**[ 지도 학습 및 비지도 학습 ]**

머신러닝은 크게 지도 학습과 비지도 학습으로 구분

**지도 학습 ( supervised learning )**

* 레이블(label)이 달린 데이터로 학습 모델을 만든다
  + 나이와 학력, 거주지와 같은 다양한 매개변수를 기반으로 개인의 소득을 예측하는 시스템을 구축하려면, 먼저 여러 사람에 대한 구체적인 정보를 모은 후 항목마다 레이블(각 사람의 소득 정보)을 달아서 데이터베이스를 구축
* 여러가지 매개변수와 소득의 관계가 정의된 데이터베이스를 학습 알고리즘에게 전달하면, 특정한 사람에 대한 매개변수가 주어졌을 때 그 사람의 소득을 계산하는 방법을 학습

**비지도 학습 ( unsupervised learning )**

* 레이블이 달리지 않은 데이터로 학습 모델을 만든다.
* 레이블이 없기 때문에 데이터의 내용만 보고 의미 있는 정보를 추출해야 한다.
  + 특정 데이터 집합에 속한 개별 데이터들을 여러 개의 그룹으로 나눌 때 비지도 학습 방식을 적용할 수 있다.
* 그룹을 나누는 기준이 명확하지 않기 때문에 가능한 최선의 방법을 동원해 데이터를 나눠야 한다.

**[ 분류 ] – 지도학습**

분류 ( classification ) : 데이터를 지정한 수만큼의 클래스(범주)로 나누는 기법  
데이터를 가장 효과적이면서 효율적으로 활용하도록 일정한 수의 그룹으로 데이터를 분류한다.

머신 러닝에서는 주어진 data 항목이 속하는 클래스를 결정하는 문제를 다룰 때 분류 기법 사용

* Ex ) 주어진 이미지에 사람 얼굴이 있는지 검사하고 싶다면, 먼저 얼굴 이미지와 얼굴이 아닌 이미지라는 두가지 클래스로 구분된 학습 데이터를 구축한 후 , 이 데이터로 모델을 학습시킨다. 이 후 , 학습이 완료된 모델로 추론(이미지 검사) 한다.

분류를 위해 샘플(학습용 데이터)의 수는 다양한 상황을 반영할 수 있도록 충분해야 함.  
샘플이 부족하면 알고리즘이 학습 데이터에 필요 이상으로 최적화되는 오버피팅(overfitting ) 현상이 발생할 수 있다.

**[ 데이터 전처리 ]**

머신 러닝을 구현 할 때 현실 세계에서 추출한 방대한 양의 미가공 데이터를 다룬다.  
따라서, 학습 시키기 전에 먼저 데이터의 포맷을 머신 러닝 알고리즘의 처리할 수 있는 형태로 변환하는 전처리(preprocessing) 작업부터 해야한다.

|  |
| --- |
| Import numpy as np  From sklearn import preprocessing  Input\_data = np.array([[5.1,-2.9,3.3],  [-1.2,7.8,-6.1],  [3.9,0.4,2.1],  [7.3,-9.9,-4.5]]) |

전처리 기법 4가지

1. 이진화
2. 평균 제거
3. 크기 조정
4. 정규화

**[ 이진화 ]**

이진화 ( binariaztion ) 란 숫자를 불리언 ( Boolean ) 수로 변환하는 기법 ( 이진수로 변환 )

사이킷런에서 제공하는 메소드로 입력 데이터를 이진화하는 방법  
기준점 2.1

|  |
| --- |
| Data\_binarized = preprocessing.Binarizer(threshold=2.1).transform(input\_data)  Print(“\nBinarized data:\n”, data\_binarized) |

결과를 보면 2.1 보다 큰 값을 1로, 나머지는 0으로 변환됨을 볼 수 있음

**[ 평균 제거 ]**

머신 러닝에서 흔히 사용하는 전처리 기법, 특히 **특징 벡터( feature vector ) 의 값들이 0을 중심으로 분포하게 만들 때** 평균 제거 기법을 많이 사용  
특징 벡터 : 물체의 특징점들을 나열한 것.

*앞에서 작성코드에 이어서 작성*

|  |
| --- |
| #평균과 표준편차를 출력  Print(“\nbefore: “)  Print(“Mean =”, input\_data.mean(axis=0))  Print(“std deviation =”, input\_data.std(axis=0))  #평균 제거  Data\_scaled = preprocessing.scale(input\_data)  Print(“\nAfter: “)  Print(“Mean =”, input\_data.mean(axis=0))  Print(“std deviation =”, input\_data.std(axis=0)) |

결과를 보면 평균 값은 0에 표준편차는 1에 가깝게 조정된 것을 알 수 있다

**[ 크기 조정 ]**

특징 벡터의 각 요소에 대한 값의 범위는 일정하지 않을 수 있다.  
머신 러닝 알고리즘을 학습 시키려면, 모든 특징을 동일 선상에서 비교할 수 있도록 각각의 특징에 대한 값의 범위를 일정한 기준으로 조정하는 크기 조정(scaling) 과정을 거쳐야 한다.

|  |
| --- |
| #최솟값/최댓값 조정  Data\_scaler\_minmax = preprocessing.MinMaxScaler(feature\_range=(0,1))  Data\_scaled\_minmax = data\_scaler\_minmax.fit\_transform(input\_data)  Print(“\nMin Max Scaled data: \n”, data\_scaled\_minmax) |

각 행마다 최댓값은 1로, 나머지는 1을 기준으로 상대적인 값으로 표현됨을 확인

**[ 정규화 ]**

특징 벡터의 값을 일정한 기준으로 측정하려면 정규화( normalization ) 과정을 거친다.  
정규화 기법

1. L1 정규화 ( 최소 절대 편차 Least Absolute Deviation ) : 각 행의 절대 값의 합이 1이 되도록 조정
2. L2 정규화 ( 최소 제곱 Least Square ) 제곱의 합이 1이 되도록 조정

L2보다 L1 기법이 더 안정적!! L1이 데이터의 이상치(Outlier) 에 영향을 덜 받기 때문에.  
반면 이상치가 문제 해결에 중요한 역할을 한다면 L2 정규화를 사용해야함

*Outlier : 잘못 평가된 값, 결과적으로 잘못된 분석 결과를 초래 할 수 있는 값들. 간단하게 말해서 샘플의 전체적인 패턴에서 벗어나게 관측되는 값*

|  |
| --- |
| #데이터 정규화  Data\_normalized\_l1 = preprocessing.normalize(input\_data, norm=’l1’)  Data\_normalized\_l2 = preprocessing.normalize(input\_data, norm=’l2’)  Print(“\n L1 normalized data: \n” , data\_normalized\_l1)  Print(“\n L2 normalized data: \n” , data\_normalized\_l2) |

**[ 레이블 인코딩 ]**

분류 작업 시, 수많은 레이블을 처리한다. 레이블은 문자나 숫자 뿐만 아니라 다양한 형태로 표현  
사이킷런에서 제공하는 머신 러닝 함수는 숫자로 된 레이블만 처리.

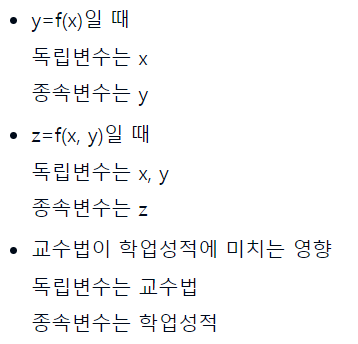
레이블은 대체로 사람이 읽기 좋도록 문자로 표현하는 경우가 많다. 문자로 된 레이블을 숫자로 변환하려면 레이블 인코더를 거쳐야 한다.

레이블 인코딩 ( label encoding ) 이란 문자로 표현된 레이블을 숫자 형태로 바꾸는 과정

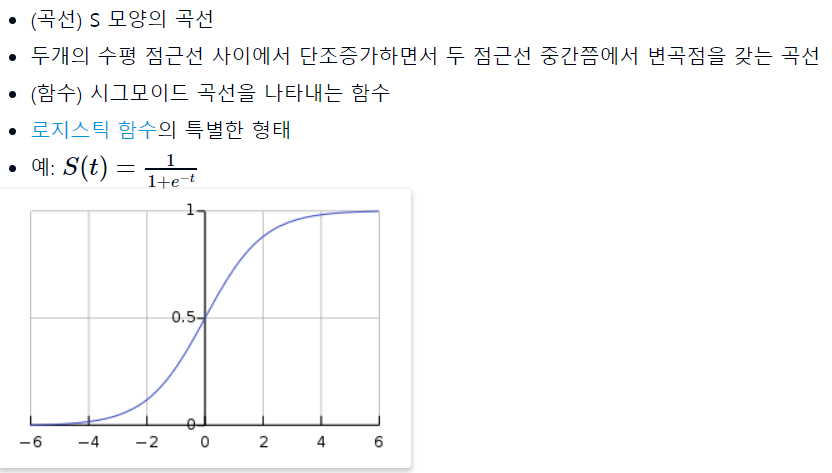
|  |
| --- |
| Import numpy as np  From sklearn import preprocessing  #샘플 입력 레이블  Input\_labels = [ ‘red’ , ‘black’ , ‘red’, ‘green’, ‘black’, ‘yellow’, ‘white’ ]  #레이블 인코더 생성 후 앞에서 정의한 레이블로 학습시키기  Encoder = preprocessing.LabelEncoder()  Encoder.fit(input\_labels)  #매핑 관계 출력  Print(\n Label mapping:\n”)  For I, item in enumerate(encoder.classes\_) :  Print(item, ‘🡪’ , i)  #레이블 인코딩  Test\_labels = [‘green’ , ‘red’, ‘black’ ]  Encoded\_values = encoder.transform(test\_labels)  Print(“\n Lables = “ , test\_labels)  Print(“\n Encoded values= “ , list(encoded\_values))  #숫자 값 디코딩  encoded\_values = [3,0,4,1]  decoded\_list = encoder.inverse\_transform(encoded\_values)  Print(“\n Encoded values = “ , encoded\_values)  Print(“\n decoded values= “ , list(decoded\_list)) |

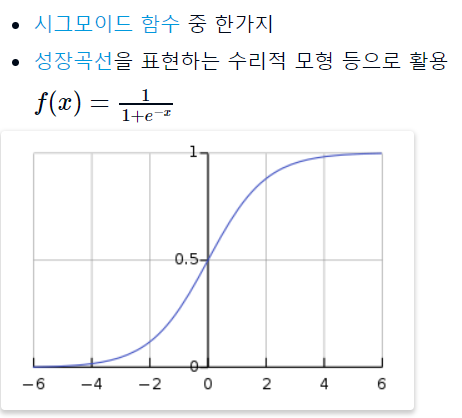
**[ 로지스틱 회귀 분류기 ]**

로지스틱 회귀 분석 ( logistic regression ) 이란 입력 변수와 출력 변수의 관계를 표현하는 기법 중 하나다. 여기서 입력은 독립 변수고 출력은 종속 변수다. 종속 변수의 수는 고정된다. 분류 문제를 다룰 때는 종속 변수로 분류할 클래스(그룹)를 표현한다.



**로지스틱 회귀 분석은** **독립 변수와 종속 변수의 관계를 로지스틱 함수를 통해 계산된 확률로 표현한다**. 이때 로지스틱 함수는 시그모이드 곡선( sigmoid curve) 으로 표현한다.

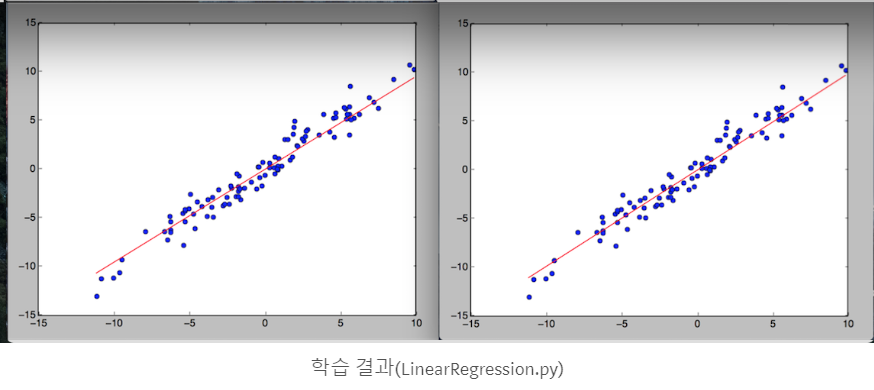
**시그모이드 곡선**

**로지스틱 함수**

여러가지 **매개변수로 구성된 함수를 만들 때 주로 사용**

로지스틱 함수는 데이터의 분포를 표현하는 직선 중에 오차가 가장 적은 직선을 구하는 일반 선형 모델 ( generalized linear model 선형 회귀 분석 ) 과 밀접한 관계가 있다.





엄밀히 말하면, 로지스틱 회귀 분석이 분류 기법은 아니지만, 분류 문제를 다루는 데 효과적이고 간결하기 때문에 머신러닝에서 굉장히 많이 사용한다.

**[ ]**

**[ ]**

**[ ]**

**[ ]**

**[ ]**