1️⃣ <처음보는 용어집>

* Broadcasting : Broadcasting은 다른 모양(shape)을 가진 배열들 간에 산술 연산을 수행할 수 있게 해줍니다. 이때 작은 배열이 자동으로 큰 배열의 모양에 맞게 확장되어 연산이 수행됩니다.(출처: <https://numpy.org/doc/stable/user/basics.broadcasting.html>)
* Fine-tuned NLP 모델 : Fine-tuned NLP 모델은 사전 훈련된 언어 모델을 특정 NLP(Natural Language Processing) 작업에 맞게 추가적인 학습을 거친 모델을 말합니다. 이러한 모델은 일반적으로 대규모 텍스트 데이터로 미리 훈련된 사전 훈련된 모델을 기반으로 합니다.(출처: <https://huggingface.co/docs/transformers/training>)
* Transformer : Transformer는 자연어 처리 분야에서 주로 사용되는 딥러닝 모델 중 하나로, 2017년에 발표된 논문 "Attention is All You Need"에서 제안되었으며, 이전까지는 주로 순환 신경망(RNN)을 기반으로 한 모델들이 많이 사용되었지만, 기존의 순차적인 처리 방식 대신에 어텐션 메커니즘을 이용하여 입력과 출력 사이의 상호작용을 모델링이다.(출처: <https://wikidocs.net/31379> )
* Self-supervised learning (자기 지도 학습) : 라벨이 없는 데이터로부터 모델이 학습하는 기계 학습 기법. 입력 데이터에서 특정 속성을 예측하거나 유용한 표현을 생성하는 방식으로 훈련됨. 데이터의 내재된 구조나 패턴을 활용하여 의미 있는 표현을 학습하며, 이후에 다른 작업에 활용할 수 있음. (출처- 위키독스)
* Reinforcement learning (강화 학습) : 상호 작용하는 환경에서 의사 결정을 내리는 것에 초점을 둔 기계 학습의 한 분야. 시행착오를 통해 에이전트가 행동을 선택하고 보상 또는 벌점의 형태로 피드백을 받는 방식으로 학습함. (출처- 위키독스)
* Back propagation (역전파) : 인공 신경망에서 가중치와 편향을 업데이트하기 위해 사용되는 알고리즘. 출력 값과 실제 값 사이의 오차를 역으로 전파하면서 각 뉴런의 가중치를 조정하고, 오차를 최소화하는 방향으로 네트워크를 학습. (출처- 위키독스)
* ANN (인공신경망) : 인간의 뇌의 동작을 모방한 기계 학습 모델. 다수의 뉴런으로 구성되며, 입력과 가중치를 곱하고 활성화 함수를 통과시켜 출력을 계산함. 딥러닝의 근간으로 다양한 복잡한 문제를 해결하는 데 사용되며, 이미지 인식, 자연어 처리, 음성 인식 등 다양한 분야에서 활용됨. (출처- 위키독스)
* 알렉스넷 : 알렉스넷은 딥러닝의 전성기를 연 변화의 시작이었습니다.  아주 복잡한 이미지나 영상일지라도 적절한 알고리즘 구조, 충분한 데이터에 기반한 학습, 컴퓨터의 성능만 뒷받침되면 탁월한 시각 인지 능력을 보여줄 수 있다는 것을 증명했으니까요. 아래 실험 결과를 보면 일부 오류가 있는 결과물도 어느 정도는 정답에 근접하게 인식해 낸 것을 알 수 있습니다. 구체적으로 알렉스넷은 뇌 구조를 본 딴 인공신경망 모델인 합성곱 신경망(CNN,Convolutional Neural Networks)을 사용해 심층 신경망(Deep NeuralNetwork)을 구현했습니다. 또한 기존 머신러닝에서 CPU를 활용했던 것에 비해 알렉스는 병렬 연산을 고속으로 처리해 동시다발적인 연산에 유리한 GPU를 딥 러닝 연구에 활용하였죠. 알렉스넷 이후 GPU 기반 딥 러닝 기술이 꾸준히 개발되면서 현재 ILSVRC에 참여하는 팀들은 인간의 이미지 인식률을 뛰어넘는 결과를 보여주고 있습니다.
* DQN(Deep Q-Network) : 딥러닝과 강화학습을 결합하여 인간 수준의 높은 성능을 달성한 첫번째 알고리즘.(출처: <https://ai-com.tistory.com/entry/RL-%EA%B0%95%ED%99%94%ED%95%99%EC%8A%B5-%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98-1-DQN-Deep-Q-Network>)
* Optimizer : loss함수의 최솟값을 찾는 것 = 최적화(Optimization), 최적화 알고리즘 = Optimizer(출처 : <https://velog.io/@freesky/Optimizer>)
* Adam optimizer : RMSProp와 Momentum 두가지를 섞은 알고리즘으로 진행하던 속도에 관성을 주고 최근 경로의 곡면의 변화량에 따른 적응적 학습률을 갖은 알고리즘(출처 : <https://velog.io/@freesky/Optimizer>)
* Residual network(ResNet/잔차 신경망) : 스킵 연결을 통해 잔차를 학습하도록 만들어진 인공지능 신경망. 일반적인 딥러닝 신경망 모델보다 예측 정확도가 높음.(출처 : <https://ko.wikipedia.org/wiki/%EC%9E%94%EC%B0%A8_%EC%8B%A0%EA%B2%BD%EB%A7%9D> )
* 자기상관관계: 시계열의 시차 값 사이의 선형 관계를 측정함.
* 이산확률함수: 이산형 확률변수로 나타낸 함수. 이산형 확률변수란, 0이 아닌 확률값을 갖는 확률 변수를 ‘셀 수 있는’ 때를 말함. ex. 동전 던지기
* 데이터 정규화: 일반화 오차를 줄이기 위한 머신러닝의 모형이나 알고리즘의 수정
* 규제: 데이터 모델링에서 과대적합을 해결하는 방법 중 하나. 가중치의 값이 커지지 않도록 제한하는 기법으로, 가중치를 규제하면 모델의 일반화성능이 올라간다.
* 손실함수: 머신러닝에서 모델의 예측값과 실제 값의 오차를 최소화하는 것이 중요한데, 이 오차를 정의한 함수를 손실함수라고 한다.
* 경사하강법: 머신러닝 모델의 옵티마이저의 한 종류이다. 옵티마이저는 주어진 데이터에 맞게 모델의 파라미터를 최적화시켜줌 (=손실함수의 값이 최대한 작아지도록 함). 특히 경사하강법은 기울기를 이용하여 손실함수의 값을 최소화한다.
* 정사영: 평면 밖의 점에서 평면으로 그은 수선의 발.
* 제2 코사인법칙: 삼각형의 길이가 각각 a,b,c라고 할 때,

a^2 = b^2 + c^2-2bc cos A

b^2 = a^2 + c^2-2ac cos B

c^2 = b^2 + a^2-2ab cos C 인 공식을 말한다.

* 내적: 머신러닝에서 두 벡터의 유사도가 어느정도 비슷한 방향으로 가는지 알 수 있음
* Robustness: 이상치가 등장했을 때, 손실함수가 얼마나 영향을 받는지를 뜻함
* Stability: 모델이 비슷한 데이터에 대해 얼마나 일관적인 예측을 할 수 있는가
* Latex 타입 : 수식, 행렬, 함수 등을 표현하기 위한 표현형(출처: https://kr.mathworks.com/help/symbolic/sym.latex\_ko\_KR.html)
* 머신러닝 model : 입력 데이터를 의미 있는 출력으로 변환하는 과정. 머신러닝에서는 샘플 데이터로부터 학습을 통해 의미있는 출력을 해주는 모델을 만들어내어 일반적인 작업에 적용시키고자 함(출처: https://modulabs.co.kr/blog/deep-learning-2/)
* 머신러닝 loss function : 손실함수는 예측값과 실제값의 차이를 구하는 기준이 되는 함수. 오차값을 통해 모델의 매개변수를 업데이트하여 손실함수를 줄이는 방향으로 모델을 수정하여 해당 모델이 적합한 모델이 되도록 해줌(출처: <http://dmqm.korea.ac.kr/activity/seminar/326>)
* DQN : DQN이라는 용어는 강화학습 분야의 리딩그룹 DeepMind에서 발표한 연구 ["Playing Atari with Deep Reinforcement"](https://arxiv.org/pdf/1312.5602.pdf)에서 처음 나왔다.

해당 논문에서 소개하는 DQN은 구체적으로 어떻게 사용되었을까?

텍스트, 도표, 디자인이(가) 표시된 사진

자동 생성된 설명

We present the first deep learning model to successfully learn control policies directly from high-dimensional sensory input using reinforcement learning. The model is a convolutional neural network, trained with a variant of Q-learning, whose input is raw pixels and whose output is a value function estimating future rewards. We apply our method to seven Atari 2600 games from the Arcade Learning Environment, with no adjustment of the architecture or learning algorithm. We find that it outperforms all previous approaches on six of the games and surpasses a human expert on three of them.

위의 문장은 논문에서 발췌한 내용으로, raw pixel을 input으로 받아, value function(≈future rewards)를 output으로 반환하는 Q-Learning의 parameter를 학습하는 Convolutional Neural Network(CNN)을 사용했다고 한다. 즉, 고차원의 sensory input을 통해 control policies를 다이렉트로 학습하는 Deep learning model이라고 볼 수 있다.

실제 세계의 문제를 해결하기 위해 고차원 데이터를 거의 무조건적으로 활용할 수 있어야 한다. 이전까지는 이 고차원 데이터를 활용하기 힘들어 Hand-crafted features를 이용했지만, DQN의 등장으로 고차원 데이터를 직접적으로 활용할 수 있게 됩니다.

(출처: https://velog.io/@sjinu/%EA%B0%9C%EB%85%90%EC%A0%95%EB%A6%AC-7.-DQNDeep-Q-NEtwork)

2️⃣ <팀 미션 - 2문제>

문제 1. 파이썬의 대표적 시각화 도구 라이브러리인 matplotlib의 기본 사용법에 대해 숙지하고, matplotlib에서 사용되는 여러 graph들에 대해 조사해 보자.

또, 해당 graph들에 어느 데이터가 실무에서 적합하게 쓰일지 조사해 보자. [ 주제 : 파이썬 ]

matplotlib 설명

* Matplotlib은 미국의 신경생리학자로 뇌전증 환자의 피질뇌파검사(electrocorticography) 시각화를 연구하던 John D. Hunter에 의해 개발되었다. Python을 기반으로 작동하며, 수학이나 통계적인 그래프는 물론, 공학·물리학·의학 등 다양한 분야, 다양한 모델의 시각화에도 활용된다. 실제로 2008년 화성탐사선 피닉스(Phoenix)의 착륙에도 쓰였다고 한다.
* Matplotlib은 시각화에 특화된 라이브러리이기 때문에, 실제로 사용될 때는 대부분 다른 라이브러리와 함께 쓰인다. 그중에서도 NumPy와 함께 활용하게 되는 경우를 많이 확인할 수 있다. 이는 NumPy가 수치 연산에 특화된 라이브러리이기 때문이기도 하지만, Matplotlib이 NumPy에 기반하는 부분이 있기도 하기 때문이다. Matplotlib의 시각화 기능에 있어서 핵심이 되는 pyplot이라는 하위 패키지는 그 자체가 NumPy의 arrays를 기반으로 작동한다.
* Matplotlib는 데이터를 그래프의 형태로 최종 도출해야 하는 데이터 분석(data analysis) 분야에서 가장 유용하게 활용된다고 할 수 있다. 또한 AI 분야에서는 딥러닝 과정에서 나타나는 학습 성과를 각 세부과정 별로 추적하면서 가시적으로 산출하는 데에 활용된다. 딥러닝은 인간이 확인할 수 없는 복잡한 숫자와 연산 과정을 거친다. 따라서 중간 과정에서 문제가 나타나거나, 성능이 떨어지는 부분이 발생하더라도 이를 알아채기가 어렵다. 이와 같은 문제를 해결하기 위해 Matplotlib으로 학습 과정을 시각화하여 그 진행 양상과 결과를 확인할 수 있는 것이다.

**출처 : [네이버 지식백과]** [Matplotlib](https://terms.naver.com/entry.naver?docId=6653663) (AI 용어사전)

* matplotlib.pyplot 모듈은 MATLAB과 비슷하게 명령어 스타일로 동작하는 함수의 모음으로, 모듈의 각각의 함수를 사용해서 간편하게 그래프를 만들고 변화를 줄 수 있다.
* matplotlib는 2차원 그래픽 패키지로, 커맨드 방식(Pyplot API)으로 그래프를 그릴 수 있다. 주로 plt라는 alias로 불러들여 사용하게 된다.

원하는 plot을 지정하여 그래프를 그릴 수 있고(.plot(), .bar(), hist(), pie()), 그래프의 제목이나 축의 범위를 지정하는 등의 추가적인 작업도 수행 가능하다.

matplotlib를 통해 그래프를 그리는 방식은 3가지로 구분할 수 있는데,

1. Pyplot API(커맨드 방식)

import matplotlib.pyplot as plt

import numpy as np

x = np.linspace(0,1,50)

y1 = np.cos(4\*np.pi\*x)

y2 = np.cos(4\*np.pi\*x)\*np.exp(-2\*x)

plt.plot(x,y1,'r-\*',lw=1)

plt.plot(x,y2,'b--',lw=1)

plt.plot()으로 원하는 값을 명령어로 바로 입력하여 그려내거나

1. 객체지향 API 이용

fig = plt.figure() # 직접 Figure 객체를 생성  
ax = fig.subplots() # 직접 axes를 생성  
ax.plot(x,y1,'r-\*',lw=1) # 생성된 axes에 대한 plot() 멤버 직접 호출  
ax.plot(x,y2,'b--',lw=1)

axes라는 container 객체를 만들어 해당 객체에 plot을 그려냄

1. 1, 2번을 조합하여 Figure와 Axes를 plt.subplots()라는 편의함수를 사용하여 그림

import matplotlib.pyplot as plt

import numpy as np

x = np.linspace(0,1,50)

y1 = np.cos(4\*np.pi\*x)

y2 = np.cos(4\*np.pi\*x)\*np.exp(-2\*x)  
fig,ax = plt.subplots()  
ax.plot(x,y1,'r-\*',lw=1)  
ax.plot(x,y2,'b--',lw=1)

(출처: 위키독스(<https://wikidocs.net/14604)>)

- matplotlib 기본적인 사용법 숙지

* 기본 그래프
* pyplot.plot()함수에 하나의 숫자 리스트를 입력하면 아래와 같이 그래프 그려진다.

plot()함수는 리스트의 값들이 y값들이라 가정하고, x 값 [0,1,2,3] 을 만들어낸다.

show()함수를 사용하여 그래프 화면에 나타낸다.

라인, 그래프, 도표, 경사이(가) 표시된 사진

자동 생성된 설명

* 스타일 지정 : x, y 값 인자에 대해 그래프 선의 색상과 형태를 지정하는 포맷 문자열을 세번째 인자에 입력할 수 있다.

plt.plot([10, 20, 30, 40], [1, 4, 9, 16], 'rs--')

plt.show()

라인, 스크린샷, 그래프이(가) 표시된 사진

자동 생성된 설명

* 첫번째값은 색상, 두번째값은 마커 스타일, 세번째값은 선 스타일을 의미
* 기본 포맷 문자열은 ‘b-‘이고 파란색(‘blue’)와 선(‘-‘)을 의미
* ‘rs-’는 빨간색(‘red’)와 사각형(‘ㅁ’), 선 모양('--')을 의미
* 색상을 지정하는 방법은 색 이름이나 약자 또는 #문자 RGB코드를 사용한다.

마커의 종류는

. point marker

o circle marker

\* star marker

+ plus marker 이외에도 여러 종류가 존재한다.

선 스타일

- solid line style

-- dashed line style

-. dash-dot line style

: dotted line style 이외에도 여러 종류가 존재한다.

기타 선 색깔, 선 굵기, 선 스타일, 마커 종류, 마커 선 색, 마커 선 굵기, 마커 내부 색 등 다양한 스타일들이 존재한다.

* matplotlib.pyplot 모듈의 xlabel(), ylabel() 함수를 사용하면 그래프의 x, y 축에 대한 레이블을 표시할 수 있습니다.
* 범례 (Legend)는 그래프에 데이터의 종류를 표시하기 위한 텍스트입니다.

matplotlib.pyplot 모듈의 legend() 함수를 사용해서 그래프에 범례를 표시할 수 있습니다.

- matplotlib graph들의 특징과 실무에서의 사용

1. 선 그래프 (Line Plot):
   * 데이터 변화를 시간에 따라 추적하고자 할 때 유용.
   * 주식 시장의 가격 변동, 기온 변화, 센서 데이터의 시계열 등을 시각화할 때 많이 사용됨.

라인, 그래프, 도표, 경사이(가) 표시된 사진

자동 생성된 설명

1. 막대 그래프 (Bar Plot):
   * 범주형 데이터의 크기를 비교하고자 할 때 유용.
   * 항목 간의 상대적인 크기, 그룹 간의 비교, 투표 결과 등을 시각화할 때 사용됨.

텍스트, 스크린샷, 도표, 직사각형이(가) 표시된 사진

자동 생성된 설명

1. 히스토그램 (Histogram):
   * 데이터의 분포를 보여주고자 할 때 유용.
   * 데이터의 빈도를 나타내며, 데이터의 분포 형태를 확인할 수 있음. 주로 데이터 분석이나 통계적 분석에 사용됨.

스크린샷, 사각형, 직사각형, 라인이(가) 표시된 사진

자동 생성된 설명

1. 산점도 (Scatter Plot):
   * 두 변수 간의 관계를 시각화하고자 할 때 유용.
   * 데이터의 패턴, 상관 관계, outlier 등을 파악할 수 있음. 주로 데이터 포인트의 분포를 확인하는 데에 사용됨.

스크린샷, 사각형, 라인이(가) 표시된 사진

자동 생성된 설명

1. 파이 차트 (Pie Chart):
   * 전체에 대한 부분의 상대적인 비율을 보여주고자 할 때 유용.
   * 범주별 구성 비율, 시장 점유율, 예산 분배 등을 시각화할 때 사용됨. 단, 데이터 종류가 많을 경우, 정보 전달이 어려울 수 있음.

원, 스크린샷, 도표, 텍스트이(가) 표시된 사진

자동 생성된 설명

1. 히트맵 (Heatmap):
   * 두 변수의 cross table을 시각화하고자 할 때 유용.
   * 데이터의 밀도, 상관 관계, 패턴 등을 확인할 수 있으며, 주로 행렬 형태의 데이터를 시각화하는 데에 사용됨.

패턴, 스크린샷, 픽셀, 라인이(가) 표시된 사진

자동 생성된 설명

1. 상자 그림 (Box Plot):
   * 데이터의 분포와 outlier를 시각화하고자 할 때 유용.
   * 데이터의 중앙값, 사분위수, 이상치 등을 파악할 수 있으며, 주로 다른 그룹 또는 범주 간의 비교를 위해 사용됨.

도표, 직사각형, 평면도, 기술 도면이(가) 표시된 사진

자동 생성된 설명

1. 스템 플롯(Stem Plot)
   * 바 차트와 유사하지만 폭(width)이 없는 스템 플롯(stem plot)
   * 주로 이산 확률 함수나 자기상관관계(auto-correlation)를 묘사할 때 사용된다.

라인, 스크린샷, 사각형, 평행이(가) 표시된 사진

자동 생성된 설명

1. 바이올린 플롯(Violine Plot)
   * 데이터의 분포와 범위를 한눈에 보기 쉽게 나타내는 그래프 형식
   * 박스 플롯과 비슷하지만 더 실제에 가까운 분포를 알 수 있다는 장점이 있음

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 버블차트(Bubble Chart)
   * 산점도에서 데이터가 2차원이 아닌 3차원 혹은 4차원의 경우
   * 점 하나의 크기 혹은 색깔을 이용하여 다른 데이터값을 나타냄

스크린샷, 사각형, 다채로움이(가) 표시된 사진

자동 생성된 설명

1. Imshow
   * 이미지 데이터처럼 행과 열을 가진 행렬 형태의 2차원 자료의 크기를 색깔로 표시함
   * 데이터 수치를 칼라맵(color map)을 통해 색으로 바꿀 수 있음
   * Imshow 함수는 자료의 시각화를 돕기위해 다양한 2차원 인터폴레이션을 지원함

스크린샷, 사각형, 패턴, 다채로움이(가) 표시된 사진

자동 생성된 설명

1. 컨투어 플롯(Contour Plot)
   * 입력 변수가 x, y 2개이고 출력 변수가 z 하나인 경우 3차원 자료가 되는데, 이를 등고선(contour) 형태로 시각화하는 방법

만화 영화, 그래픽, 스크린샷, 클립아트이(가) 표시된 사진

자동 생성된 설명

1. 3D 서피스 플롯(3D Surface Plot)
   * 컨투어 플롯과 달리 3차원 전용 axes를 생성하여 3차원 데이터를 입체적으로 표시함

스크린샷, 텍스트, 디자인이(가) 표시된 사진

자동 생성된 설명

[토론]

Matplotlib 파트는 자료조사에서 서로 몰랐던 내용들이 포함되어 있어 따로 토론을 진행하지는 않았습니다.

[결론]

Matplotlib는 다양한 분야에서 시각화를 위해 사용하는 pyhon 모듈이다.

데이터 분석을 통해 얻어낸 최종 결과를 시각화 하기도 하고, 여러 단계에 걸쳐 이뤄지는 딥러닝이나 머신러닝의 단계별 처리 과정에서 분석에 이상이 발생하지 않는지를 추적하는데에도 이용할 수 있다.

시각화를 위해 다양한 기능들(그래프 함수, 레이블 표시, 범례, 색 지정 등)을 지원하고 데이터의 특징에 따라 주로 사용하는 plot이 달라지며, 어떤 데이터에 어떤 plot을 사용하는 것이 적합할지를 알면 효과적인 시각화를 해낼 수 있다.

문제 2. 벡터의 노름은 데이터를 정제하는 방법 중 정규화(Normalization), 규제(Regularization) 등에서 유용하게 쓰여진다. 벡터의 노름의 구조와 개념에 대해 더 자세히 알아보고 머신러닝 분야에서 어떻게 쓰이는지 자세히 알아보자.

또, L1 노름과 L2 노름이 머신러닝 분야에서 어떤 차이점을 가지고 사용되는지에 대해 비교 분석해보자. [ 주제 : 벡터 ]

벡터의 노름은 벡터의 크기를 측정하는 방법 중 하나로, 벡터의 길이나 크기를 나타낸다. 일반적으로 노름은 벡터의 원소들을 제곱하여 더한 후 제곱근을 취하는 과정을 통해 계산된다. 벡터의 노름은 보통 ||x|| 또는 ||x||p로 표기된다. (x = 벡터, p = 노름의 차수)

벡터의 노름은 주어진 데이터의 스케일을 조정하고 정규화하는 데 사용 되며, 정규화는 데이터의 범위를 조정하여 특성 간의 크기 차이를 줄이고, 데이터의 분포를 더 잘 이해하고 처리할 수 있게 도와준다. 이를 통해 머신러닝 모델의 성능을 향상시키고 예측 결과를 개선할 수 있게된다.

머신러닝 분야에서는 주로 L1 노름과 L2 노름을 사용하며, 이 두 노름은 벡터의 크기를 계산하는 방법에 차이가 있다.

L1 노름은 벡터의 원소들의 절댓값을 더한 값으로 정의된다.

||x||1 = |x₁| + |x₂| + ... + |xₙ|

L2 노름은 벡터의 원소들의 제곱을 더한 후 제곱근을 취한 값으로 정의된다.

||x||2 = √(x₁² + x₂² + ... + xₙ²)

***머신러닝 분야에서의 사용 -***

1. L1 노름

- L1 노름은 벡터의 절댓값을 사용하므로 원점에서 L1 거리에 해당.

- 벡터의 원소들 중 일부가 0이 될 수 있어 희소성을 가질 수 있음.

- L1 노름은 이상치에 더 민감하게 반응함.

- 특성 선택(feature selection)이나 차원 축소(dimensionality reduction)에서 사용됨.

- L1 정규화(L1 regularization)로 불리는 Lasso 회귀와 같이 모델의 가중치를 제한하는 데에 사용됨.

2. L2 노름

- L2 노름은 유클리드 거리에 해당하며, 원점에서 L2 거리에 해당됨.

- 모든 원소들의 영향을 균등하게 고려함.

- L2 노름은 이상치에 대해 덜 민감하며, 이상치가 전체 모델에 큰 영향을 미치지 않음.

- 회귀 문제나 분류 문제에서 모델의 가중치를 제한하는 데에 사용됨.

- L2 정규화(L2 regularization)로 불리는 릿지 회귀와 같이 모델의 가중치를 제한하는 데에 사용됨.

L1노름과 L2노름의 종류에 따라 기하학적 성질이 달라지는데 2개의 노름을 사용해서 각각 원을 구하게 되면 다음과 같다.

도표, 라인, 그래프, 폰트이(가) 표시된 사진

자동 생성된 설명

L1의 경우 이동한 거리이기 때문에 중간점인 (0.5,0,5)가 이 되어 위에 원이 성립하고 L2의 경우 L1의 중간점으로 계산했을 때 가 되어 L1과의 차이를 보이게 된다. 여기서 알 수 있는 점은 주어진 점의 크기가 작을 경우 L2의 값이 더 작아지게 되며 에러에 더작은 패널티를 주게 된다.

L1과 L2 노름이 오차를 구하는데 쓰인다고 할 때, L2는 오차의 제곱을 더하기 때문에 이상치에 더 큰 영향을 받는다. 따라서 아웃라이어의 영향을 덜 받고 싶다면 L1 norm을 사용해서 손실을 계산하고 아웃라이어를 신경써야 하면 L2 loss를 사용하는 것이 좋다.

L1의경우 lasso에서 사용되는데 가중치의 원소가 0이 되거나 0에 가깝게 되도록 해야하는데 여기에 L1노름이 사용되어 L1노름이 작아질수록 가중치들이 0에 가까워진다. 가중치들이 0이 됨으로써 그에 해당하는 특성들을 제외시켜준다. 결과적으로 중요한 특성이 무엇인지 알게 되어 모델 해석력이 좋아진다.

l2의 경우 릿지에서 사용되는데 라/소와 다르게 가중치가 0에 가까워 질 뿐 0이 되지는 않는다. 특성이 많은데 그중 일부분만 중요하다면 라쏘가 특성의 중요도가 전체적으로 비슷하다면 릿지를 사용한다.

[토론]

* L1 노름과 L2 노름은 모델 학습에서 가중치를 제한하거나 특성의 중요도를 결정하는 데에 사용된다. L1 노름은 희소한 솔루션을 유도하며, 중요한 특성만을 선택하여 모델을 구성할 수 있다. 반면에 L2 노름은 모든 특성을 균등하게 고려하여 부드럽고 안정적인 솔루션을 유도한다. 따라서 데이터 셋의 특성과 모델의 목적에 맞게 L1 노름 또는 L2 노름을 선택하여 사용해야 한다.
* 요약하면, L1 노름은 희소성을 강조하고 특성 선택에 적합하고, L2 노름은 모든 특성을 고려하는 일반화 성능 개선에 적합한 것 같다.

알아보니 보통 실제로는 L1 노름과 L2 노름을 함께 사용하는 경우가 많이 보였으며 이를Elastic Net이라고 하며, Elastic Net은 L1 노름과 L2 노름을 조합하여 모델의 성능과 특성 선택의 장점을 모두 활용하는데 사용하는 것 같았다.

* 과제 중 질문사항  
  https://seongkyun.github.io/study/2019/04/18/l1\_l2/

이 블로그를 보면, L1은 때때로 서로 다른 벡터라도 같은 값을 가지게 할 수 있지만, L2는 서로 다른 벡터라면 unique한 값을 가지게 한다고 합니다. 그런데 이런 특징으로 L1 norm은 feature selection에 사용 가능하다는데, 이게 무슨 말인지 모르겠어요. 사실 갑자기 L1이 특정 feature 없이도 같은 값을 낼 수 있기 때문에 feature들을 0으로 처리해버리는 게 가능하고, 그래서 feature selection이 가능하다고 하는데 무슨말인지 이해가 어렵네요.

[결론]

데이터를 벡터들의 집합인 행렬로 계산하는 머신러닝에서는 모델을 일반화하는 정규화 과정에서 벡터의 노름이 사용된다.

각 노름이 어떤 거리를 의미하는지를 파악하고 그 특징에 따라 적합한 방법과 실제로는 어떻게 응용되는지를 알 수 있었다. 다양한 머신러닝 모델에서 각 모델의 특징과 목적에 따라 L1과 L2를 선택하여 사용하거나 둘을 함께 사용한다는 것을 알았다.

특히 각 노름을 수식과 기하학적인 방법으로 표현하며 그 특징을 한눈에 알 수 있었다.