한 프로그램을 기호 실행하기 위해서는 경로가 기하급수적으로 증가하는 경로 폭발(path explosion) 문제가 발생하기에 작은 규모의 프로그램에서만 기호 실행을 진행할 수 있다는 단점이 존재한다. 그러나 연구[3]과 같은 경우에는 경로 폭발 문제가 발생하기 전에 특정 분기 아래의 모든 경로를 제거할 수 있으므로 더 큰 규모의 프로그램에 대해서도 최악의 경로를 제대로 탐색할 수 있었다.

Ⅲ. 결 론

본 논문에서는 테스트 케이스 생성 기법에 관한 최신 연구들을 살펴보면서 이들이 각기 다른 컴퓨터 공학의 분야에 어떤 식으로 적용될 수 있었는지 알아보았다. 주로 소프트웨어 테스트 분야에 사용된다는 점을 알 수 있었으며, 이들 연구에서의 주된 기여(contribution)는 테스트 케이스를 자동으로 만드는 데 성공하였고 비교 대상인 다른 소프트웨어들보다 이해하기 쉽고 성능이 좋은 테스트 케이스를 만들었으며 방대한 양의 고품질 데이터 세트 및 코드들을 만든 후 온라인으로 공개한 점들이었다. 비록 이번 논문에서는 테스트 케이스 생성, 감축, 우선 순위화에 관한 연구들을 모두 다 다루지는 못하였지만, 소프트웨어 엔지니어링 분야에서 관련 연구가 매우 많이 진행되고 있기에 이 분야들에 조금 더 관심 있게 본다면 기계학습 연구에 있어서 큰 도움이 될 수 있을 것이다.

ACKNOWLEDGMENT

이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2021-0-00400, 저사양 디바이스 대상 고효율 PQC 안전성 및 성능 검증 기술 개발)

참 고 문 헌

[1] Tufano, Michele, et al. "Unit Test Case Generation with Transformers." *arXiv preprint arXiv:2009.05617* (2020).

[2] Spieker, Helge, et al. "Reinforcement learning for automatic test case prioritization and selection in continuous integration." *Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 2017.

[3] Koo, Jinkyu, et al. "Pyse: Automatic worst-case test generation by reinforcement learning." *2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST)*. IEEE, 2019.