**파이썬으로 배우는 기계학습 입문+**

교육기관: kmooc.kr

교수 : 한동대학교 김영섭 교수

수강기간: 2022.12.26 ! 2023.2.29

**1-1강. 기계학습 소개**

1. **. 기계학습의 개념**
   1. **개요**
      1. 전통적 프로그램 vs. 기계학습

* 전통적 프로그램: 정해진규칙(프로그래머), 새로운규칙(수정), 자료 축적 불필요
* 기계학습: 스스로 학습, 프로그램 수정 불필요, 자료 축적 필요

(스팸메일 구분, 고양이 구분 알로리즘이 대동소이 함)

* + 1. AI > 기계학습(Machine Learning > Deep Learning
       1. - Machine Learning(기계학습): 인공지능의 한 분야이며, 데이터를 이용하여 학습시키는 방법론
       2. - Deep Learning (심층학습, 심화학습) : 다층적인 인공신경망 알고리즘을 이용한 기계학습의 한 분야

1. **기계학습의 정의**

* **기계 학습(1980~)**
* 컴퓨터가 배울 수 있는 능력. 프로그램으로 정의하지 않아도 컴퓨터가 스스로 학습하여 실행할 수 있는 능력에 관한 연구분야
* 인공지능의 한 분야
* **인공지능**
* Artificial Intelligence(AI). 1950.
* **딥러닝(심층학습/심화학습. 2010~)**
* **기계학습의 한 분야**

1. **기계학습의 종류**
   1. **종류** 
      * 1. - 컴퓨터가 어떻게 학습하는지에 따라서 방법론이 다름.
        2. - 종류 : **지도학습, 비지도학습, 강화학습**
   2. **지도학습**
      1. 방법: 자료로 학습하고 예측
         1. - 예 이미지- 이미지를 각 영역별로 매칭되는 숫자로 된 레이블이 있는 학습자료를 반복적으로 학습하는 과정🡨 지도학습
         2. - 학습 후, 새로운 이미지 시험문제의 답을 예측하게 하는 학습 방법
      2. 특징
         1. - 예측한 값과 실제 정답인 레이블과 비교하여 정확하게 어떻게 예측했는지 여부를 피드백 받아 기계가 열심히 학습을 반복할 수 있음
         2. 예) 사물인식, 날씨예측
         3. - 컴퓨터에게 문제와 답을 주어 그 패턴을 익히도록 하는 학습방법]
         4. - 사람있는 사진을 인터넷에서 찾아내어, CCTC영상에 사람이 있는지 분별하려는 학습 알고리즘
      3. 분류(Classification)와 회귀(Regression)으로 분류

* 분류방법
  + 문제와 답을 주어 학습후 새로운 패턴에 대해서 적절한 답 예측
    - 기상 조건등을 가지고 날씨 흐림, 맑음 분류
* 회귀방법
  + 연속된 값(real number, 실숫값)을 예측하는 것
  + 예. 사진속의 인물 몸무게 예측, 집의 면적으로 집값 예측, 2시간30분 공부한 학생이 받을 퀴즈 점수 예측
  + \*
  1. **비지도 학습**
     1. 방법: 레이블없이 데이터에서 패턴 추정.
  + 예) 신문기사 분석(많은기사들이 경제,시사,스포츠뉴스인지 **그룹핑**하며 식별), 1000만개의 인터넷 사진으로 학습한후 사진에서 고양이 구별
  + 예) 마케팅 활용 목적으로 현재 보유 고객 정보를 나이, 직업, 성별, 위치, 취미, 재산등 자료를 분류(군집화,clustering)하고자 할 때 사용하는 알고리즘
  + 컴퓨터가 스스로 자료를 모델링함. 레이블링이 없음. 스스로 학습을 해가며 데이터 자체에서 패턴을 찾아내는 학습 방법
  1. **강화학습**
     1. 방법: **행동한 것에 대해 보상을 적절히 해주어서 보상을 극대화하는** 방법으로 컴퓨터가 학습
  + 예) 행동 주체 강아지, 강아지가 먹을수 있는 기계 존재
    - 강아지가 이것저것 하면서 조금씩 먹을수 있는 기계라는 것을 알게됨. 버튼 누르면서 조금씩 보상을 받으면서 학습하게 됨.
  + 바둑(2016. 이세돌과 알파고), 게임

상태(State)

**환경**

**(Environment)**

**행동의 주체**

**(Agent)**

보상(Reward)

행동(actions)

* + 예) 영국. 주)Deep Mind의 벽돌부수기
    - 120분후에 잘 맞춤, 4시간후 한쪽으로 집중벽돌 구멍을 뚫으면 점수 높게 받음
    - 구글이 딮마인드회사 인수 🡪 알파고. 이세돌과 대국 🡪 알파고. 은퇴

**1-2강 기계학습 개발 환경**

1. **왜 파이썬인가?**
   1. **파이썬의 장점**

* 간단하고 배우기 쉬움
* 컴파일링 없는 스크립트 언어
* 풍부한 라이브러리(자료 저장, 이미지 처리, 자연어처리)
* 딥러닝 프레임워크의 파이썬 API제공
  + - 딥러닝 프레임워크인 **카페, 텐서플로, 파이토치, 케라스등** 유명프레임워크에서 파이썬 API를 제공)

1. **파이썬버젼과 튜토리얼**

* 파이썬 3.4(3.X)버젼 사용 – 2.\* 대와는 호환되지 않음
* **튜토리얼 :** [**http://learpython.org**](http://learpython.org)
* **Dos창에서 $conda update conda 🡨 업데이트**

1. **파이썬 라이브러리**

* 표준라이브러리: math, random, pickle, csv, os, time, urlib, etd
* 외부라이브러리
  + Numpy : 다차원 배열/행렬 생성 및 프로세싱
  + Matplotlib : 과학 계산용 그래프 라이브러리
* 개발자 개별 라이브러리: 모듈

1. **기계학습 개발 프레임워크**
   1. **Tenorflow**
   * 구글이 오픈소스로 개발, 핵심기술은 C++로 작성
   * 프론트엔드는 파이썬으로 작성, GPU사용
   * 시각화 툴인 TensorBoard제공
   * **용어**
     + 스칼라 : 0차원의 수
     + 벡터 : 1차원의 배열
     + 행열, Matrix : 2차원의 배열
     + **텐서** : n차원의 배열
   1. **Keras(케라스)**

* Tensorflow 를 기반으로 한 framework
* 직관적인 API
* 현재 Tensorflow와 통합되어 Tensorflow안에서도 사용 가능
  1. **Pytoarch**
* Lua로 개발된Torch를 Python API로 개발
* 디버깅이 쉬운 직관적인 코드로 구성
* 동적그래프 : 언제든지 데이터에 따라 모델 조정 가능

1. **기계학습 개발환경**
   1. **IDE : Jupyter Notebook**

* IPython(interactive python)의 장점 모두 물려받음
* 웹브라우서에서 코딩과 문서 작성 모두 가능
* 최고의 파이썬 개발환경
  1. **아나콘다 배포판 설치와 확인**
* **Anaconda**
* **설치시 주의할 점(옵션. 윈도우의 경우)**
  + 공용으로 설치(권장, 한글때문)
  + PATH 설정 체크(권장, 콘솔 사용위해)
  + 설치 directory: 한글로 설정 하지 말 것
* **파이썬 설치버젼 확인**

$ python –-version

$ pip --version

1. **쥬피터 노트북 사용하기**

* **윈도우, 리눅스 버전**
* **쥬피터 노트북 콘솔에서 시작**
  + - Command창 🡪 $python –version
    - $jupytor notebook
    - New 🡪 (쎌에서) In: 1+3+2 🡪 shift+Enter 또는 Ctrl+Enter(실행)
* **쎌에서**

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

x=np.arange((0,6, 0.1)

y= np.sin(x)

plt.plot(x, y)

plt.show()

1. **마크다운 언어 (문서 작성용)**

* 쥬피터 노트북에서 Markdown선택
* 제목 크기 : # , ##, ### (파운드)
* 볼드체 : \_\_ oooo \_\_ : 언더바 2개
* 이탤릭체 : ooo \_ : 언더바 1개
* 유용한 셀 명령어

(셀 편집상태에서 <ESC>, CTRL+m 커맨드 모드)

* + - m : 셀 타입을 마크다운으로 전환
    - y : 셀 타입을 코드로 전환
    - dd: 셀 삭제
    - x : 셀 잘라내기
    - C: 셀 복사
    - S : 파일 저장
    - Shift + m : 아래 셀과 병합
    - Enter : 편집모드로 진입
    - L (엘): 코드 셀의 라인에 번호 on/off

**1-3강 행렬**

**1 왜 행렬인가**

* 연립 방정식 🡪 행렬로 변환하여 간결 하게 표시(계수와 변수를 분리하여 사용)

|  |  |
| --- | --- |
| 연립 방정식 | 4X1 -5X2 = -12  4-21 + 3X2 = 8 |

|  |  |
| --- | --- |
| 행렬 | AX = b  A = |4 -5|, b = | -12 |  |02 3| | 8 | |

* 기계 학습에서 행렬의 이점
  + 간결한 표기
  + 실수 감소
  + 컴퓨터에 적합한 형식

**2 용어의 이해**

* **스칼라** : 크기(O), 방향(X)
  + - 4칙연산이 가능한 물리량
    - 예) 질량, 부피, 거리, 속력, 실수, 정수
* **벡터(Vector)** : 크기와 방향 ( 1차원의 배열)
  + - 속도, 가속도, 운동량, 전기장
* **행렬(matrix)** : 2차원 배열 (네모꼴로 숫자를 배열)
* **텐서(Tensor)** : N차원 배열
  + - 0차원인 스칼라, 1차원인 벡터, 2차원인 행렬, 혹은 N차원, 고차원 모든 행렬을 포함

**3. 행렬의 정의**

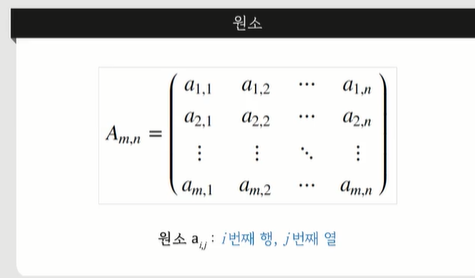
\* 가로줄 : 행(row), 세로줄: 열(column)

텍스트이(가) 표시된 사진

자동 생성된 설명



**4. 행렬과 원소, 표기**



\* 원소: 행렬안의 하나, 하나 숫자

**5. 행 벡터와 열 벡터**



* 행벡터 : 행🡪 1개, 열 🡪 n개
* 열벡터 : 행🡪n개, 열 🡪 1개

**6. 행렬의 곱**

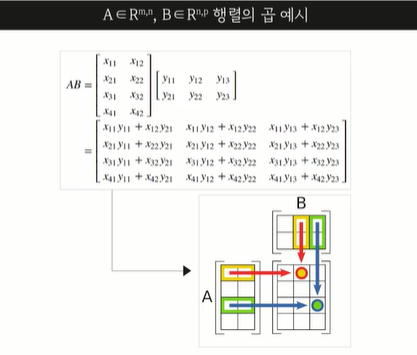
* 행렬의 곱은 다른 행렬의 연산들과 같이 크기가 맞는 경우에만 계산 가능

🡪 앞의 행렬의 열(column) 수와 뒤 행렬의 행(row) 수가 동일 해야 함

텍스트이(가) 표시된 사진

자동 생성된 설명

Cij = Sigma(k=1…n) aik\* bjk (Sigma 는 합)



7. 두 벡터의 내적

* 내적

- 두개의 열벡터(X, Y)가 있을 때, 첫번째 X를 전치하고, Y를 곱하는 것

텍스트이(가) 표시된 사진

자동 생성된 설명

전치 : Transpose)

* + 규칙 : X( T ) \* Y = X \* Y( T ) 🡪 값은 스칼라
* 두벡터의 외적
  + - 두 개의 벡터 X, Y가 주어질떄, 두벡터의 크기 동일할 필요 없음
    - X \* Y( T ) 🡪 스칼라가 아닌 행렬을 얻게 됨

9. 행렬의 곱셈 법칙

텍스트이(가) 표시된 사진

자동 생성된 설명

\* 결합, 분배 법칙 ; O

\* 교환 법칙 : X

**10 . 행렬의 덧셈과 뺄셈**

- 대응하는 원소끼리 연산

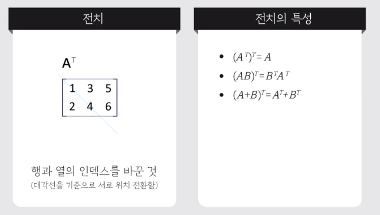
- 행렬의 크기가 서로 동일시에만 가능

텍스트이(가) 표시된 사진

자동 생성된 설명

11. 행렬의 전치

- 행과 열의 인덱스를 바꾼건



**12. 단위 행렬**

\* 대각선이 모두 1이고, 나머지 원소는 0인 행렬

\* 변화가 없음. 곱셈의 항등원이 됨

**1-4강 함수와 뉴론**

1. 함수와 뉴론

* 고양이 사진으로부터 고양이를 인식하기 위해서는 고양이의 특징을 함수로 정의해야 한다.
* 사진s 🡪 기계학습(만능 함수 제조기) 🡪 사람처럼 사물 구분 가능

X(입력, 독립변수) 🡪 함수 f 🡪 Y(출력, 종속변수)

예) x(섭씨) f(x) = 9/5 \* x + 32

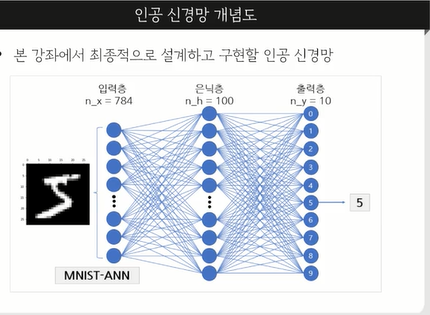
기계학습 : 만능 함수 제조기

1. 뉴론과 신경망
   1. 뉴론(뇌세포)

- 뇌의 기본 단위

- 850억개

* 1. 뉴론 연산자
     + 임계값
     + 입력은 다수, 출력은하나
     + 입력의 합산
     + 신경망 구성
  2. 생물학적 뉴론 모델링
     + 축색돌기는 시냅스를 통해 하나 이상의 뉴런으로 출력을 전달
     + 시냅스는 인간의 뇌에 100조개 존재. 신호의 강약이 조정됨(인공 뉴론에서 가중치에 해당), 학습하고 기억하는 것
     + 수상돌기는 입력신호를 받음
     + 뉴론은 시냅스로 연결되어 있음



1. 인공 뉴론의 구현

입력 🡪 f(x) = 1.6\* x 🡪 출력

mileToKm : 인공 뉴론

plotMileToKm : 뉴론의 계산과정이나 결과 시각화

|  |
| --- |
|  |
| import matplotlib.pyplot as plt  %matplotlib inline  ## f(x) = 1.6\* x (mile 🡪 km로 변환하는 함수)  def mileToKm(x) :  return 1.61 \* x  def plotMileToKm(x,y):  plt.figure()  plt.plot(x,y)  plt.title("Mile to K")  plt.xlabel('Mile')  plt.ylabel(Km, fontsize=20)  plt.show() |

* 1. for-loop를 사용한 함수 호출

// 0~ 4 까지 결과 출력 함수

for mile in range(0, 5):

print("{}mi : {}km".

format(mile, mileToKm(mile)))

|  |
| --- |
|  |

* 함수 결과의 시각화

x = [ 0, 1, 2, 3, 4]

y = [ 0, 1.61, 3.22, 4.83, 6.44 ]

plotMileToKm(x,y)

|  |
| --- |
| 데이터 포인트 시각화 ::직선포함 |
| plotMileToKm(x, y) ## 수정, 실행  plt.plot(x,y) 🡪 plt.plot(x,y, ‘-ob’) ##o : 동그라미, b: blue색상  plt..plot(x,y, ‘-sr’) ## - : 점들을 선으로 연결, s: Square r: red생상 |

* 1. 손코딩방식과 아닌 방식

|  |
| --- |
| 리스트 데이터 만들기(손코딩 방식) |
| x = [ 0, 1, 2, 3, 4]  y = [ 0, 1.61, 3.22, 4.83, 6.44 ] |

|  |
| --- |
| 리스트 데이터 쉽게 만들기(arrange 사용) |
| range(start, end, step) //x 값 만들 때 사용 range(5) // 0~4까지 5:end(제외됨)  linespace(start, end, npoints) |

* **for 루프**

|  |
| --- |
| for 루프로 리스트 y 만들기 |
| y=[]  for mile in range(0, 5):  y.append(mileToKm(mile)) ##append : list class 메소드  print(y) |

* **list comprehension으로** 리스트 만들기

|  |
| --- |
| list comprehension 으로 리스트 만들기 |
| y=[]  y = [ mileToKm(mile) for mile in range(0, 5) ] ##append : list class 메소드  print(y) |

* numpy로 데이터 만들기

|  |
| --- |
| numpy로 데이터 만들기 |
| import numpy as np  x = [0, 1, 2, 3, 4 ] ## 1) 리스트를 직접 만들기  print(x)  xi = np.array(x) ## 2) 리스트를 numpy배열로 전환  print(xi)  xi= np.array(np.arange(0, 5)) ## 3) numpy로 리스트 새로 만들기  print(xi) |

* numpy 로 사용 이점
* pythonic 프로그램

|  |
| --- |
| numpy로 데이터 만들기 |
| import numpy as np  x= np.arange(0, 5)  y= mileToKm(x)  print(x)  print(y)  plotMileToKm(x,y) |

cf. np.linespace(-5, 5, 11) ##11개 선형 균등 간격(등간격) 값을 만듬

* + range() vs. np.arange()
    - range는 list로 변환해야 함
    - np.arange는 리턴값이 list임

ex) print(range(10) 🡪 “range(10)”

Print(list(range(10) 🡪 [0,1,2,3,4,5,6,7,8,9]

Import numpy as np

A= np.arange(1,10)

Print(A) 🡪 [0,1,2,3,4,5,6,7,8,9]

**1-5강. 넘파이 튜토리얼 – 배열의 형상 (#넘파이/인덱싱/슬라이싱)**

1. 넘파이 개요

* 넘파이의 특징
  + - 강력한 다차원 배열과 행렬 연산
    - 다양한 선형 대수학 함수와 난수
    - 간단한 코딩
* Numpy(Numerical Python)
* Numpy

|  |
| --- |
| Numpy 라이브러리 사용법 |
| import numpy  ## numpy.\_\_version\_\_  import numpy as np  ## np.\_\_version\_\_ |

1. 왜 넘파이인가?
   1. 쉬운 다차원 행렬 연산
      1. 예제: 일천만번 곱셈과 합
         1. 두개의 list에 난수를 저장
         2. 두개의 numpy배열에 복사
         3. 각 원소별로 곱하고 합산
      * 한번은 for-loop로 실행
      * 한번은 numpy로 실행
        1. 두 계산의 방법의 비교

|  |
| --- |
| import numpy as np  n=10000000  w=[np.random.random() for \_ in range(n)]  x=[np.random.random() for \_ in range(n)]  # 리스트의 값을 np배열로 복사  wnum = np.array(w)  xnum = np.array(x) |

|  |
| --- |
| ## for 문 사용 계산시 3배 시간 소요  %time // line magic function  total = 0  for i in range(n):  total += w[i] \* x[i]    print(total) |

|  |
| --- |
| ## Numpy array 사용시 쉽게 코딩 가능##  %%time  total = np.dot(wnum, xnum)  print(total) |

1. NumPy 배열의 속성

|  |
| --- |
| Ndarray( 넘파이 클래스) 속성 |
| ndim : 차원, axis 개수, rank  shape: 형상, 각 차원의 배열의 크기  size: 배열의 모든 원소의 개수  dtype: 원소의 자료 형식 |

* 1차원 배열 : 0,1,2 🡨 shape(3, ) //shape(열)
* 2차원 배열 : 0, 1, 2 🡨 shape(2, 3) // shape(축1(행), 축2(열))

3, 4, 5

* 3차원 배열 : 🡨 shape(4,2,3) shape(축1, 축2, 축3)

|  |
| --- |
| Pprint()함수 |
| def pprint(arr)  print(“type:{}, size{}”.format(type(arr), arr.size))  print(“shape:{}, ndim/rank:{}, dtype:{}”.format(arr.shape, arr.ndim, arr.dtype))  print(“Array’s Data:” )  print(arr) |

1. 배열의 생성

* np.array() – 리스트, 튜플이용

a= np.array( [1,2,3,4] ) # right -list

a= np.array( (1,2,3,4) ) # right -tuple

a= np.array(1,2,3,4) # wrong

|  |
| --- |
| import numpy as np  a = np.arange(12)  pprint(a) |

배열이 float형식이고 행:3, 열4 인 행렬 생성

|  |
| --- |
| import numpy as np  a = np.arange(12, dtype=float).reshape(3,4)  pprint(a) |

|  |
| --- |
| 배열 생성 함수 |
| * zeros : 모든 원소 0 * ones : 모든 원소 1 * full : 사용자가 지정한 하나의 값 * empty : 임의의 값 * eye : 단위 행렬 |

np.ones( (3, 4) )

* + - * array ( [ [1,1,1,1 ],
      * [1,1,1,1 ],
      * [1,1,1,1 ] ] )

np.full((2,3 ), 7)

* + - * array ( [ [7, 7, 7 ],

[7, 7, 7 ] ] )

|  |
| --- |
| 데이터 생성 함수 |
| * arange( start, stop, step, dtype=None) * linspace(start, stop, endpoint=True, retstep=False, dtype=None)   //start ~ step 내에서 균일간격으로 num개 데이터 생성   * logspace(start, step, num=50, endpoint=True, base=10.0, dtype=None)   //start ~ step 내에서 로그 스케일로 num개 데이터 생성 |

sample)

|  |
| --- |
| import matplotlib.pyplot as plt  %matplotlib inline  a= np.arange(10)  plt.plot(a, ”or”)  plt.show() |

|  |
| --- |
| import matplotlib.pyplot as plt  import numpy as np  %matplotlib inline  x= np.linspace(0, 2\*np.pi) //1~ 2\*pi 라디안까지 50개 포인트 생성  y= np.sin(x)  plt.plot(x, y)  plt.show() |

|  |
| --- |
|  |

|  |
| --- |
| import matplotlib.pyplot as plt  import numpy as np  %matplotlib inline  a= np.logspace(0.1, 2) // 10의 0.1제곱 ~ 10의 2제곱까지 50개 포인트 생성  plt.plot(a, ‘xb’) // ‘x’자로 blue 색  plt.show() |

1. 배열 인덱싱고 슬라이싱

|  |
| --- |
| 인덱싱(Indexing) |
| * 0부터 시작 * -1 : 배열의 끝(음수 indexing) * : 범위 지정 (start : end)   [2:5] 2~4 [:5] 0 ~4 [5: ] 5,6,.. [:] 모든원소 |

예제) 2차원 배열: shape(3,4)

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |

* 원소 7: A[1, 2]
* 원소 12: A[2,3]
* A의 마지막 원소 : A[-1: -1]
* 마지막 행: A[-1 ]
* 첫번째 열: A[:, 0]
* 마지막 열: A[:, -1]
* A의 위, 왼쪽의 2x2 : A[:2, :2]
* a의 아래, 오른쪽 2x2 : A[-2:, -2: ]
* A의 첫 2열 : A[:, :2]
* A전체 : A[:, :]

|  |
| --- |
| a = np.arange(1,13).reshape(3, 4)  print(a[1, 2])  print(a[-1,-1])  print(a[:, 0] )  print(“a[:, -1] : {}“, a[:, -1] )  print(a[:2, :2])  print(a[-2:, -2:])  print(a[:, :2])  print(a[:, :]) |

* 슬라이싱 – subarray

|  |
| --- |
| a = np.arange(1,13).reshape(3,4)  b = a[:2, :2]  print(b)  b[0.0] =99  print(a) |

* b[0,0], a[0,0] 모두 변경되어 있음

b는 a의 복사본이 아니라 view일뿐 (reference) 의미

8. Boolean 배열 인덱싱

|  |
| --- |
| a= np.random.random(7)  results = a > 0.6  print(results) |

[False, False, False, True, True, True, False]

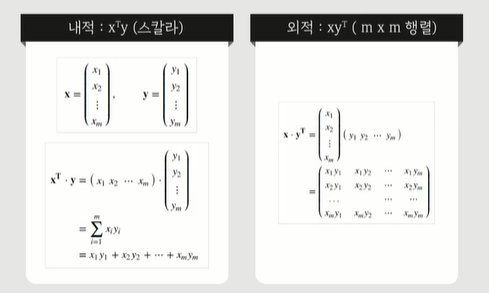
print(np.sum(results)) //3

print(np.argwhere(results)) // 3,4,5

9. 배열 / 행열의 곱

|  |
| --- |
| 두 열 벡터 x,y < R\*m승 |
| * 내적: X(T제공)\*Y 스칼라 * 외적: XY(T제곱) M\*M행렬 |

내적 외적 모두 npdot() 사용



**1-6강. 넘파이 튜토리얼 – 브로드캐스팅**

**#2차원배열 #3차원배열 #난수 배열**

1. **브로드캐스팅**

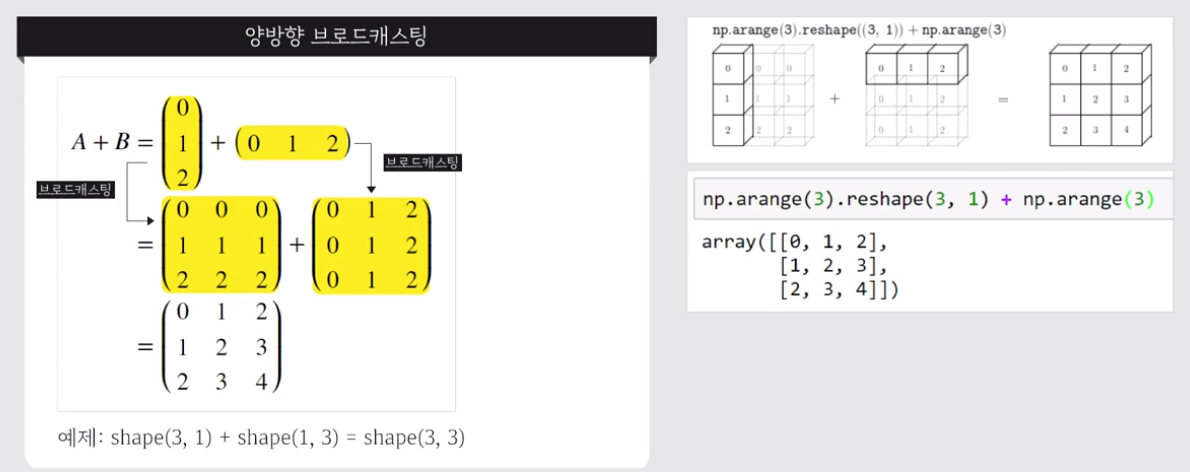
|  |  |
| --- | --- |
| **브로드캐스팅 기본 개념** | |
| **a = np.array([1,2,3])**  **b = np.array([1,2])**  **a+b** | **a = np.array([1,2,3])**  **b = np.array([1])**  **a+b** |

**좌측 연산 : X 우측-연산 : O**

* 브로드캐스팅: “널리 전하다”

|  |  |
| --- | --- |
| **브로드캐스팅 기본 개념** | |
| **A+B = |1 2 3 |**  **|4 5 6 | + (1 0 1)**  **|7 8 9 |**  **|10 11 12 |**  **= |1 2 3 | | 1 0 1 |**  **|4 5 6 | + | 1 0 1 |**  **|7 8 9 | | 1 0 1 |**  **|10 11 12 | | 1 0 1 |** | **a = np.array(np.arange(1, 13)).reshape(4,3)**  **b = np.array([1, 0, 1])**  **print(a+b)** |

* 양방향 브로드캐스팅



## 양방향 브로드캐스팅

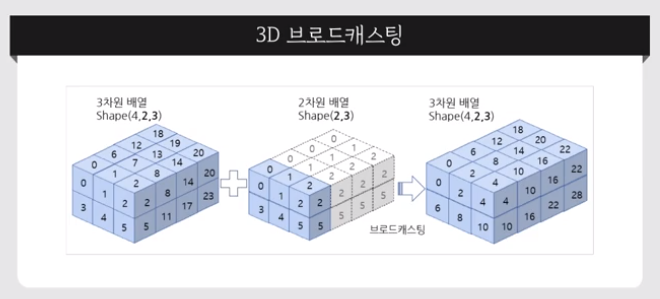
import numpy as np

a = np.array(np.arange(0,3)).reshape(3,1)

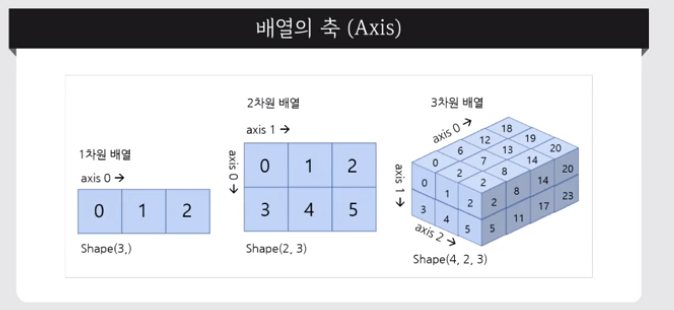
b = np.array(np.arange(0,3)).reshape(1,3)

print(a+b)

* 3차원 브로드캐스팅



1. 배열의 축(Axis) 다루기



|  |
| --- |
| 배열의 2차원 축(Axis) |
| np.sum(a, axis=0) =?  print(“sum(axis=0):”, np.sum(a, axis=0))   * + - * 2차원 배열 Shape(2,3) |

* **배열의 2차원 축의 합**

import numpy as np

a = np.array(np.arange(0,6)).reshape(2,3)

print("a =", a)

print("sum(axis=0):", np.sum(a, axis=0))

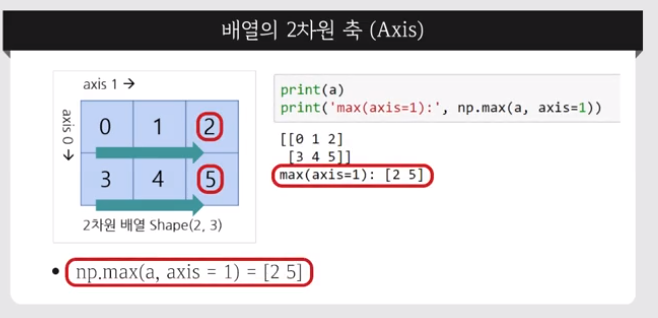
* **배열의 2차원 축의 최대값**

import numpy as np

a = np.array(np.arange(0,6)).reshape(2,3)

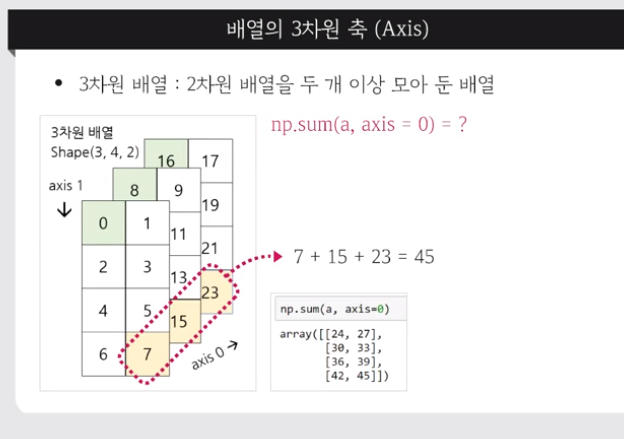
print("a =", a)

print("max(axis=1):", np.max(a, axis=1))



|  |
| --- |
| 배열의 3차원 축(Axis) |
| * 3차원 배열: 2차원 배열을 두개 이상 모아 둔 배열 * 열 : 2, 행: 4, 면: 3   a = np.arange(24).reshape(3, 4, 2)  print(a) |

* 3차원 배열의 축으로의 합계



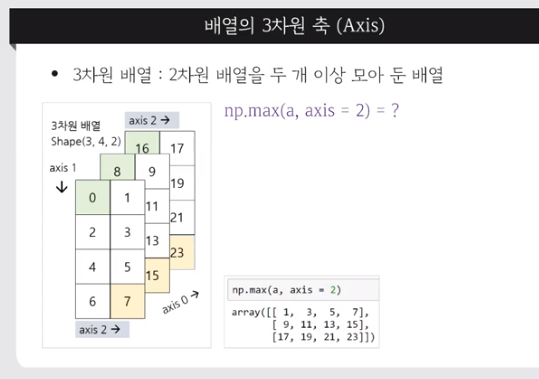
* 3차원 배열의 축으로의 min

# 3차원배열의 축으로의 min

a = np.arange(24).reshape(3, 4, 2)

print("Sum of 3차원 배열:", np.min(a, axis=1))

테이블이(가) 표시된 사진

자동 생성된 설명

1. **난수 배열 생성**

|  |
| --- |
| 난수 배열 생성 함수 |
| * 1. numpy.random모듈      1. randint(low, high=None, size=None, dtype=”1”) :정수 표본을 추출하여 배열을 반환      2. normal(loc=0.0, scale=1.0, size=None)      3. random( (size=None) ) |

예)

## 난수 2차원배열 생성

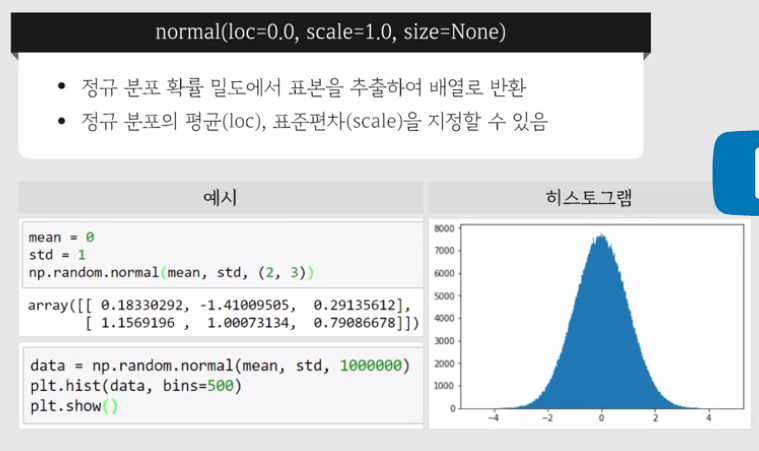
a= np.random.randint(-5, 5, size=(2,4))

print(a)

[[-4 -4 3 3]

[-1 1 3 0]]

|  |
| --- |
| normal(loc=0.0, scale=1.0, size=None) |
| * 정규분포 확률 밀도에서 표본을 추출하여 배열로 변환 * 정규분포의 평균(loc), 표준편자(scale)을 지정할 수 있음   예) mean = 0  std = 1  np.random.normal(mean, std, (2,3)) |



|  |
| --- |
| random( (size=None) ) |
| * (0, 1) 범위의 난수를 균등 분포에서 표본 추출하여 배열로 변환   np.random.random( (2,3) )  예) data= np.random.random( 1000000 )  plt.hist(data, bins=500)  plt.show() |

* 참고

import random

random.random() ##0.023445

random.randint(1,10) ## 5

random.randrange(1, 3) ## 1 <= X <=2

* Magic command

%time : 명령어 뒤에 한줄 코드가 수행되는데 걸리는 시간 반환

%timeit: time +iteration 명평어 뒤에 한줄 코드를 몇번 반복 수행후 평균시간 반환

%%time : Cell 전체 코드를 모두 수행한후 시간 측정 반환

%%timeit : Celll 코드 전체를 반복 수행후 소요되는 평균시간

%who : 변수명을 tab 으로 단순하게 나열 하여 반환 합니다.

%who\_ls : 변수명을 리스트 형태로 반환합니다.

%whos : 변수명, 유형, 데이터 정보를 상세히 반환합니다.

|  |  |
| --- | --- |
| %who | 변수명을 tab 으로 단순하게 나열 하여 반환 합니다. |
| %who\_ls | 변수명을 리스트 형태로 반환합니다. |
| %whos | 변수명, 유형, 데이터 정보를 상세히 반환합니다. |

**2-1강. 인공뉴론의 동작원리**

1. **인공뉴론의 동작 원리**

**x(입력) 🡪 🡪 y(출력)**

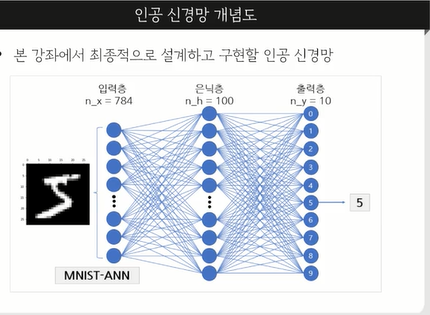
입력값 : 1, x1, x2… :

가중치 : b, w1,w2

임계값 : h(x)

|  |
| --- |
| 인공 뉴론 가중치 |
| w1  y  X1  Y = w1\*x1  y = ax (a:기울기 = W1:가중치(Weight) |

|  |
| --- |
| 뉴론의 동작원리 |
| X1  w1  y  X2  W2  순입력값(net input): = w1\*x1 + w2\*x2  임계값( threshold) :  활성화(activated): >  Y = 0 if (w1\*x1 + w2\*x2 <= )  1 if (w1\*x1 + w2\*x2 > )  **해석: 뉴런에 들어온 순입력의 값이 임계값을 넘어설때만 뉴론이 출력됨**  **(activate라고 함)** |



여러입력과 가중치들과 많은 뉴런들이 연결되어 완성됨.

각 노드들을 연결하는 가중치를 찾아내는 것이 곧 꿈의 함수를 만들어 내는 것임

Y = 0 if (w1\*x1 + w2\*x2 <= )

1 if (w1\*x1 + w2\*x2 > )

를 외쪽으로 넘길수 있는데 이때 b를 편향(bias)라고 함

Y = 0 if (b + w1\*x1 + w2\*x2 <= )

1 if (b + w1\*x1 + w2\*x2 > )

|  |
| --- |
| 뉴론의 동작원리  1 |
| b  w1  X1  y  X2  W2  Y = 0 if (b + w1\*x1 + w2\*x2 <= )  1 if (b+ w1\*x1 + w2\*x2 > )  **해석: threshold인 편향도 하나의 입력값으로 간주함.** |

|  |
| --- |
| 예제 1 |
| 예제 1) 학습을 통해 가중치 W가 학습 되었고 , 임계값 T(쎼타)와 입력값 X1, X2가 주여졌다고 가정시, 뉴런이 활성화될지 판단해 보기  w =(w1, w2) = (0.6, 0.3)  T(세타) = 0.5  입력값 X1, Y1 =(0, 1)   * + - 1. 활성화된다 2) 활성화되지 않는다. ?   solve) net input = w1 \* x1 + w2 \* x2 = 0.6\*0 +0.3\*1 🡪 0.3 < 0.5 므로 “활성화되지 않는다” |

1. AND 게이트 뉴론 구현하기

|  |
| --- |
| AND 게이트 구현 |
| |  |  |  | | --- | --- | --- | | X1 | X2 | Y | | 0 | 0 | 0 | | 0 | 1 | 0 | | 1 | 0 | 0 | | 1 | 1 | 1 |   Y = 0 if (b + w1\*x1 + w2\*x2 <= )  1 if (b+ w1\*x1 + w2\*x2 > ) |

|  |
| --- |
| AND 게이트 구현 예제 |
| * AND게이트 뉴론을 만들고자 함   Y = 0 if (b + w1\*x1 + w2\*x2 <= )  1 if (b+ w1\*x1 + w2\*x2 > )  위 식을 만족하는 아래의 가중치, 편향의 조합중에서 AND게이트를 수행하는 조합을 찾으시요?   1. (w1,w2) = (0.5, 0.5),b=-0.7 2. (w1,w2) = (0.5, 0.5),b=-0.3 3. (w1,w2) = (0.5, 0.5),b=0.2 |

(AND게이트를 수행하는 조합은) 1번

-0.7 + 1\*0.5 + 1\*0.5 = 0.3 🡪 0.3 > 0므로 AND게이트 출력은 1이됨

🡪 AND게이트를 만족함

-0.3 + 1\*0.5 + 1\*0.5 = 0.7 🡪 0.7 > 0므로 AND게이트 출력은 1이됨

-0.3 + 1\*0.5 + 0\*0.5 = 0.2 🡪 0.2 > 0므로 AND게이트 출력은 0이되어야 하나, 1이되므로 🡪 AND게이트를 만족하지 못함

0. 2 + 1\*0.5 + 1\*0.5 = 1.2 🡪 1.2 > 0므로 AND게이트 출력은 1이됨

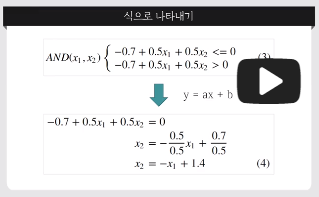
0.2 + 1\*0.5 + 0\*0.5 = 0.2 🡪 0.7 > 0므로 AND게이트 출력은 0이되어야 하나, 출력이 1이되므로 🡪 AND게이트를 만족하지 못함

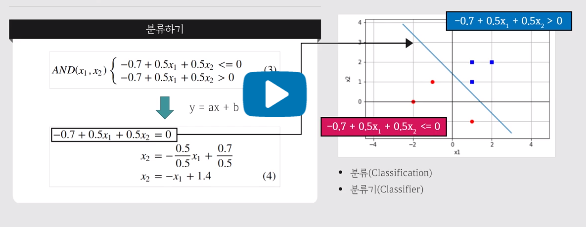
|  |
| --- |
| AND 게이트 구현 실습 |
| * (w1, w2) = **(0.5, 0.5), b=-0.7**   Y = 0 if (b + w1\*x1 + w2\*x2 <= )  1 if (b+ w1\*x1 + w2\*x2 > )  위 식을 만족하는 아래의 가중치, 편향의 조합중에서 AND게이트를 수행하는 조합을 찾으시요?   1. (w1,w2) = (0.5, 0.5),b=-0.7 2. (w1,w2) = (0.5, 0.5),b=-0.3 3. (w1,w2) = (0.5, 0.5),b=0.2 |

|  |
| --- |
| AND 게이트 구현 실습 |
| * (w1, w2) = **(0.5, 0.5), b=-0.7**   Y = 0 if (b + w1\*x1 + w2\*x2 <= )  1 if (b+ w1\*x1 + w2\*x2 > )  1  W0=b  w1  X1  y  X2  W2   1. AND(x1, x2) 2. AND(x0, x1, x2) 3. AND(1, x1, x2) |

정답은 1번 🡪 기울기는 -1, 절편은 0.7/0.5 인 직선

3. AND게이트 뉴론 시각화





퀴즈)

|  |
| --- |
| OR게이트 구현 실습 |
| * (w1, w2) = **(0.5, 0.5), b=-0.7**   Y = 0 if (b + w1\*x1 + w2\*x2 <= )  1 if (b+ w1\*x1 + w2\*x2 > )  위 식을 만족하는 아래의 가중치, 편향의 조합중에서 AND게이트를 수행하는 조합을 찾으시요?   1. (w1,w2) = (0.5, 0.5),b=-0.7 2. (w1,w2) = (0.5, 0.5),b=-0.3 3. (w1,w2) = (0.5, 0.5),b=0.2 |

정답은 2번 🡪 기울기는 -1인 직선, 절편은 0.3/0.5

**2-2강. 미분**

* 학습목표

1. 미분 계수 이해
2. 미분법 이해
3. 미분을 통한 최대, 최소 구하는 법 이해
4. 미분

-

|  |
| --- |
| 수학에서 기울기의 정의 |
| * 수학에서 기울기는 변화율   - 변화율을 알아야 하는 이유: 세상은 지속적으로 변화함. 예측 가능케해줌.  - 날씨의 변화율을 알게되면 우산을 가지고 갈 필요가 없음.  \* 직선에서의 기울기  - f(x) =-2x +5 ; x의 계수인 -2가 기울기  A(x1, y1), A’(x2, y2) 두점의 차이가 기울기 (y2-y1) / (x2-x1) |
| * 곡선을 나타내는 방정식   + - X의 계수는 기울기가 아님.     - 두점의 좌표를 알아도 모든 변화율을 알수가 없음.     - 하지만 순간 변화율을 구하는 것은 가능, 구간 평균 변화율도 구할 수 있음(두점 사이에서.. ) |

1. 여러가지 함수의 미분법

|  |
| --- |
| 기본적인 미분법 |
| f(x) = C이면 F’(x) =0  f(x) = cg(x)이면, f’(x) = cg’(x)  f(x) = g(x) \*+- t(x) 이면, g’(x) +- t’(x)  f(x) = g(x)t(x) 이면, g’(x)\* t(x) + g(x).t’(x)  f(x) =t(x) / g(x) 이면, f’(x) = ( t’(x).g(x)) – t(x). g(x) ) /g(x)  f(x) = Xn이면 f’(x) = n\* X(n-1)  f(x) =f(g(x))n이면, f’(x) =n(g(x))n-1 \* g’(x)  f(x) = t(x) / g(x), f’(x) =( t’(x)g(x) – t(x).gt(x) ) / g2(x)  f(x) = 1/x, f’(x) = -1 |

|  |
| --- |
| 삼각함수 미분법 |
| f(x) = sin(x)이면, f’(x) = cos (x)  f(x) = cos (x)이면, f’(x) = -sin (x)  f(x) = tan (x)이면, f’(x) = -sin (x)  f(x) =t(x) / g(x) 이면, f’(x) = 1 / (1/cos(x))2 = sec2(x) |

|  |
| --- |
| 지수함수의 미분법 |
| 1. a > 0, a 1 인 ax 2. f(x) = ax이면, f’(x) = axln a 3. f(x) = ex이면, f’(x) = ex |

|  |
| --- |
| 합성함수의 미분법 |
| 1. t=f(x), u=g(x)일떄, f(x), g(x)가 모두 미분 가능하다면 2. f(g(x))’ = f’(g(x)) \* g’(x)   f(g(x)) = 1/(1+e-x) 일때 , f(g(x))’= ?   1. f(x) = 1/x2, g(x) = 1+e -x 2. f’(x) = -1/ x2 3. g’(x) = -e -x 4. f’(g(x)) = 1/ (1 +ex)2 5. f(g(x))’ = f’(g(x)) \* g’(x)   = 1 / (1 + e-x)2 \* e -x |

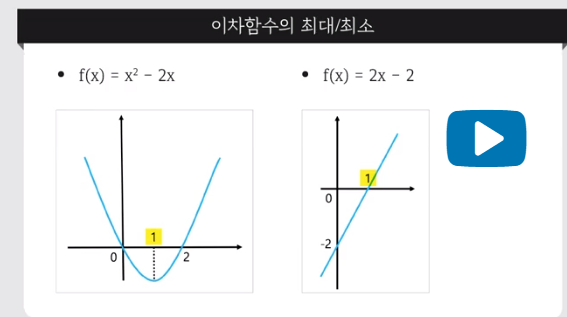
|  |
| --- |
| 함수가 여러 개의 변수를 갖는다면? |
| f(x) 🡪 f(x, y): 미분하고 싶은 변수만 미분하고 나머지는 상수 취급  f(x, y) = x2 +xy +y2   1. x에 대해 편미분 : fx(x, y) = 2x + y 2. y에 대해 편미분 : fy(x, y) = 2y + x |

1. 미분을 이용한 최대/최소 구하기

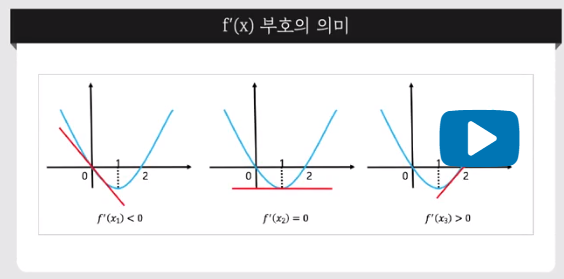
f(x) = X2 -2X

최대값, 최소값 확인위해서는 일일히 x값을 대입해서 알수 밖에 없음.

이를 피하기 위해서 미분을 사용함.



f’(x)=2x-2 식은 곡선에서의 기울기의 방향 기준이 됨.



f’(x) 가 < 0 데에서는 곡선에서 값이 점점 줄어든다는 의미이고

f’(x) =0 에서는 값의 변화가 없다는 의미이고 ( 최저값의 의미)

f’(x) >0 에서는 값의 변화가 점점 증가한다는 의미임.

<<참고>> 이차방정식 예쁘게 프린트

import sympy as sp

sp.init\_printing()

x, y, z=sp.symbols(‘x y z’)

z=x\*\*2 +y\*\*2

sp.pprint(z)

CTRL+Enter

z

CTRN+Enter

import sympy as sp

sp.init\_printing()

x, y=sp.symbols("x y ")

y=sp.integrate(x\*\*2, x)

sp.pprint(y)

## 정적분 표시

import sympy as sp

sp.init\_printing()

x, y=sp.symbols("x y")

a, b=sp.symbols("a b")

y=sp.integral(x\*\*2, (x, a, b)) ## x: 적분변수 0: 적분하한, 2: 적분상한

sp.pprint(y)

## 정적분 값 계산

import sympy as sp

sp.init\_printing()

x, y=sp.symbols("x y ")

y=sp.integrate(x\*\*2, (x, 0, 2)) ## x: 적분변수 0: 적분하한, 2: 적분상한

sp.pprint(y)

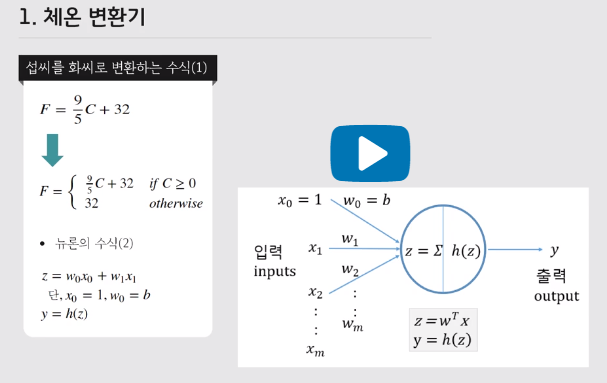
**2주차**

**2-3 활성화 함수 > 곡탄젠트**

#활성화 함수# 시그모이드#렐루#계단#쌍곡탄젠트

1. **활성화 함수**

섭씨 0도 이하에서는 화씨 32도를 리턴해주고, 0도 이상에서는 화씨로 변환해 주는 체온 변환기 구현(체온계가 영상일때만 작동하는 것을 구현)



y = h(z) // y 는 h of z 활성화 함수(activation function)

// 임계값은 섭씨 0도, 화씨 32도가 되어야 함

z = w T. x (입력 x 와 가중치 W의 곱의 총합은 z, z는 활성화 함수의 입력값)

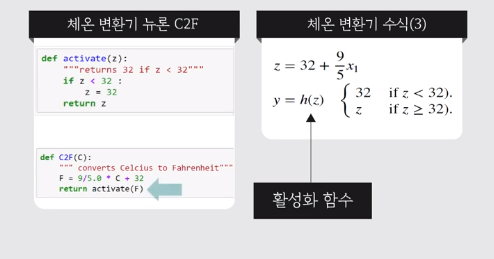
z= 32 +9/5 \* x1

y=h(z) --+-- if 입력 z < 32, 32를 출력

+-- z > 32, z를 출력, 활성화 함수의 최종 출력값은 🡪 y

1. 체온변환기 C2F뉴론 구현

|  |
| --- |
| 체온 변환기 뉴론 .C2F |
| def activate(z):  \*\* returns 32 if a < 32”  if z < 32 :  z = 32  return z |
| def C2F ( C ):  \*\* Convers Celsius to Fahrenheit \*\*  F = 9 / 5.0 \* C + 32  return activate(F) |



1. 체온 변환 C2F 뉴론 테스트

test\_c = [ -20, -10, 0, 36.5, 40, 50, 100 } ##테스트할 여러 온도 데이터 셋

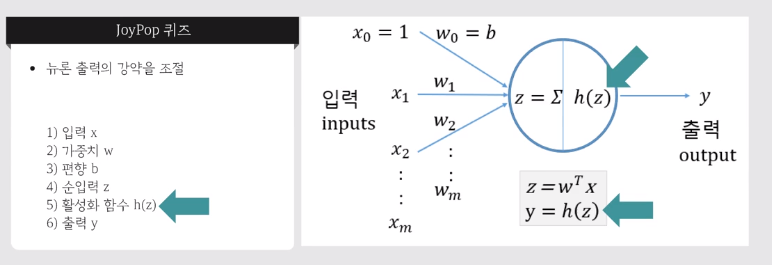
test\_f = [ C2F© for c in test\_c ]

prin(test\_f)

이런 테스트를 시각화한 코드

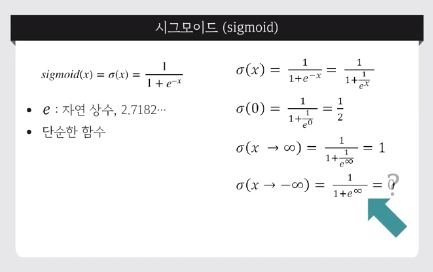
|  |
| --- |
| import matplotlib.pyplot as plt  import numpy as np  **%**matplotlib inline  #Plotting the simple neuron  x = np.arange( -100, 100, 0.1 )  y = [C2F(ix) for ix in x]  plt.figure()  plt plot(x, y)  plt.axis( [-20, 50, 0, 150 ] )  plt.xlabel(‘celcius’)  plt.ylabel(‘Fahreinheit’)  plt.show() |

1. 활성화 함수



* 활성화함수의 예

|  |
| --- |
|  |
|  |



* e : 자연상수 2.7182…
  + - x=0 : 🡪 함수값은 1/2

x = 100, 1000등 커지면 무한대🡪 함수값은 1

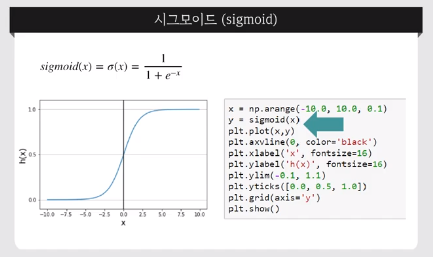
x = -100등 음의 무한대이면 🡪 함수값은 0

% x값과 관계없이 함수 출력 0 < y < +1 내에 있음

|  |
| --- |
| ##시그모이드 함수 구현  import numpy as np  def sigmoid(x):  return 1 / ( 1 + np.exp(-x))  x =np.arange(-10, 10, 0.1)  y = sigmoid(x)  plt.figure()  plt.plot(x, y)  plt.axvline(0, color = 'black')  plt.xlabel('x', fontsize=16)  plt.ylabel('Y', fontsize=16)  plt.ylim(-0.1, 1.1)  plt.yticks([0.0, 0.5, 1.0])  plt.grid(axis='y')  plt.show() |
|  |

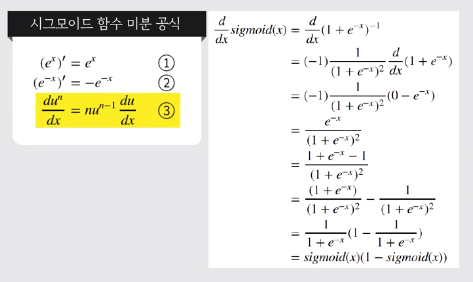
향후 logistic classfification에서 많이 사용할 예정

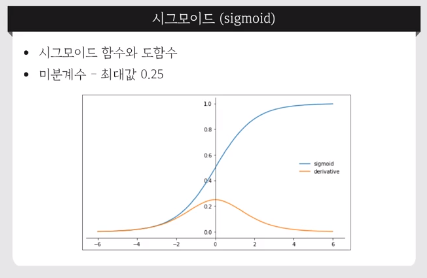
함수의 반환값 : 0 ~ 1사이이므로 확률 해석시 유용



텍스트이(가) 표시된 사진

자동 생성된 설명미분공식은 2-3개만 알면됨.





1. 활성화 함수 : 계단함수

|  |
| --- |
| 단극성 및 양극성 계단함수 |
| z=w0X0 +w1x1 + w2x2 + …  y = h(z)  h(z) = | 1 if z>= 0 (단극성)  | 0 otherwize  h(z) = | +1 if z>= 0 (양극성)  | -1 otherwize  def step(x): ## 단극성 계단함수  if x >= 0:  return 1  else:  return 0  print(‘step(3) = ‘. step(3)  step(3) = 1  z = step(np.array([-1, 2,3 ]))  printf(‘step[-1, 2,3]) = ‘, z) |
| 넘파이의 불린 인덱시 이용  x = np.array([-1, 2,3 ])  print(x > 0) ## 배열의 로직  🡪 [False True True] |

**2-4강 퍼셉트론**

**목표:** 퍼셉트론 이론과 퍼셉트론을 이용한 분류기(classifier)

1. 퍼셉트론의 발명

* 인공뉴론 – 뉴론, 노드, 퍼셉트론
* 퍼셉트론

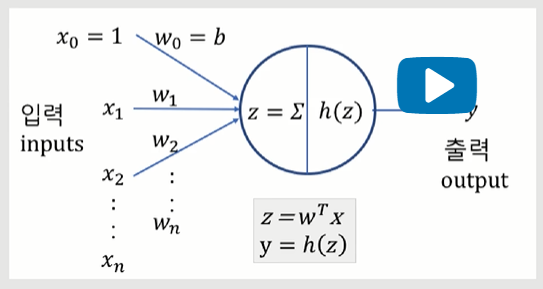
– 최초의 신경신경망

- 제안자 : 플랑크 로젠블라드

(소속: 1957년, 코넬 항공 연구소. 심리학전공)

- 하나의 뉴론이 정보를 받은후에 그 정보를 다음 뉴런으로 전달할지 아닌지 결정하기 위해서, 자기가 받은 입력특성에 가중치를 곱하는데, 기계가 최적의 가중치를 자동적으로 학습 (Mark I. 퍼셉트론 개발)

1. 퍼셉트론의 구조



* **퍼셉트론 알로리즘**

step 1) 각 입력 값과 각 가중치를 곱하여 합한다. WT. X, xj\* yj

step 2) 그 합한 값을 활성화 함수에 보냄.

step 3) 활성화 함수에서 그 값이 임계 값 보다 크면 1을 출력, 작으면 -1을 출력함 y=h(z)

step 4) 이 출력 값을 우리가 원하는 값과 비교해서 오차가 있으면 이를 반영하여 가중치를 변경. (기존 가중치 + delta 가중치)

* 입력값
  + - 입력 노드라고 부르며 작업해야 할 모든 학습자료가 이 변수에 담김.
    - 기계학습에서 입력은 n차원의 열벡터로 표기
    - 입력 x = | x0 |

| x1 |

| x2 |

| : |

| xn |

계산이 쉽도록 하기 위해 , 편향 x0=1, W0=b

* + - 학습자료 🡪 자료의 특성 == 벡터의 크기
* 가중치

- 처음에는 임의의 작은 수로 시작하지만, 학습을 진행하면서 결정되어짐.

- 가중치 벡터의 크기 = 항상 입력 벡터의 크기

* 편향
  + - 계산의 편의를 위해, 가중치 벡터의 첫 원소 W0를 편향으로 사용함 (편향에 계산 쉽게 하기 위하여 X0 = 1로 잡음)
  + 순입력 z

(활성함수의 순입력)

z= w0x0 + w1x1 + …. wnxn = xj\* wj

z= WT. X (T: Transpose. 전치)

(W, X : 벡터나 행렬. W.X를 내적하면 스칼라값)

* + y=h(z)

1. 순입력 계산과정

|  |
| --- |
| 순입력 z 계산 예제 |
| 1. 입력 x =[ 0, 1, 2, 3] ## 열벡터 2. 가중치 w = [0부터 1사이의 작은값] 3. 순입력 z를 계산하십시요 |
| 순입력 Z  z= w0x0 + w1x1 + …. wnxn  xj\* wj  = WT. X (T: Transpose. 전치) |
| 바람직하지 않은 코드(예제 1)  x =np.array( [ 0,1,2,3] )  w = np.array( [ 0, 0.1, 0.2, 0.3 ] )  z = np.dot( x, w )  print(z)  🡺1.4 (자료가 많아지면 불합리) |
| 바람직한 코드 ?(예제 2) : random()사용 바람직하나 W를 전치하지 않음  import numpy as np  x = np.array(np.arange(4))  w = np.array(np. random.random(4))  z= np.dot(w, x) ##🡨 weight 가 전치를 하지 않음: 잘못됨  print(z)  🡪 1.32905 |
| (예제 3)바람직한 코드: random()사용 바람직하나 W를 전치  import numpy as np  x = np.array(np.arange(4))  w = np.array(np. random.random(4))  z= np.dot(w, x)  print(z)  🡪 1.47818 |
| 예제 4) 바람직한 코드 : 가중치 전치(Transpose)한 것  -------------------------------------------------------------------------------  import numpy as np  x = np.array(np.arange(4))  w = np.array(np. random.random(4)) 🡨random값이므로 값이 자꾸바뀌어 잘된 코딩인 줄 모름: 잘못 코딩  z= np.dot(w.T, x)  print(z)  🡪 1.47819.. |
| \* 똑 같은 난수를 사용하여 디버깅하기 용이하게 코딩   * + - * np.random.seed(0) 사용   ---------------------------------------------------------------------------  예제 3A)  import numpy as np  np.random.seed(0)  x = np.array(np.arange(4))  w = np.array(np. random.random(4))  z= np.dot(w., x)  print(z)  3.555365 |
| 예제 2A)  import numpy as np  np.random.seed(0) //똑 같은 난수 사용하여 디버깅 용이  A. 코딩  x = np.array(np.arange(4))  w = np.array(np. random.random(4))  z= np.dot(w.T, x)  print(z)  3.555365   * + - * 전치하나, 하지 않으나 결과 값이 동일       * 즉 모델링을 잘못함. W 도 , X도 열벡터를 사용했기떄문임 |
| 1. x = np.array(np.arange(4))   w = np.array(np. random.random(4))  z= np.dot(w.T, x)  print(‘x.shape={}, w.shape{{}, w.T.shape{}’.format(x.shape, w.shape, w.T.shape)}  print(z)  🡪 x.shape=(4,), w.shape=(4,), w.T.shape=(4,) . 즉 형상이 같음. 잘못된 코딩임을 확인.  x와 w가 열벡터(Column vector)인데 1차원 배열로 만들었기 떄문임.  x는 행벡터이어야함. 즉 n\* 1  x, w도 행벡터로 변환하여야 하고, 단지 W는 전치해야 함. |
| print(x)  print(x.T)  print(z)  print(‘x.shape={}, w.shape{{}, w.T.shape{} z.shape{}’.format(x.shape, w.shape, w.T.shape, z.shape)}  print(z)   * + - * x.shape=(4, 1), w.shape=(4, 1), w.T.shape=(1, 4), z(1,)       * [[0.5488[ [0.715] 0.602 0.544 3.555]]   [[]] 부호 두개를 없애는 명령어 .squeeze()   * + - * z= np.dot(w.T, x).squeeze()   3.55 |

1. **퍼셉트론 (선형)이진 분류**

- 입력을 두가지로 분류하는 기계 학습(Binary classification)

- 입력은 주어지고, 가중치는 임의로 정하면 되고 활성화 함수는 어떻게 할지 고민필요.

|  |
| --- |
| Binary classification(이진 분류) |
| 1. 선형 이진 분류기 (Linear Binary classification) 2. 2차원에서 직선으로 분류가능하다고 전체 3. 퍼센트론으로 이진분류기를 만들기 위해 필요한 요소   – 입력, 가중치, ?  ? 🡪 활성화 함수  - 활성화 함수  . 시그모이드 함수  . 계단함수  . 쌍곡탄젠트 함수  . 렐루(ReLU) 함수 |
| 이진분류기 이므로 계단함수를 사용하면 됨 |

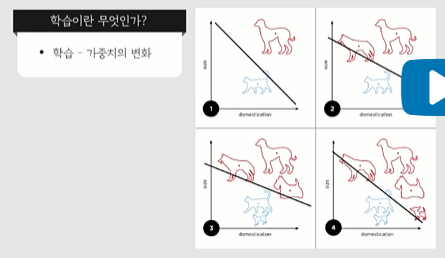
|  |
| --- |
| 선형이진 분류기의 활성화 함수 |
| 1. h(z) = | +1 (if z> 0)   | - 1 (otherwise)  계단함수(양극성)를 사용하면 됨. |
| y = h(z)  h(z) <0  h(z) = 0  |**---- (+1)**  h(z) >0  -------|-------  (-1)**----**|  계단함수(양극성)  선형 이진 분류기( z= wT.x) |

1. 퍼셉트론의 학습방법

- 이제 가중치를 임의로 정의하고 학습하면 된다.

- 입력이 들어오는 대로 분류하고, 잘못되면 가중치를 바꿔가며 개선하면 최적의 가중치를 얻게 됨.

|  |
| --- |
| 학습이란 무엇인가? |
| * **학습 – 가중치의 변화** |

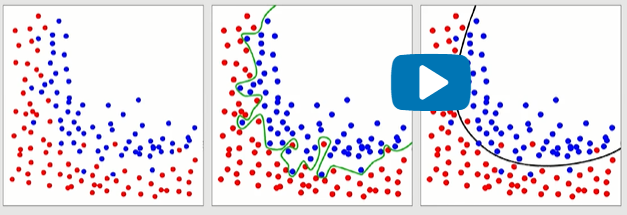


* 가중치 직선이 점점 개선됨
  + - 직선의 기울기와 절편은 모두 W에 포함됨
* 퍼셉트론 알고리즘
  + - 모든 학습자료가 모두 분류될떄 까지 학습 지속
    - 직선의 기울기와 절편은 모두 W에 포함되어 있음.
    - 가중치 조정이 끝나면, 새로운 자료를 받아 잘 평가 할 수 있는 분류해본다.
    - 새로운 데이터를 잘 분류하지 못하면, 가중치를 조정 설정해 주어야 한다.

1. **과대적합과 과소적합**

|  |
| --- |
| 과대적합, 과소적합은 언제 일어나는가? |
| * 과대적합(overfitting) – 학습자료 자체만 완벽하게 분류하려고 할떄 * 과소적합(underfitting) – 오차를 너무 허용할떄 |

경계선상에 데이터가 많이 들어오면 잘 분류를 하지 못함.



더 나은 분류를 하는 선은 ?

* 초록색

실제 문제에서는 차후에 더 나은 분류를 할 수 있는 것은 🡪 검정색

**2-5강 퍼셉트론 알고리즘**

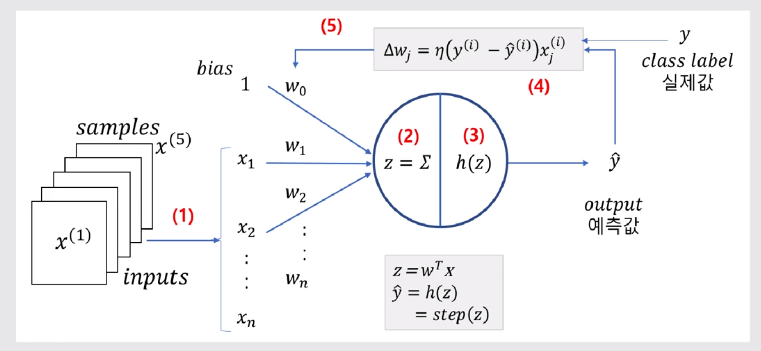
1. 퍼셉트론 알고리즘

* 퍼셉트론 이론
  + - 입력값 x들을 분류해 낼 수 있는 가중치 W를 구하는 것.

1. 퍼셉트론 요약

|  |  |  |
| --- | --- | --- |
| 목적 | | |
| * 입력값 x들을 분류해 낼 수 있는 가중치 W를 구하는 것. | | |
| 알고리즘 요약 | | 표기법 |
| * 가중치를 작은 난수(0~1)로 초기화 * 각 학습자료 X(i)에 대해   -출력 : 계산, =h(WT \* X)  -가중치 Wj, 조정, Wj := Wj +Wj | * X(i) : (i)번째 입력된 학습자료 * Xj(i) : (i)번째 입력된 학습자료의 j번째 특성 * : 퍼셉트론의 출력(읽기 : yhat) , 퍼셉트론의 예측값 * y : 클래스 레이블(퍼셉트론이 입력하길 원하는 실제값) * Wj : j번째 특성에 대한 가중치 * : 델타(미세한) j번 째 특성에 대한 미세한 가중치 조정값 | |
| 가중치 계산법 | | |
| 델타Wj : 가중치 미세한 조정값  입력 특성이 2개인 경우임(편향값 포함): 델라W0, 델타W1, 델타W2  3개의 원소로 구성되어 있음.   * **(1)번식이 퍼셉트론의 핵심**     case 1: y= y hat 🡪 델타Wj = 0, 즉 잘 예측한 경우임. 가중치는 변화없음  case 2: y =1, y hat = -1 🡪 델타Wj = n\*2\* Xj >0, 즉 가중치는 증가  y =-1, y hat = 1 🡪 델타Wj = n\*(-2)\* Xj <0, 즉 가중치는 감소  y =-1, y hat = -1 🡪 델타Wj = n\*0\* Xj = 0, 즉 가중치는 변화 없음  y = 1, y hat = 1 🡪 델타Wj = n\*(-0)\* Xj = 0, 즉 가중치는 변화 없음 | | |

1. **퍼셉트론 학습 전체 과정: 알고리즘 도식화**



(j: j회차 입력 i: i번째 특성)

1. **입력단계 :** n개의 특성
2. **순 입력 계산단계 :** x1, x2, … Xn을 가지고 W1, w2.. Wn 과 곱해서 합산한 순 입력 z를 구함 (**z** =(WT(i)\*X(i) )
3. **예측값 구하는 단계:** 순 입력 z를 활성화 함수에 넣어서 예측 값 y hat 퍼셉트론(활성화 함수 결과) 출력을 구한다. (= h(z), h(z): 활성함수)
4. **예측 값과 출력 값을 비교.** 🡪 y와 y hat을 비교. 델타w **()**를 계산
5. **가중치 조정단계:** 기존 가중치 w에 델타W를 더해서 델타만큼 조정

**W(j) = w(j-1) + (j-1)**

**(예제)**하나의 입력 (x1, x2) =(1,2) 에 대한 클래스 레이블과 예측값(결과)가 각각 1, -1 이면, 가중치 조정값 (Delta W1, Delta W2)은 무엇인가? 학습률 n(에타) =0.1

**Solve)** 0.1 \* (1-(-1)) (1, 2) = (0.2, 0.4) 🡨 Delta Wj = n(y(i), y hat(i)) \* xj(i)

1. **퍼셉트론 알고리즘의 한계**

-1957 로젠 블랏트 퍼셉트론 발표

-1969 MIT 마빈 민스키

. 퍼셉트론 한계 :XOR 풀이 불가

. 다층 퍼셉트론은 XOR풀이 가능 . 학습방법은 찾지 못함

- 1974 하버드 대학원생. 펄 워브스

. 다층 퍼셉트론을 학습할 수 있는 역전파 알고리즘 발표

**배타적 논리합(XOR)**

X2

XOR 진리표

**?**

(0,1)

X1

(1,1)

(0,0)

|  |  |  |
| --- | --- | --- |
| X1 | X2 | y |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

(1,0)

* 퍼셉트론은 (XOR풀이 문제) 이 분류 문제를 해결하지 못함. – 선형분리기의 한계 (1969. MIT마빈 민스키)
* 그러나 다층 퍼셉트론을 학습시킬수 있는 역전파 알고리즘을 사용하면 해결 가능. 🡪 이게 인공신경망.

(1974. 하버드 대학원생. 펄 워브스)

* 판별식 : Decision boundary

X2

(1,1)

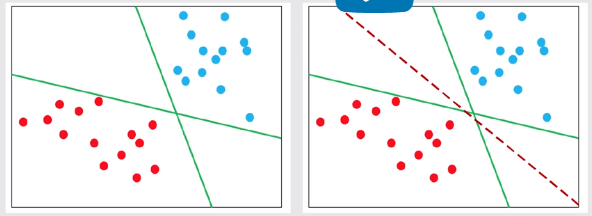
(0,1)

X1

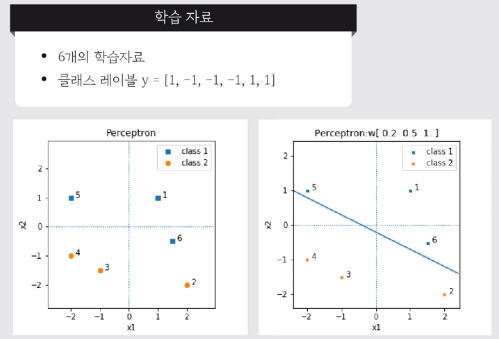
(0,0)

(1,0)

|  |
| --- |
| 최적 분류의 한계 |
| 직선으로 분류는 하지만 최적의 직선은 아님 |



1. **퍼셉트론의 예제**

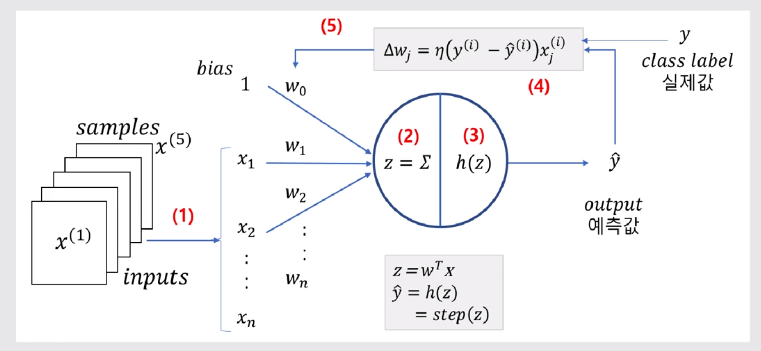


**(문제 정의)**

- 6개의 학습자료가 있고, 클래스 레이블 y, 주어진 자료를 학습시켜서, 학습 자료들을 분류할 수 있는 두 클래스로 분류할 수 있는 가중치 구하기

* 풀이

|  |
| --- |
| **순서** |
| * **step 1 : 가중치 w 계산하기**   o 초기 가중치는 임의로 정함 : WT =[ 0, 1, 0.5]  o 학습률 n=0.1  o 학습자료 : x(1) = [1,1], x(2) = [2,-2], x(3) = [-1, -1.5]  x(4) = [-2, -1], x(5) =[-2.0, 1.0], x(6) = [1.5, -0.5] |



WT \* x

테이블이(가) 표시된 사진

자동 생성된 설명

테이블이(가) 표시된 사진

자동 생성된 설명

WT \* x

delta Wi = n(기대결과Y(i) – 활성함수결과(Yhat (i))\* Xj(i)

테이블이(가) 표시된 사진

자동 생성된 설명

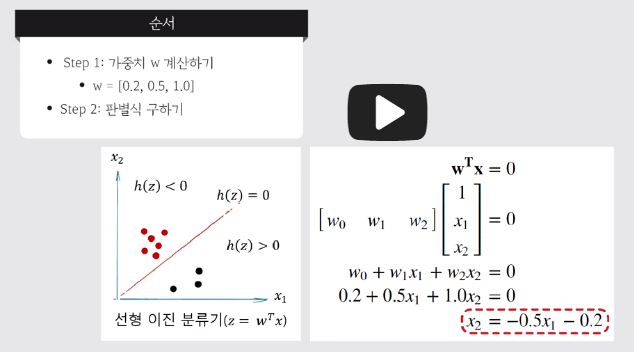
Wj(i)= Wj(i-1) + delta Wj(i-1) (j: j회차 입력 i: i번째 특성)

테이블이(가) 표시된 사진

자동 생성된 설명

가장 중요한 것은 final 가중치

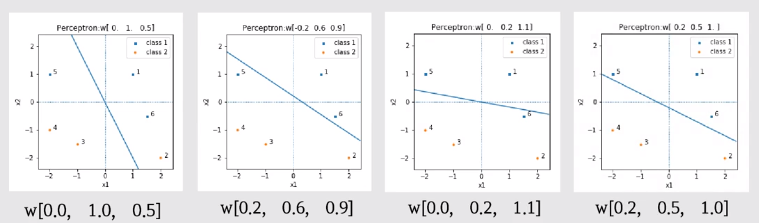
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  | W(i) = W(i-1) +delta W(i-1) | | | |  | 활성.함결과 | |  |  | 기대결과 |  | Rate율 | Del W(i) = n( y(i) -y hat (i) )\* X(i) | | | |
| i | x0(i) | x1(i) | | x2(i) |  | W0 | W1 | W2 |  | sumW.T \* x |  | y hat(i) |  | y(i) |  | n |  | delta W0 | delta W2 | delta W2 |  |
| 1 | 1 | 1 | | 1 |  | 0 | 1 | 0.5 |  | 1.5 |  | 1 |  | 1 |  | 0.1 |  | 0 | 0 | 0 |  |
| 2 | 1 | 2 | | -2 |  | 0 | 1 | 0.5 |  | 1 |  | 1 |  | -1 |  | 0.1 |  | -0.2 | -0.4 | 0.4 |  |
| 3 | 1 | -1 | | -1.5 |  | -0.2 | 0.6 | 0.9 |  | -2.15 |  | -1 |  | -1 |  | 0.1 |  | 0 | 0 | 0 |  |
| 4 | 1 | -2 | | -1 |  | -0.2 | 0.6 | 0.9 |  | -2.3 |  | -1 |  | -1 |  | 0.1 |  | 0 | 0 | 0 |  |
| 5 | 1 | -2 | | 1 |  | -0.2 | 0.6 | 0.9 |  | -0.5 |  | -1 |  | 1 |  | 0.1 |  | 0.2 | -0.4 | 0.2 |  |
| 6 | 1 | 1.5 | | -0.5 |  | 0 | 0.2 | 1.1 |  | -0.25 |  | -1 |  | 1 |  | 0.1 |  | 0.2 | 0.3 | -0.1 |  |
| final |  |  | |  |  | 0.2 | 0.5 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | | |  | | | | | | | | | | | |

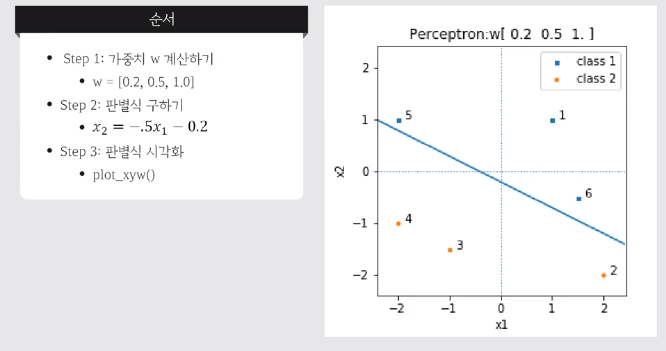


x2에 관하여 풀면 y -ax +b와 같은 직선의 식이 나옴.

예제)

|  |
| --- |
| %%writefile  plot\_xyw.py ##셀의 첫줄에 작성시 파일로 저장  ## %%load code/plt\_xyw.py ## 셀의 첫줄에 작성이 파일에 저장된 것 읽어올수 있음  import matplotlib.pyplot as plt  import numpy as np  %matplotlib inline  %run code/plot\_xyw.py ##run 후 함수 호출  x= np.array( [1.0, 1.0], [2.0, -2.0], [-1.0,-1.5],  [-2.0, -1.0], [-2.0, 1.0], [1.5, -0.5])  X = np.c\_[np.ones(len(x), x)]  y= np.array([1, -1, -1, -1, 1, 1])  w=np.array([0.2, 0.5, 1.0])  plot\_xyw(X, y, w, X0=True, annotate=True)  test\_c = [-20, -10, 0, 32, 36, 40, 50, 400]  test\_f = [C2F(c) for c in test\_c]  print(test\_f) |





**2-6강 퍼셉트론 코딩**

1. **퍼셉트론 예제**

|  |
| --- |
| **조 건** |
| o 초기 가중치는 임의로 정함 : WT =[ 0, 1, 0.5]  o 학습률 n=0.1  o 학습자료 : x(1) = [1,1], x(2) =[2,-2] x(3) = [-1, -1.5] x(4) = [-2, -1] x(5)=[-2.0, 1.0] x(6)=[1.5, -0.5]  o 클래스 레이블 y:  y =[1, -1, -1, -1, 1, 1] |

|  |
| --- |
| Step 1 x 입력 자료 준비 |
| import numpy as np  x= np.array( [1.0, 1.0], [2.0, -2.0], [-1.0,-1.5],  [-2.0, -1.0], [-2.0, 1.0], [1.5, -0.5])  🡪 편향(bias)가 추가되어 있지 않으므로 추가해서 아래 구조로 바꾸어야 함.  [1, 1.0, 1.0], [1, 2.0, -2.0], [1, -1.0,-1.5],  [1, -2.0, -1.0], [1, -2.0, 1.0], [1, 1.5, -0.5])  \* X = np.c\_[np.ones(len(x), x)] 🡨사용 c\_ 함수는 두배열을 순서대로 합치는 방법 |

|  |
| --- |
| Step 2 클래스레이블 y 자료 준비 |
| w= np.random.random(( X.shape[1], 1))  cf. x.shape= (6, 2)  X.shape= (6, 3)  w=np.array( [ 0.0, 1.0, 0.5 ])  y=np.array([1, -1, -1, -1, 1, 1])  🡪 편향(bias)가 추가되어 있지 않으므로 추가해서 아래 구조로 바꾸어야 함.  [1, 1.0, 1.0], [1, 2.0, -2.0], [1, -1.0,-1.5],  [1, -2.0, -1.0], [1, -2.0, 1.0], [1, 1.5, -0.5])  \* X = np.c\_[np.ones(len(x), x)] 🡨사용 c\_ 함수는 두배열을 순서대로 합치는 방법  - len()함수는 형상의 첫번쨰 값 반환  - x는 학습자료  - y는 클래스 레이블 |

1. 학습자료 연산하기

|  |
| --- |
| 알고리즘 |
| * 각 학습자료에 대해 z =wT.x 계산   z=np.dot(w.T, x)   * 활성화 함수(계단함수)에 z 적용   yhat = h(z)   * 가중치 조정값 계산과 가중치 조정   - Wj = n(y(i) – (i) )xj(i)  -  Wj = Wj +Wj   * 모든 학습자료에 대해 실행 |
| eta = 0.1 ##학습율 초기화  for xi, yi in zip(X, y): **# zip 을 사용하여 반복할 수 있는 이터러블 리스트**  xi = xi.reshape(w.shape) # **열벡터 🡪 행벡터로 변환**  z= np.dot(x.T, xi)  yhat = np.where(z > 0.0, 1, -1) #계단함수 사용하지 않고 if z>0, 1 else -1    delta = eta \* (yi -yhat) \*xi # delta 값 구하기  w += delta # 가중치를 조정  print(np.round(w,2)) # 결과를 소수 2째자리까지 출력 |
|  |