맨 처음 1장에서는 인공지능과 머신러닝, 딥러닝에 대해 소개해주고 있습니다.

우선 인공지능은 영화에서나 등장하는, 사람과 구분하기 어려운 지능을 가진 강인공지능이 존재하고,
제한된 영역에서 사람의 일을 도와주는 보조 역할을 하는 약인공지능이 존재합니다.

머신러닝이란
규칙을 일일이 프로그래밍 하지 않아도 자동으로 데이터 규칙을 학습하는 알고리즘을 연구하는 분야라고 합니다. 최근의 머신러닝 발전은 통계나 수학 이론보다 경험을 바탕으로 발전하는 경우가 많다고 하는데 이 말이 무슨 뜻인지는 잘 모르겠습니다.

중요한 것은 파이썬의 사이킷런, 구글의 텐서플로우, 페이스북의 파이토치 등 머신러닝 관련 오픈소스 라이브러리가 굉장히 많다는 것입니다. 때문에 머신러닝에 필요한 수학적 백그라운드, 알고리즘을 구현하기 위한 프로그래밍 실력 진입장벽이 많이 낮아졌습니다. 근데 또 월급은 많이 괜찮은 것 같습니다.

딥러닝이란
머신러닝의 알고리즘 중 인공신경망을 기반으로 한 방법입니다.
딥러닝 기법을 이용해 1998년 손 글씨, 숫자 인식에 성공하였고, 2012년에는 이미지 분류에 성공하였습니다. 그리고 2016년 알파고가 등장하여 세상을 놀라게 했습니다.
딥러닝은 많은 머신러닝 알고리즘의 한 부분이지만, 사람들에게 인공지능의 힘을 알려주고 인공지능 붐을 일으키는데 큰 역할을 했기에 유명해진 것 같습니다.

K - nearest neighbor (k 근접 이웃) 모델은 책 앞부분에 자주 등장하기 때문에 따로 정리를 해보았습니다.
훈련의 목적이 종류의 분류이냐, 값의 예측이냐 에 따라 분류와 회귀 로 나뉩니다.

우선 분류의 경우 데이터가 들어왔을 때 가장 가까운 k개의 데이터를 찾은 후 높은 비율을 차지하는 클래스를 반환합니다. 그림을 보면 k가 3인 경우인데 2개의 초록색, 1개의 주황색을 포함하므로 초록색을 반환합니다.

회귀의 경우 가격을 예측하는 것이 목적이라면, 근처 k개의 평균 가격을 반환합니다.

하지만 가장 가까운 k개의 데이터라 하더라도 각각의 거리가 다른 것은 사실입니다.
이를 보완하기 위해 거리의 반비례하게 가중치를 적용하는 weighted - knn 모델도 존재합니다.
그림을 보면 거리를 고려하지 않았을 때 35를 반환하고, 가중치를 고려했을 때 조금 더 합리적인 45.4를 반환하는 것을 알 수 있습니다.

다음은 선형모델과 knn을 비교한 그림입니다.

Knn의 경우 선형이라는 모델에 구애받지 않기 때문에 노이즈의 영향을 크게 받지 않습니다.
또한 학습 데이터의 수가 많은 경우 굉장히 정밀한 모델을 만들 수 있습니다.

하지만 새로운 데이터가 들어왔을 때 모든 데이터와의 거리를 구한 후 정렬하는 방식이므로 지속적인 업데이트가 필요한 경우 비효율적입니다. 또한 이웃의 갯수 k를 최적화 하는 과정을 거쳐야 합니다. 동떨어진 데이터 아웃라이어의 경우 거리가 먼 값들의 평균을 예측하기에 아웃라이어에 취약합니다.

최적의 k값을 구하는 방법은 chapter 3장 규제 파트에서 등장합니다.

K 가 너무 낮으면 데이터의 지역적인 특성을 지나치게 반영합니다. 이를 over - fitting (과대적합)되었다고 표현합니다. 반면 k 가 너무 높으면 다른 개체의 범주를 너무 많이 포함하게 됩니다. 이는 under - fitting 되었다고 합니다. 따라서 훈련데이터의 점수가 높으며, 테스트데이터의 점수와 크게 차이가 나지 않는 k값을 구해주어야 합니다. 이는 다음 단원에서 배우게 됩니다.

Chapter 2는 머신러닝의 분류를 설명하는 것에서 출발합니다. 머신러닝은 크게 지도 학습, 비지도 학습, 강화 학습으로 나눌 수 있습니다.

지도 학습의 경우 학습데이터를 통해 컴퓨터를 훈련하는 방식입니다. 강아지와 고양이를 분류하거나, 부동산의 가격을 예측하는 것이 이에 해당합니다.

비지도 학습의 경우 학습데이터 없이 오직 입력 데이터만으로 컴퓨터를 훈련하는 방식입니다. 비지도 학습의 경우 예측이 목적이 아닌 데이터의 구성, 특징을 분류하는데 사용됩니다. 따라서 학습데이터가 필요 없고, 뉴스 기사를 분류하거나 특정 집단을 나누는 것이 이에 해당합니다.

강화 학습의 경우 입력데이터라는 것이 존재하지 않습니다. 주어진 환경에서 선택가능한 행동들 중 보상을 최대화 하는 행동 또는 행동 순서를 학습하는 방식입니다. 대표적으로 알파고와 같이 게임 ai들이 이에 해당합니다.

우리가 배울 k 근접 이웃 모델은 지도학습에 속합니다. 앞서 설명했듯이 지도 학습에는 데이터(input) 와 정답(target) 으로 이루어진 훈련 데이터가 필요합니다.

하나의 데이터는 데이터를 구성하는 특성(feature)들로 나눌 수 있습니다. 예제에 나온 길이나 무게가 특성에 해당합니다. 타겟의 경우 분류이냐, 회귀이냐에 따라 다른데 예제의 경우 도미와 빙어를 분류하는 것이므로 도미(1), 빙어(0)와 같이 표현합니다.

참고로 사이킷런에 사용되는 모델은 입력 데이터를 2차원 배열로 받습니다. 이 때문에 파이썬의 배열 라이브러리인 넘파이 사용에 익숙해져야 합니다. 데이터가 큰 경우 리스트와 배열은 큰 성능 차이를 갖게 됩니다.

모든 데이터를 학습시키면 알고리즘의 성능을 평가할 테스트 데이터가 없게 돕니다.
따라서 학습에 필요한 훈련 데이터 뿐 아니라, 성능을 평가하는 테스트 데이터도 필요합니다.
이때 특정 샘플이 과도하게 많은 샘플링 편향이 일어나지 않게 샘플 데이터를 잘 섞어서 훈련 세트와 테스트 데이터를 나누어야 합니다.
Chapter 2-1 에서는 numpy.shuffle 을 이용해 나누어 주지만, 샘플의 비율을 유지하면서 테스트 데이터를 만드는 간편한 함수가 존재합니다.

https://colab.research.google.com/drive/1yei0eTIgUu1LF08QGWgmMDFfLQbHkY5f

코드를 보면 작은 도미 (25, 150) 을 넣었더니 빙어가 나왔다.
왜 이런 결과가 나왔나 확인해보려 입력한 데이터와 가까운 5개의 점을 표시해봤더니
5개 중 4개가 빙어였다. 시각적인 느낌과 달라 자세히 보니 무게는 0 ~ 1000인데에 비해 길이는 10 ~ 40 으로 x축, y축 스케일 단위가 다르다는 것을 알 수 있다.
x축과 y축의 비율을 맞춰주니 길이는 무게에 비해 영향력이 굉장히 적다.
이를 반영하는 방법은 없을까?
이것이 바로 데이터 전처리가 담당하는 영역이다.

데이터의 평균값을 빼고 표준 편차로 나눈 표준화 과정을 거치면 각각의 특성이 스케일 단위와 관련 없이 동일한 영향력을 행사하게 된다.
데이터 전처리 (표준화) 이후 재학습 및 테스트를 해보니 정상적으로 도미를 예측하는 결과가 나온다.

대부분의 머신러닝 알고리즘은 특성들의 스케일이 다르면 잘 작동하지 않는다. 따라서 전처리 과정이 필요하다. 또한 표준화 말고 다른 전처리 방법도 존재한다.

Q& A