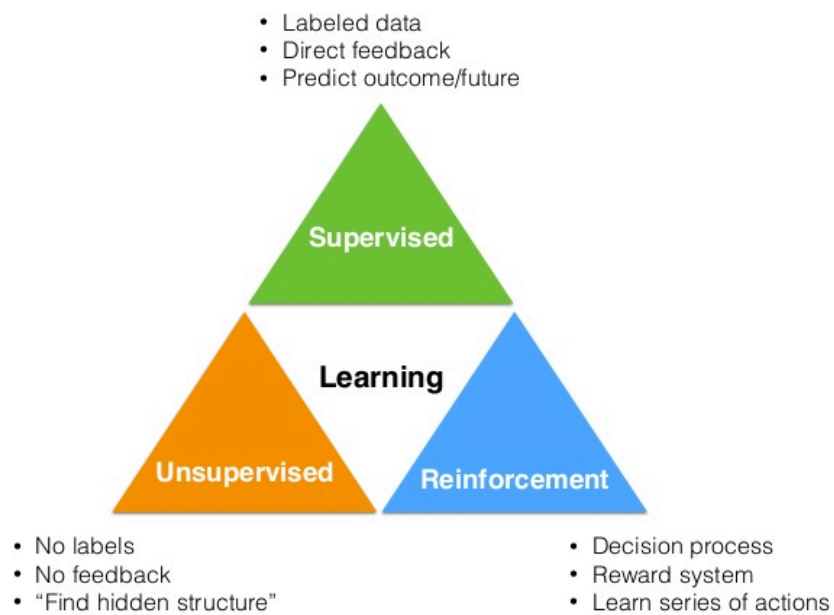


6. 머신러닝과 딥러닝 이해하기 - I

머신러닝 개요

기계 학습 또는 머신 러닝은 인공 지능의 한 분야로, 컴퓨터가 학습할 수 있도록 하는 알고리즘과 기술을 개발하는 분야를 말한다. 가령, 기계 학습을 통해서 수신한 이메일이 스팸인지 아닌지를 구분할 수 있도록 훈련할 수 있다. 기계 학습의 핵심은 표현과 일반화에 있다. 표현이란 데이터의 평가이며, 일반화란 아직 알 수 없는 데이터에 대한 처리이다. 이는 전산 학습 이론 분야이기도 하다. 다양한 기계 학습의 응용이 존재한다. (출처: 위키백과)

머신러닝 방법(알고리즘)



지도학습	Classification (분류)	kNN
		Naïve Bayes
		Support Vector Machine (SVM)
		Decision Tree
	Regression (회귀 or 예측)	Linear regression
		Locally weighted linear regression
		Ridge
		Lasso
비지도학습		Clustering
		K means
		Density estimation
		Expectation maximization
		Pazen window
		DBSCAN
강화학습		Q-learning

<표 출처: <http://chapter5k.blogspot.kr/2016/01/supervised-learning-unsupervised.html>>

● 지도 학습(Supervised learning)

레이블(label)이 달린 데이터로 학습 모델을 만든다. 예를 들어 나이와 학력, 거주지와 같은 다양한 매개변수(parameter)를 기반으로 개인의 소득을 예측하는 시스템을 구축하려면, 먼저 여러 사람에 대한 구체적인 정보를 모은 후 항목마다 레이블(각 사람의 소득 정보)을 달아서 데이터베이스를 구축한다. 이렇게 여러 가지 매개변수와 소득의 관계가 정의된 데이터베이스를 학습 알고리즘에게 전달하면, 특정한 사람에 대한 매개변수가 주어졌을 때 그 사람의 소득을 계산하는 방법을 학습한다.

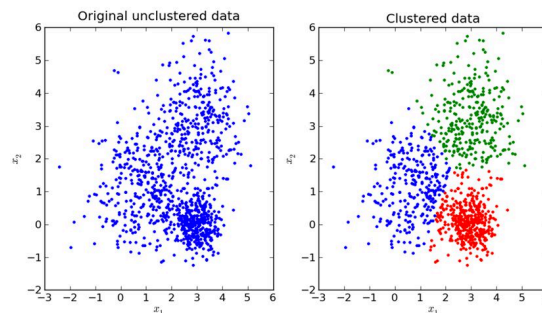


<MINIST 데이터셋>

● 비지도 학습(Unsupervised learning)

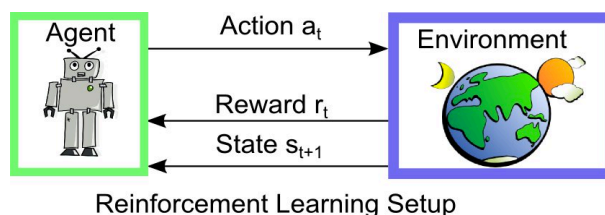
레이블이 달리지 않은 데이터로 학습 모델을 만든다. 데이터에 레이블이 달려 있지 않기 때문에, 데이터의 내용만 보고 의미 있는 정보를 추출해야 한다. 예를 들어 특정 데이터 집합에 속한 개별 데이터들을 여러 개의 그룹으로 나눌 때 비지도 학습 방식을 적용할 수 있다. 이 때 가장 어려운 부분은 그룹을 나누는 기준이 명확하지 않다는 점이다. 비지도 학습 방식의 알고리즘은 가능한 최선의 방법을 동원해 데이터를 나눠야 한다.

Unsupervised Learning



● 강화 학습(Reinforcement learning)

입력값(액션)에 대한 평가만 주어짐. 보상이 주어짐.



Reinforcement Learning Setup

참고링크 : <http://solarisailab.com/archives/1785>

1. 지도학습

1) Classification (분류)

미리 정의된 여러 클래스 레이블 중 하나를 예측하는 것

분류 기법은 데이터를 가장 효과적이면서 효율적으로 활용하도록 일정한 수의 그룹으로 데이터를 분류

머신 러닝에서는 주어진 데이터 항목이 속하는 클래스를 결정하는 문제를 다룰 때 분류 기법을 활용

얼굴 인식, 스팸 필터, 추천 엔진 등에 활용

이진분류와 다중분류

이진분류(Binary Classification) :

- 딱 두 개의 클래스로 분류 (예, 아니오)
- ex. 스팸 메일인가 아닌가? 비만인가 아닌가? 당뇨병인가 아닌가?

다중분류(Multiclass Classification) :

- 세 개 이상의 클래스로 분류
- ex. 뉴스기사의 주제분야, 언어의 종류, 붓꽃의 품종, 강아지 품종 등

오버피팅(Overfitting)

샘플 데이터의 수가 부족하여 알고리즘이 학습 데이터에 필요 이상으로 최적화되는 오버피팅(과다적응, 과최적화) 현상이 발생할 수 있음

다시 말해, 분류 기준이 학습 데이터에 너무 치우쳐서 학습 데이터에 없는 다른 값에 대해서는 제대로 분류할 수 없음

(예를 들어, 얼굴인식 모델을 만들 때 동양인 얼굴로만 구성되면, 서양인 얼굴을 얼굴로 분류하지 않게 됨)

이는 머신러닝 과정에서 굉장히 자주 겪는 문제 중 하나이므로, 머신 러닝 모델을 만들 때 오버피팅의 발생 가능성을 항상 염두에 두어야 함

2) Regression (회귀 or 예측)

연속적인 숫자(실수, 부동소수점소수)를 예측하는 방법

ex. 어떤 사람의 교육수준, 나이, 주거지를 바탕으로 연간소득을 예측하는 것

ex. 옥수수 농장에서 전년도 수확량과 날씨, 고용 인원수 등으로 올해 수확량을 예측하는 것

출력 값에 연속성이 있는지를 질문해 보면 회귀와 분류 문제를 쉽게 구분할 수 있음 (예상 출력값 사이에 연속성 유무)

ex. A라는 사람의 연소득을 예측해야 하는 경우 40,000,000원을 예측해야 하는데 알고리즘이 39,999,999원을 예측했다 하더라도 큰 문제되지 않음

반대로, 강아지의 품종을 인식해야 하는데 ‘푸들’을 ‘불독’으로 예측하는 것은 큰 문제가 됨

* 데이터 전처리

데이터를 머신러닝 알고리즘이 처리할 수 있는 형태로 변환하는 작업

- **이진화(binartization)** : 숫자를 이진수(boolean)로 전환
- **평균 제거(mean removal)** : 특징 벡터(feature vector)의 값들이 0을 중심으로 분포(치우치지 않게)하도록 만들 때 사용
- **크기 조정(scaling)** : 특징 벡터 값의 범위를 일정 기준으로 조정. 어떤 특징이 비정상적으로 크거나 작지 않도록 조정
- **정규화(normalization)** : 특징 벡터 값을 일정한 기준으로 측정하려면 정규화 필요.
 - L1 정규화(최소 절대편차) : 각 행의 절대값의 합이 1이 되도록 조정
 - L2 정규화(최대 절대편차) : 각 행의 제곱의 합이 1이 되도록 조정

실습1. 데이터 전처리

#패키지 호출

```
import numpy as np
from sklearn import preprocessing
```

#샘플 데이터 정의

```
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]])
```

#데이터 이진화 (임계값(threshold)은 2.1로 지정. 즉, 임계값 2.1을 기준으로 1과 0을 구분)

```
data_binarized = preprocessing.Binarizer(threshold=2.1).transform(input_data)
print("\nBinarized data : \n", data_binarized)
```

#평균과 표준편차 출력

```
print("\n BEFORE : ")
print("Mean = ", input_data.mean(axis=0))
print("Std deviation = ", input_data.std(axis=0))
```

#평균 제거

```
data_scaed = preprocessing.scale(input_data)
print("\n AFTER : ")
print("Mean = ", data_scaed.mean(axis=0))
print("Std deviation = ", data_scaed.std(axis=0))
```

#크기 조정

#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)
```

#정규화

#데이터 정규화

L1 정규화 : 각 행의 합이 1이 되도록 조정 L2보다 안정적임

```
data_normalized_l1 = preprocessing.normalize(input_data, norm='l1')
```

L2 정규화 : 각 행의 제곱의 합이 1이 되도록 조정

```
data_normalized_l2 = preprocessing.normalize(input_data, norm='l2')
print("\nL1 normalized data : \n", data_normalized_l1)
print("\nL2 normalized data : \n", data_normalized_l2)
```

```

Binarized data : ※ 이진화
[[ 1.  0.  1.]
 [ 0.  1.  0.]
 [ 1.  0.  0.]
 [ 1.  0.  0.]]

BEFORE :
Mean = [ 3.775 -1.15 -1.3 ]
Std deviation = [ 3.12039661  6.36651396  4.0620192 ]

AFTER :
Mean = [ 1.11022302e-16  0.00000000e+00  2.77555756e-17]
Std deviation = [ 1.  1.  1.]

Min max scaled data :
[[ 0.74117647  0.39548023  1.
  0.
  1.
  0.6
  0.5819209  0.87234043
  1.
  0.
  0.17021277]] ※ 크기 조정

L1 normalized data :
[[ 0.45132743 -0.25663717  0.2920354 ]
 [-0.0794702  0.51655629 -0.40397351]
 [ 0.609375  0.0625  0.328125 ]
 [ 0.33640553 -0.4562212 -0.20737327]] ※ 정규화

L1 normalized data :
[[ 0.75765788 -0.43082507  0.49024922]
 [-0.12030718  0.78199664 -0.61156148]
 [ 0.87690281  0.08993875  0.47217844]
 [ 0.55734935 -0.75585734 -0.34357152]]
>>>

```

출력결과

실습2. 이미지 형태 인식

(Hash: 암호화된 정보)

이미지와 라이브러리 호출

```
from PIL import Image
import numpy as np
```

이미지 데이터를 Averages Hash로 변환 함수 선언

```
def average_hash(fname, size = 16): #average_hash(파일이름, 사이즈)
    img = Image.open(fname) # 이미지 데이터 열기
    img = img.convert('L') # 그레이스케일로 변환
    # '1' 지정하게 되면 이진화 그밖에 "RGB", "RGBA", "CMYAK" 모드 지정가능
    img = img.resize((size, size), Image.ANTIALIAS) # 이미지 리사이즈
    pixel_data = img.getdata() # 픽셀 데이터 가져오기
    pixels = np.array(pixel_data) # Numpy 배열로 변환하기

    pixels = pixels.reshape((size, size)) # 2차원 배열로 변환
    avg = pixels.mean() # 평균 구하기
    diff = 1*(pixels>avg) # 평균보다 크면 1, 작으면 0으로 변환하기
    print(diff)
    return diff
```

이진 해시를 16진수 해시값으로 변환 함수 선언

```
def np2hash(n):
    bhash = []
    for n1 in ahash.tolist():
        s1 = [str(i) for i in n1]
        s2 = "".join(s1)
        i = int(s2,2) # 이진수를 정수로 변환하기
        bhash.append("%04x"%i)
    return "".join(bhash)
```

Average Hash 출력하기

```
ahash = average_hash('tower.jpg')
print(ahash)
print(np2hash(ahash))
```

학습을 시킬 준비가 됨

```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
Python 3.6.3 (v3.6.3:2c5fed8, Oct 3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:\파이썬 연습\hash2.py =====
==
[[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0]
 [0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0]
 [0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0]
 [1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0]
 [0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0]]
0000000001800180010001800180018001800180018001801ffc1ffcbbfe0ffe
>>> |
```

출력결과

실습3. 유사 이미지 검출하기

이미지와 라이브러리 호출

```
from PIL import Image
import numpy as np
```

이미지 데이터를 Averages Hash로 변환 함수 선언

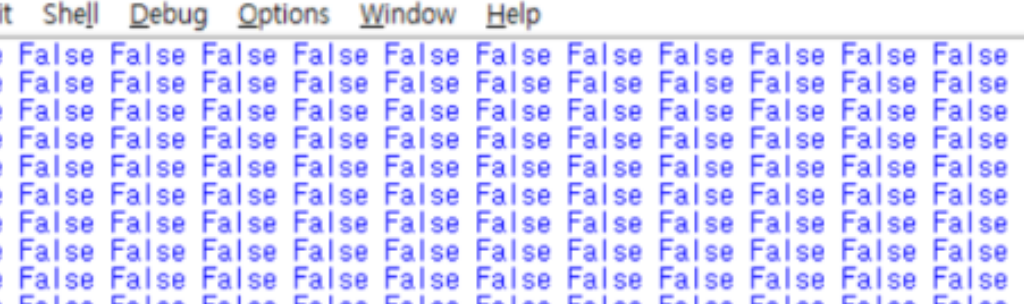
```
def average_hash(fname, size = 16): #average_hash(파일이름, 사이즈)
    img = Image.open(fname) # 이미지 데이터 열기
    img = img.convert('L') # 그레이스케일로 변환
    #1'지정하게 되면 이진화 그밖에 "RGB", "RGBA", "CMYAK" 모드 지정가능
    img = img.resize((size, size), Image.ANTIALIAS) # 리사이즈
    pixel_data = img.getdata() # 픽셀 데이터 가져오기
    pixels = np.array(pixel_data) # Numpy 배열로 변환하기
    pixels = pixels.reshape((size, size)) # 2차원 배열로 변환
    avg = pixels.mean() # 평균 구하기
    diff = 1*(pixels>avg) #평균보다 크면 1, 작으면 0으로 변환하기
    return diff
```

이전 16진수 해시로 변환하는 함수 선언

```
def np2hash(n):
    bhash = []
    for n1 in ahash.tolist():
        s1 = [str(i) for i in n1]
        s2 = "".join(s1)
        i = int(s2,2) #이진수를 정수로 변환하기
        bhash.append("%04x"%i)
    return "".join(bhash)
```

Average Hash 출력하기

```
avhash_1 = average_hash('tower.jpg')
avhash_2 = average_hash('tower.jpg')
a = avhash_1.reshape(1,-1) # 2차원 배열이 다시 1차원 배열로 변환됨
b = avhash_2.reshape(1,-1) # 2차원 배열이 다시 1차원 배열로 변환됨
print((a != b)) # a와 b의 같은부분 다른 부분 출력
print('\n',(a != b).sum()) # 다른 포인트 출력 = 해밍 거리
```

The screenshot shows a Python 3.6.3 Shell window. The title bar reads "Python 3.6.3 Shell". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area displays a list of 16 "False" values, each on a new line, enclosed in square brackets: `[False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False]`. The prompt `>>>` is visible at the bottom left.

출력결과