1장

인공지능이 사람처럼 행동하고 사고할 것이 가능한지는 잠수함이 수영할 수 있는지 묻는 것과 같다.

이 책에서 다루는 알고리즘 분류는 기존에 사람만 할 수 있는 일(이미지를 설명하는 문장을 출력하거나 대본을 낭독하는 일)을 인공지능이 자동화해주는 것뿐이다.

위 과정은 딥러닝에 해당하고, 심층 신경망을 통해 대량의 데이터를 사용해 입력과 출력이 동떨어진 복잡한 함수들을 근사하는 방법을 사용한다.

- 입력 데이터와 결과에 대하여 복잡한 관계를 수치화한다.

1.1 딥러닝 혁명

초기 머신러닝은 피처(feature) 엔지니어링에 크게 의존하였다.

피처 엔지니어링

- 분류하기 위해서 입력 데이터를 크게 특성으로 분류하여 변환하는 작업
- 이 과정을 통해 필터 집합(숫자를 분류하기 위해서는 동그라미가 들어있는지 안들어있는지, 직선이 몇 개인지 등의 필터)이 생겨나고, 이를 근거로 데이터를 변환

딥러닝

- 원본 데이터로부터 자동으로 표현들을 찾아낸다.
- 피처 엔지니어링에서 수작업으로 찾았던 필터가 데이터와 대응 레이블(타겟)을 반복적으로 훈련시킨 결과 자동으로 개선된다..
- 기존 피처 엔지니어링보다 빠르고, 성능도 좋아졌다.

허나 피처 엔지니어링이 아예 사라진 것은 아니므로, 학습 전에 사전 지식을 주입할 필요가 있다.(초기 가중치 설정)

훈련

- 기대한 값과(타겟 데이터) 실제 모델 출력의 차이(손실 함수)를 통해 점수를 제공한다. 더 낮은 점수(손실)을 얻도록 점진적으로 딥러닝 머신을 변경(가중치 연산)하여 입력되지 않은 데이터에 대해서도 낮은 점수 달성하게 한다.

1.2 딥러닝을 위한 파이토치

파이토치에서는 텐서라는 데이터 구조를 제공하며, 이는 numpy 배열과 여러 면에서 유사하다.
더 복잡하고 새로운 기술을 습득하기 전 파이토치를 통해 기본적인 딥러닝 도구를 훈련해보면 좋을 것 같다.

1.3 왜 파이토치인가?

실제 학습을 위해서는 다양한 문제에 적응할 수 있을 정도로 유연하면서도, 적당한 시간 안에 많은 양의 데이터로 훈련 가능한 효율적인 도구가 필요하다.

- 파라미터의 개수 제한에 관련된 내용

이 도구로 훈련된 모델은 다양한 입력에 대해 정확하게 작동해야 한다.

단순함

- 파이토치는 학습, 활용, 확장, 디버깅 면에서 수월하다. 파이토치는 파이썬스러운 면이 있어서 친숙하다.
- numpy에서 배열, 벡터, 행렬을 텐서로 표현함으로써 넘파이를 안다면 금방 친숙해진다.

특별한 기능

1. GPU로 연산이 가능하여 CPU보다 50배 빠른 결과를 보여주기도 한다.
2. 딥러닝이 사용하는 일반 수학식에 대해 산술 최적화를 지원한다.(ex: Dynamic typing을 지원하지 않고 실수 혹은 정수로만 이루어져 있어서 연산이 빠르다)

파이토치는 라이브러리다(프레임워크가 아니다!)

라이브러리와 프레임워크의 차이

라이브러리와 프레임워크의 차이는 제어의 흐름의 권한이 어디있냐의 차이이다. 라이브러리를 사용할 때에는 애플리케이션 코드의 흐름을 직접 제어해야 하고, 개발 시 필요한 기능이 있을 경우 라이브러리에서 직접 호출한다.

프레임워크는 코드가 프레임워크에 의해 사용된다. 코드는 프레임워크가 짜놓은 틀에 수동적으로 동작하기 때문에 제어의 흐름은 프레임워크에 있다.(Google 검색)

1.4 파이토치 딥러닝 프로젝트 둘러보기

파이토치는 기본적으로 다차원 배열(텐서) 자료구조를 사용하고, 텐서 연산을 torch 모듈로 제공한다. 단 몇 줄로 CPU에서의 연산을 GPU로 옮길 수 있다.

특정 텐서에서 수행한 모든 연산을 기억했다가 주어진 입력값을 기준으로 미분값을 내부적으로 자동 계산해준다.

Torch.nn에서 신경망 구축을 지원하고, 완전 연결 계층, 컨볼루션층, 활성화 함수, 손실 함수가 여기에 포함되어 있다. 훈련을 위한 데이터를 가져오는 과정에서 torch.utils.data의 Dataset 클래스가 필요하다. 데이터 변환은 데이터형에 따라 직접 구현해야 한다.

접근 대기시간을 줄이기 위해 데이터를 병렬로 로딩해야 하고, 이 과정에서 배치 형태로 묶기 위해 DataLoader 클래스에 담는다. 이 배치 샘플 묶음은 파이썬의 for문을 통해 훈련 루프에 들어가게 된다.

앞에서 설명한 점수를 계산하기 위해 손실함수/기준을 사용하게 되는데 이는 torch.nn에 있다. 손실 함수을 근거로 모델을 조정하는 옵티마이저는 torch.optim에서 제공한다.

이 과정이 끝나면 훈련된 모델이 만들어지고, 사용되기 위해 배포 과정을 거쳐야 한다. 배포 프로세스는 모델을 서버에 올리거나 클라우드 엔진에 로딩할 수 있도록 내보내는 과정이다.

1.5.1주피터 노트북

커널: 코드를 실행한 후 결과를 반환하는 서버에서 동작하는 프로세스.

셀: 주피터 노트북과 상호작용하는 기본 단위

1.6 연습 문제

1.

a) 3.9.7 version 사용중이다

b) torch.__version__

'1.12.0+cpu' – 1.12.0버전을 사용중이며 cpu를 사용중이다. GPU 사용시 cu###이 출력될 것이다.

c) torch.cuda.is_available()

False

Cuda(GPU)를 사용하지 않으므로 False가 알맞게 출력된다.

2.

a) import sys → sys.version → 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]

b) ??