AIoT AutoCar Prime 으로 배우는

온디바이스 AI 프로그래밍

7. 인공지능

7.1 머신러닝

인공지능

- 인공지능은 지적 능력 수준에 따라 3가지로 분류 가능
  - 약인공지능 : 특정 분야에서만 인간보다 우수한 능력을 보이는 인공지능
    - 바둑에 특화된 구글의 알파고, 머신러닝
  - 강인공지능 : 모든 분야에서 인간 수준의 지적 능력을 갖추고 있는 인공지능
    - 비대면 대화를 했을 때 인간과 구분이 힘든 수준의 인공지능
    - 영화 '아이언맨'의 자비스
  - 초인공지능 : 모든 분야에서 인간을 초월한 수준의 인공지능

머신러닝

- 머신러닝 : 컴퓨터 시스템이 주어진 데이터를 학습하는 과정
  - 지도 학습
    - 훈련 데이터로부터 학습 결과가 어떻게 출력되어야 하는지 알려주며 학습 모델을 유도
    - 학습 모델이 추측한 값과 실제 결과와 비교하며 모델을 최적화하는 학습 방법
  - 비지도 학습
    - 훈련 데이터로부터 학습 모델을 유도할 때 목표값 없이 학습 모델 스스로 각 데이터의 특징을 추론
    - 특징값의 편차에 따라 데이터를 군집화하는 학습 방법
  - 강화 학습
    - 보상이라는 개념을 이용
    - 학습 모델이 이전 출력과 비교해 더 나은 결과를 출력할 때 보상을 주며 오차를 줄여나가는 학습 방법.

지도 학습

- 지도 학습의 장점 :
  - 짧은 학습 시간을 투자해 낮은 오차를 얻을 수 있음
  - 데이터양 대비 감소하는 오차가 적은 편
    - 학습의 한계가 빨리 나타남
    - 복잡한 학습 모델일수록 필요한 데이터양은 급격히 증가
  - 훈련에 필요한 데이터를 가공하는 과정이 필요
  - 학습 데이터와 목표가 단순한 문제에 적합
  - 기본적인 지도 학습 기법 : 분류, 회귀

분류

- 분류
  - 입력과 처리 결과로 이루어진 훈련 데이터에서 입력 처리 기준을 학습
  - 학습된 기준에 따라 새로운 데이터를 어떤 종류로 구분할지 선택하는 기법
  - 훈련 데이터
    - 값과 클래스 또는 라벨로 이루어져 있음
  - 분류 모델
    - 값과 클래스를 학습해 새로운 값이 입력되면 어떤 클래스를 부여할지 선택

분류

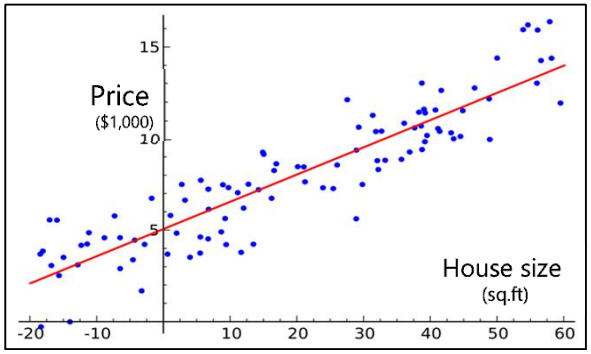
- 분류 기법
  - 이진 분류
    - 클래스가 True와 False로 이루어져 있는 모델로, 데이터를 두 가지로 분류
    - 로지스틱 회귀를 이용해 구현하기도 함
  - 다중 분류
    - 클래스가 여러 개로 이루어져 있는 모델로, 데이터를 여러 클래스로 분류
    - 소프트맥스 회귀를 이용해 구현하기도 함

회귀

- 회귀
  - 입력과 처리 결과로 이루어진 훈련 데이터로부터 두 값의 관계를 학습
  - 어떤 데이터가 새롭게 입력될지 예측하는 기법

선형 회귀

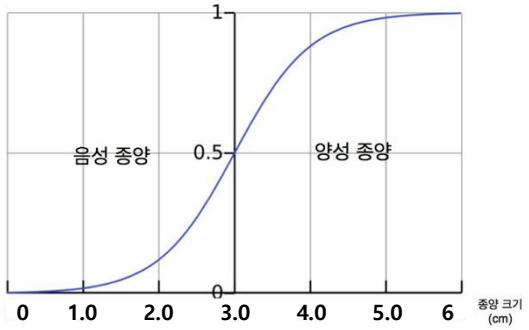
- 입력과 처리 결과를 선형적 관계로 모델링
- 관계식이 1차원 방정식
- 결과값의 범위 : $-\infty \sim \infty$
- 입력과 결과에 대한 관계식을 추측
- 대표적으로 함수 회귀 모델, 기후 예측 모델이 있음



집 면적 대비 가격에 대한 선형 회귀

로지스틱 회귀

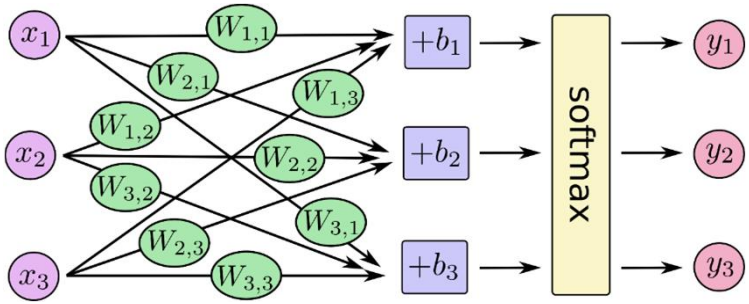
- 입력과 처리 결과를 이항 확률 관계로 모델링
- 관계식의 결과는 2개이며 값은 0과 1 사이
- 입력이 True(1)와 False(0) 중 어느 것일 확률이 높은지에 대한 관계식을 갖음
- 대표적으로 논리 회귀 모델, 시험 합격/불합격 예측 모델이 있음



종양의 크기에 따라 음성과 양성을 구분하는 로지스틱 회귀

소프트맥스 회귀

- 입력과 처리 결과를 다항 확률 관계로 모델링
- 관계식의 결과는 3개 이상이며 값은 0과 1 사이
- 입력이 3개 이상의 클래스 중 각 클래스일 확률에 대한 관계식을 갖음
- 대표적으로 품종 예측 모델이 있음



소프트맥스 회귀

비지도 학습

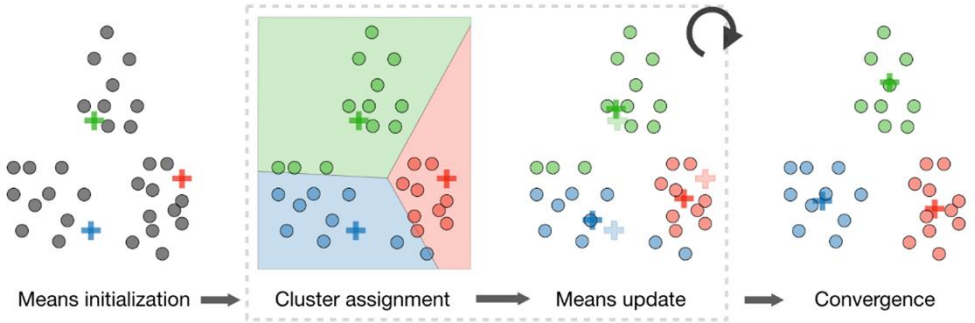
- 학습 초기 오차가 비교적 높음
- 시간이 지날수록 오차가 급격히 감소
- 학습 데이터를 가공할 필요 없이 많은 학습을 진행할 수 있는 장점이 있음
- 데이터 특징을 스스로 찾음
  - 설계자가 예측하지 못한 특징을 찾아낼 수도 있음
- 학습 시간이 긴 편이고 목표가 분명하지 않음
  - 원하지 않은 학습 모델이 나올 수 있음
- 주어진 학습 데이터가 복잡하거나 데이터 특징을 분석할 때 적합한 방법
- 기본적인 비지도 학습 기법 : 군집화

군집화

- 새로운 데이터에 대한 군집 기준을 만들어내는 기법
  - 입력으로만 이루어진 훈련데이터에서 비슷한 것끼리 군집시킴
- 군집 기준값과의 오차가 가장 낮은 군집에 소속시키고 군집 기준값을 조정
- 군집 기준값을 조정해나가며 최적의 값을 찾는 과정이 학습 과정
  - 군집 기준값을 이용해 새로운 데이터의 군집을 결정하는 과정이 응용 과정

군집화

- K 평균 군집화 (군집화 기법)
  - 군집의 개수 K 개의 군집을 데이터 밀도가 높은 곳에 군집화
  - 입력과 관계없이 초기값을 랜덤으로 정함
  - 원치 않는 결과가 나올 수 있으며 학습할 때마다 결과가 변함
  - 군집마다 밀도 차이가 크거나 그 경계가 복잡하고 모호할수록 원하는 결과를 얻기 힘듦



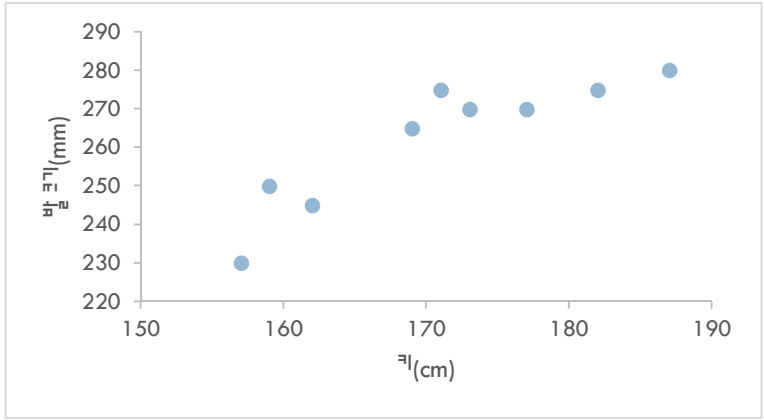
머신러닝 기법

- 대표적인 머신러닝 기법
  - 선형 회귀, 로지스틱 회귀, 소프트맥스 회귀, K-평균 군집화 등이 있음
  - 이 기법들은 딥러닝의 기반이 되고 전처리 알고리즘으로 사용되기도 함

선형 회귀

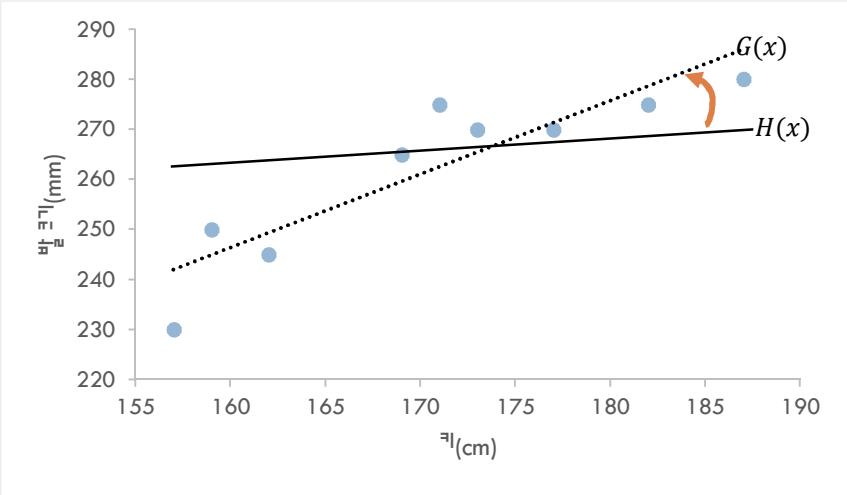
- 입력에 대한 결과가 선형 관계
  - 선형 관계 : 입력이 결과에 대해 직접적인 관계가 있을 때

키(cm)	발 크기(mm)
173	270
171	275
162	245
187	280
157	230
169	265
177	270
159	250
182	275



선형 회귀

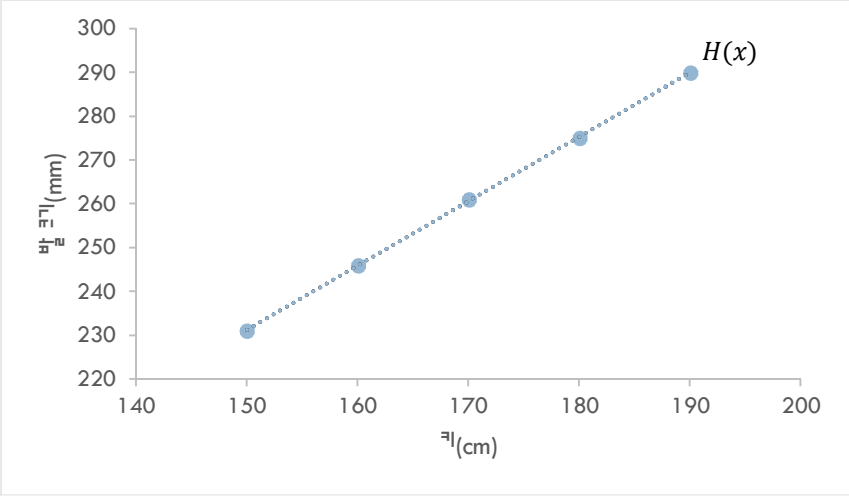
- 직선 G(x) : 모든 데이터의 관계를 하나의 직선을 표현
- 가설 H(x) : G(x)를 구하기 위해 선형 모델이 임의로 설정한 직선
- 최적화 : 선형 회귀 모델이 가설 H(x)를 조절하며 관계식 G(x)를 찾는 과정



선형 회귀

- 성공적으로 가설 최적화를 끝낸 선형 회귀 모델
  - 새로운 키 데이터가 주어졌을 때 발 크기가 몇일지 선형적으로 예측

키(cm)	발 크기(mm)
190	291
180	276
170	261
160	246
150	230



선형 회귀

- Keras와 Pop.AI 라이브러리를 비교

Keras를 이용해 구현한 선형 회귀

```
01: from tensorflow import keras
02:
03: model = keras.models.Sequential()
04:
05: model.add(keras.layers.Input(shape=(1,)))
06: model.add(keras.layers.Dense(1))
07:
08: model.compile(loss='MSE', optimizer='SGD')
09:
10: model.fit(X, Y, epochs=100)
```

Pop.AI를 이용해 구현한 선형 회귀

```
01: from pop import AI
02:
03: LR = AI.Linear_Regression()
04:
05: LR.train()
06: LR.run()
```

선형 회귀

- Keras와 Pop.AI 라이브러리를 비교
  - Keras는 범용성을 위해 만들어진 라이브러리
    - 선형 회귀를 구현하려면 학습 모델 설계, 손실 함수, 최적화 함수 등 고려해야할 것이 많음
  - Pop.AI 라이브러리는 특정 모델을 쉽게 사용할 수 있도록 사전 설계
    - 입문자가 빠르게 실습 가능
    - Pop.AI 라이브러리는 Keras를 기반으로 설계
    - Keras를 이용한 학습 모델 설계, 손실 함수 등은 이후 챕터에서 진행

선형 회귀

- pop.AI 라이브러리로 선형 회귀 구현
  - Linear_Regression 객체
    - restore: 최근 모델에 이어서 학습할지에 대한 여부를 Boolean으로 입력 (기본값: False)
    - ckpt_name: 저장 및 불러올 모델 파일의 이름 (기본값: linear_regression)
  - pop.AI라이브러리 import, Linear_Regression객체를 LR이라는 변수에 생성

```
01:     from pop import AI
02:
03:     LR = AI.Linear_Regression()
```

선형 회귀

- Linear_Regression 객체의 속성
  - X_data : 입력 데이터
  - Y_data : 입력에 대한 결과값 데이터
  - 입력 리스트와 결과 리스트는 1대1 대응

```
04:     LR.X_data = [[173],[171],[162],[187],[157],[169],[177],[159],[182]]
05:     LR.Y_data = [[270],[275],[245],[280],[230],[265],[270],[250],[275]]
```

선형 회귀

- Linear_Regression 객체의 train() 메소드 : 회귀 학습을 시작
  - 파라미터 : times와 print_every
    - times : 학습할 횟수 (기본값은 100)
    - print_every : 학습 상황을 몇 번째마다 출력할지를 의미 (기본값은 10)
  - train 메소드 : 실행하면 10회마다 회귀 모델의 오차 출력

```
06:          LR.train()
```

선형 회귀

- Linear_Regression 객체의 run() 메소드 : 학습된 모델 사용
  - 파라미터 : inputs
    - 모델에 사용할 데이터
    - inputs에는 [[172], [162]]와 같이 2차원 리스트로 입력
    - 기본값은 X_data를 사용
  - run 메소드를 실행하면 입력에 대한 회귀 모델의 예측 발 크기를 출력

```
07:             LR.run()
```

선형 회귀

전체 코드

```
01:     from pop import AI
02:
03:     LR = AI.Linear_Regression()
04:
05:     LR.X_data = [[173],[171],[162],[187],[157],[169],[177],[159],[182]]
06:     LR.Y_data = [[270],[275],[245],[280],[230],[265],[270],[250],[275]]
07:
08:     LR.train()
09:     LR.run()
```

선형 회귀

- 추가 학습을 위해 train 메소드의 times 파라미터를 1000으로 설정하고 학습
- print_every 파라미터를 이용해 출력량을 조절가능

```
10:             LR.train(times=1000, print_every=100)
```

선형 회귀

- 이전 학습 모델에 이어서 1,000회 학습해 총 1,100회를 학습
- 100회마다 학습 오차를 출력
- run 메소드를 이용해 학습 모델 예측값 출력

```
11:             LR.run()
```

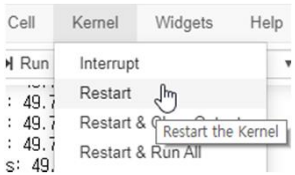
선형 회귀

- run 메소드의 파라미터로 새로운 데이터를 입력하여 출력 확인

```
12:             LR.run([[150], [160], [170], [180], [190]])
```

선형 회귀

- 프로그램이 종료된 이후 학습 모델을 다시 불러와 사용하는 방법
  - Jupyter Notebook 상단 툴바에서 커널 재시작



선형 회귀

- AI 모듈을 import
- Linear_Regression 객체를 생성할 때 restore 파라미터를 True로 설정

```
01:     from pop import AI
02:
03:     LR = AI.Linear_Regression(restore=True)
```

- Linear_Regression 객체에 X_data와 Y_data를 입력
- run메소드 호출 -> 이전에 학습된 모델의 출력 결과 확인

```
04:     LR.X_data = [[173],[171],[162],[187],[157],[169],[177],[159],[182]]
05:     LR.Y_data = [[270],[275],[245],[280],[230],[265],[270],[250],[275]]
06:
07:     LR.run()
```

선형 회귀

- train 메소드를 호출하면 이전 학습 모델에 이어서 학습 가능

```
08:         LR.train()
```

- Restore 파라미터를 설정하지 않거나 False로 설정한 경우
  - 이전 학습 모델에 덮어 씌워지므로 주의
  - ckpt_name 파라미터에 학습 모델을 구분할 수 있는 이름을 지정하여 개별 저장 가능

```
09:         LR = AI.Linear_Regression(restore=True, ckpt_name="model_1")
```

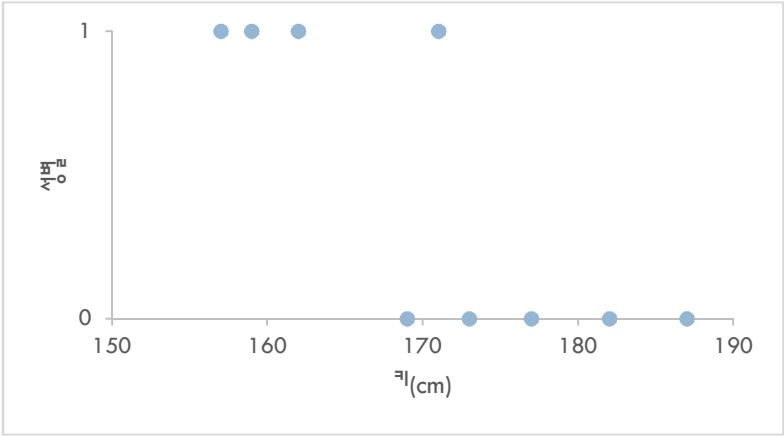
로지스틱 회귀

- 입력에 대해 이항 확률 관계
- 입력이 주어졌을 때 결과를 0~1 사이값으로 표현
  - 입력이 True(1)일 확률을 의미

로지스틱 회귀

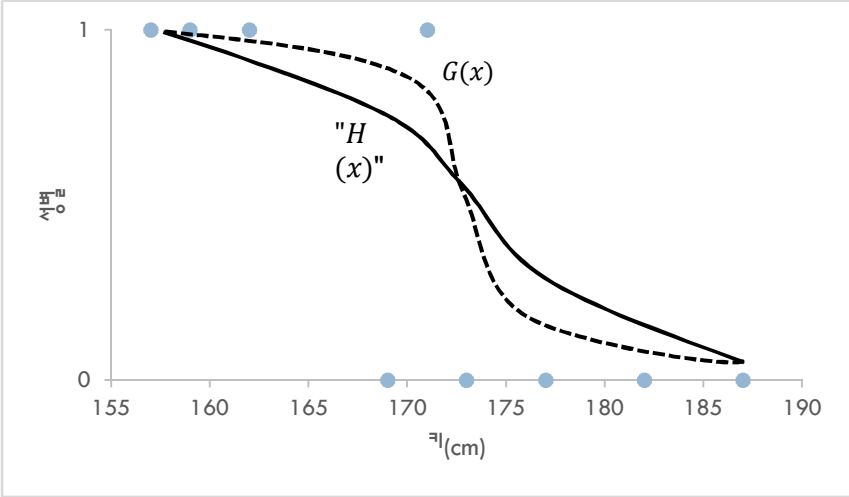
- 키에 따른 성별을 예측하기 위한 회귀 모델을 구하는 예
  - 키와 성별에 관한 데이터가 주어짐
  - 성별 값은 True(1)가 여성, False(0)가 남성일 경우로 설정

키(cm)	성별
173	0
171	1
162	1
187	0
157	1
169	0
177	0
159	1
182	0



로지스틱 회귀

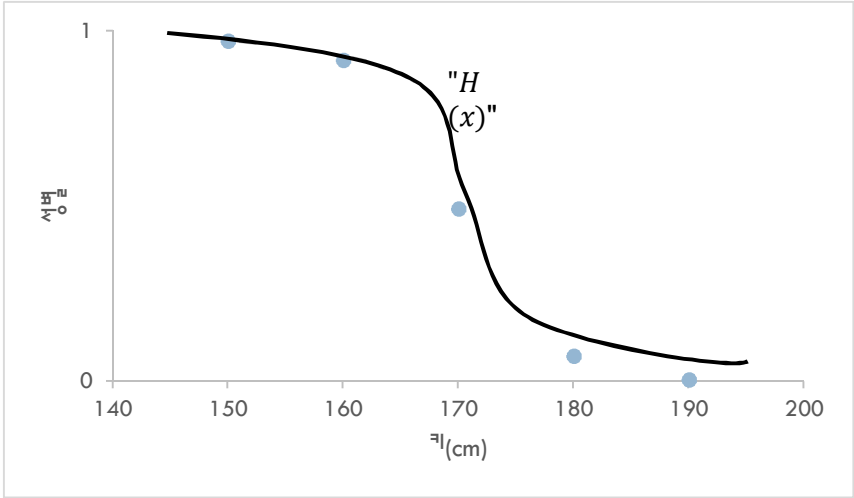
- G(x) : 모든 데이터의 관계를 하나의 선으로 표현
- G(x)를 구하기 위해 로지스틱 회귀 모델이 임의의 곡선을 가설 H(x)로 설정
- 로지스틱 회귀 모델은 가설 H(x)를 데이터의 관계를 가장 잘 표현하도록 최적화



로지스틱 회귀

- 최적화에 성공한 로지스틱 회귀 모델
  - 새로운 키 데이터가 주어졌을 때 여성일 확률을 예측
  - 단, 여성보다 키가 작은 남성이 존재하므로 확률로 예측

키(cm)	성별
190	0.005
180	0.073
170	0.493
160	0.916
150	0.972



로지스틱 회귀

- 로지스틱 회귀 실습
  - 키와 성별 데이터를 입력받아 로지스틱 회귀
  - 새로운 키 데이터를 입력하면 성별을 예측하는 모델을 실습
  - Pop.AI라이브러리 import, Logistic_Regression객체를 LR변수에 생성

```
01:     from pop import AI
02:
03:     LR = AI.Logistic_Regression()
```

로지스틱 회귀

- Logistic_Regression 객체 파라미터
  - input_size: 입력 데이터의 크기 (기본값: 1)
  - restore: 최근 모델에 이어서 학습할지에 대한 여부를 입력 (기본값: False)
  - ckpt_name: 저장 및 불러올 모델 파일의 이름 (기본값: logistic_regression)
- Logistic_Regression 객체 속성
  - X_data : 입력 데이터
  - Y_data : 입력에 대한 결과값 데이터

```
04:         LR.X_data = [[173],[171],[162],[187],[157],[169],[177],[159],[182]]
05:         LR.Y_data = [[0],[1],[1],[0],[1],[0],[0],[1],[0]]
```

로지스틱 회귀

- Logistic_Regression 객체의 train() 메소드
  - 회귀 학습 시작
  - 파라미터 times : 기본값 100
  - 파라미터 print_every : 기본값 10
  - Train 메소드를 실행하면 10회마다 회귀 모델 오차 출력

```
06:          LR.train()
```

로지스틱 회귀

- Logistic_Regression 객체의 run() 메소드
  - 학습된 모델을 사용 가능
  - 파라미터 inputs : 기본값 X_data 사용
  - run 메소드를 실행하면 입력에 대한 회귀 모델의 성별 확률을 출력

```
07:             LR.run()
```

로지스틱 회귀

전체 코드

```
01:     from pop import AI
02:
03:     LR = AI.Logistic_Regression()
04:
05:     LR.X_data = [[173],[171],[162],[187],[157],[169],[177],[159],[182]]
06:     LR.Y_data = [[0],[1],[1],[0],[1],[0],[0],[1],[0]]
07:
08:     LR.train()
09:     LR.run()
```

로지스틱 회귀

- 추가 학습 : train 메소드의 times 파라미터를 10000으로 설정하고 학습

```
10:          LR.train(times=10000, print_every=1000)
```

  - 이전 학습 모델에 이어서 10,000회 학습해 총 10,100회를 학습
  - 1000회마다 학습 오차를 출력
  - run 메소드를 이용해 학습 모델의 예측값을 출력

```
11:          Logistic_Regression.run()
```

로지스틱 회귀

- run 메소드의 파라미터로 새로운 데이터를 입력하여 출력 확인

```
12:         LR.run([[150], [160], [170], [180], [190]])
```

- restore 파라미터를 True로 설정하면 최근 사용한 학습 모델을 불러와 다시 사용 가능

```
01:         LR = AI.Logistic_Regression(restore=True, ckpt_name="model_1")
```

소프트맥스 회귀

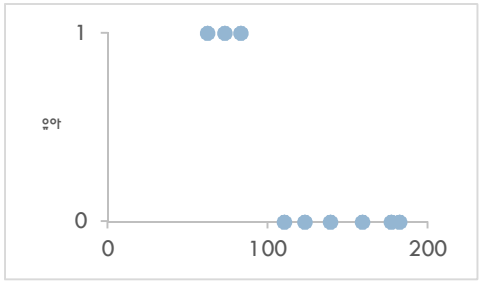
- 인공신경망(ANN) 개념 필요
  - 인공신경망은 딥러닝에서 설명
  - 현재 챕터에서는 다차원 방정식으로 대체하여 설명
- 입력과 결과가 다항 확률 관계
- 입력이 주어졌을 때 각 클래스별로 0~1 사이값으로 표현
  - 모든 클래스 값의 합은 1

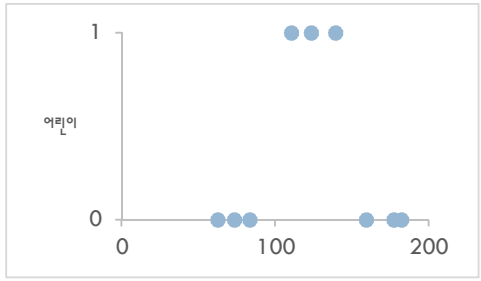
소프트맥스 회귀

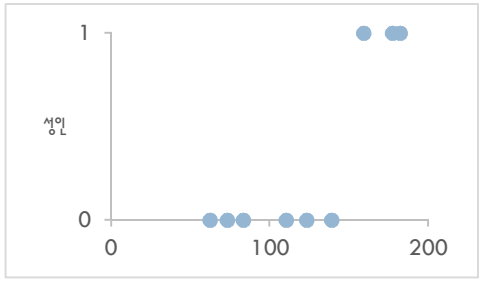
- 키에 따른 연령층을 예측하기 위한 회귀 모델을 구하는 예제
  - 키와 연령층에 관한 데이터가 주어졌을 때
  - 각 연령층에 관한 클래스는 유아, 어린이, 성인으로 설정
  - 클래스의 개수 K는 3

키(cm)	유아	어린이	성인
73	1	0	0
62	1	0	0
83	1	0	0
110	0	1	0
139	0	1	0
123	0	1	0
177	0	0	1
159	0	0	1
182	0	0	1

소프트맥스 회귀

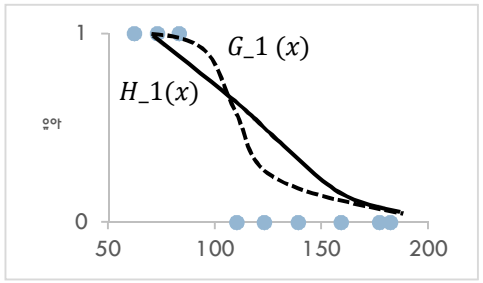


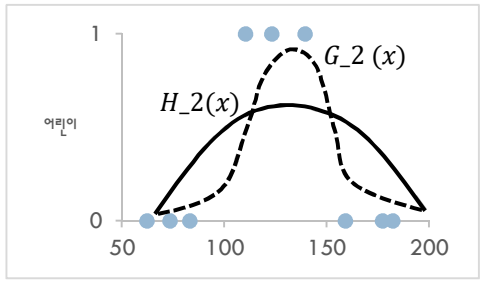


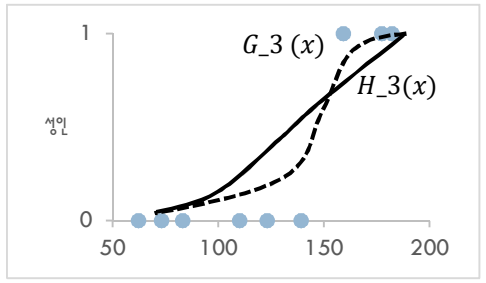


소프트맥스 회귀

- G_i (x) : 모든 데이터와 클래스 간의 관계를 표현
  - G_i (x)을 구하기 위해 소프트맥스 회귀 모델이 임의의 곡선들을 가설 H_i (x)로 설정
  - 각 클래스에 대한 가설들 H_i (x)을 데이터의 관계를 가장 잘 표현하도록 최적화





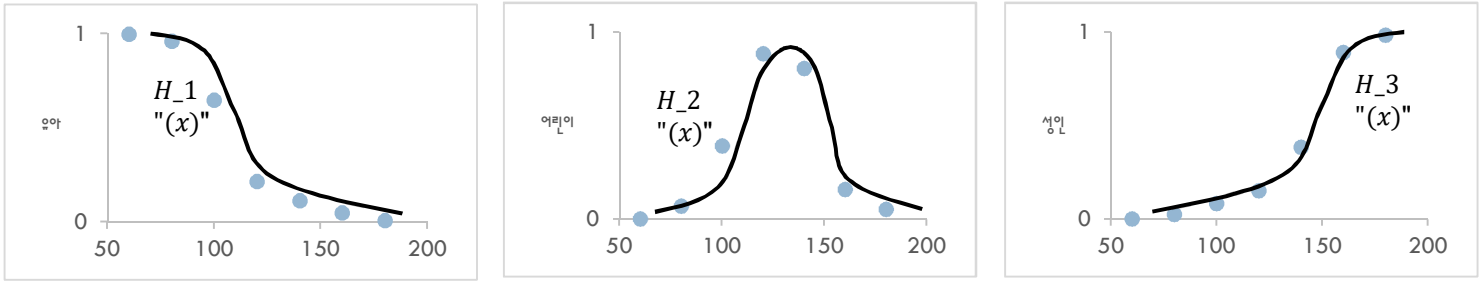


소프트맥스 회귀

- 최적화에 성공한 소프트맥스 회귀 모델
  - 새로운 키 데이터가 주어졌을 때 유아, 어린이, 성인일 확률 예측 가능
  - 모든 클래스 값의 합이 1이어야 하므로 소프트맥스 함수를 이용해 각 클래스 값 조정

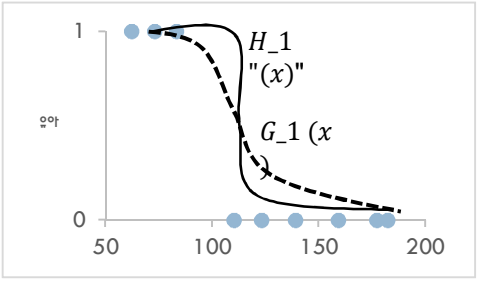
키(cm)	유아	어린이	성인
60	0.998	0.002	0
80	0.961	0.068	0.025
100	0.647	0.391	0.083
120	0.217	0.884	0.152
140	0.113	0.807	0.384
160	0.051	0.16	0.891
180	0.008	0.052	0.985

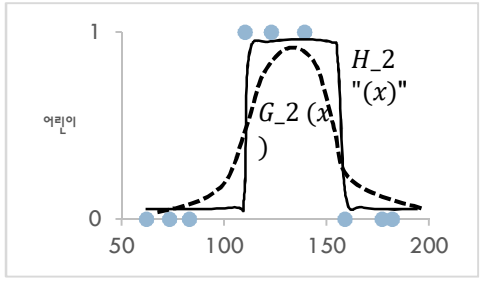
소프트맥스 회귀

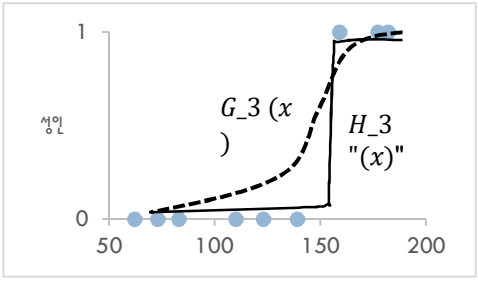


소프트맥스 회귀

- 과적합 (Overfit)
  - 과도하게 최적화 과정을 진행하면 극단적인 형태의 모델 H_i (x)이 생성
  - 과도한 최적화로 인해 원하는 결과를 얻을 수 없는 상태
- 과소적합 (Underfit)
  - 부족한 최적화로 인해 원하는 결과를 얻을 수 없는 상태







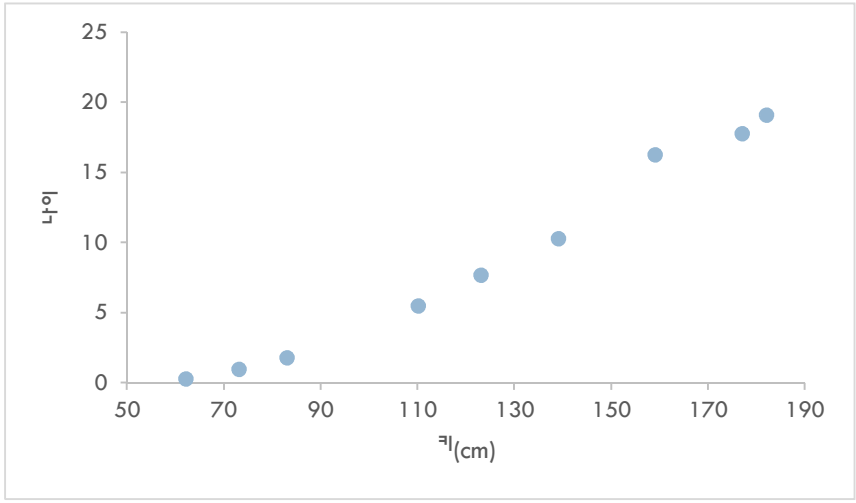
K-평균 군집화

- 예측과 최대화를 반복하며 최적해로 수렴하는 EM알고리즘을 기반
  - 1차 : 군집 기준점으로부터 가까운 데이터들을 묶음
  - 2차 : 묶인 데이터들의 중심점을 군집 기준점으로 재설정하는 과정 반복

K-평균 군집화

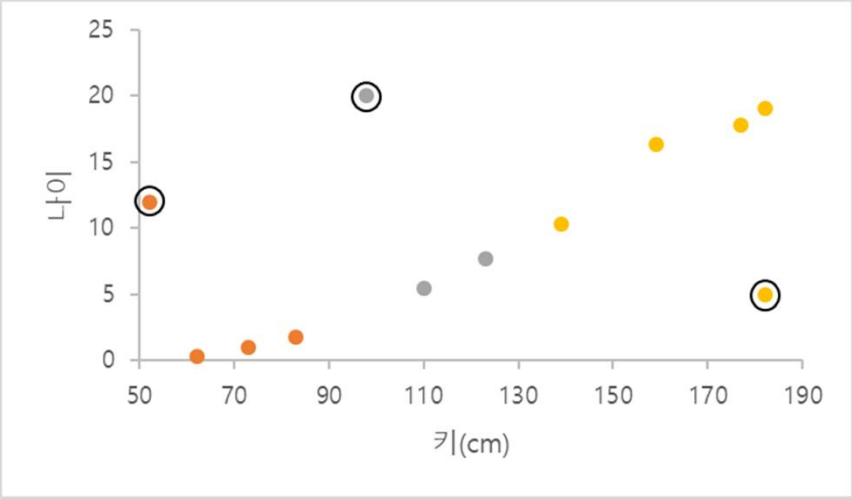
- 데이터들의 특징에 따라 K개의 군집을 구하는 예제
  - 키와 나이에 대한 데이터가 주어졌을 때

키(cm)	나이
73	1
62	0.3
83	1.8
110	5.5
139	10.3
123	7.7
177	17.8
159	16.3
182	19.1



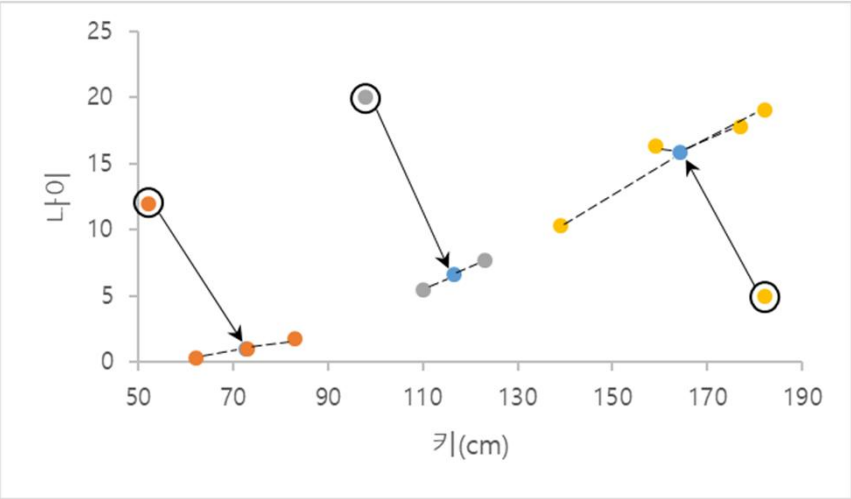
K-평균 군집화

- 랜덤으로 K개의 군집점 설정
- 각 군집점을 기준으로 가까운 데이터들을 묶음



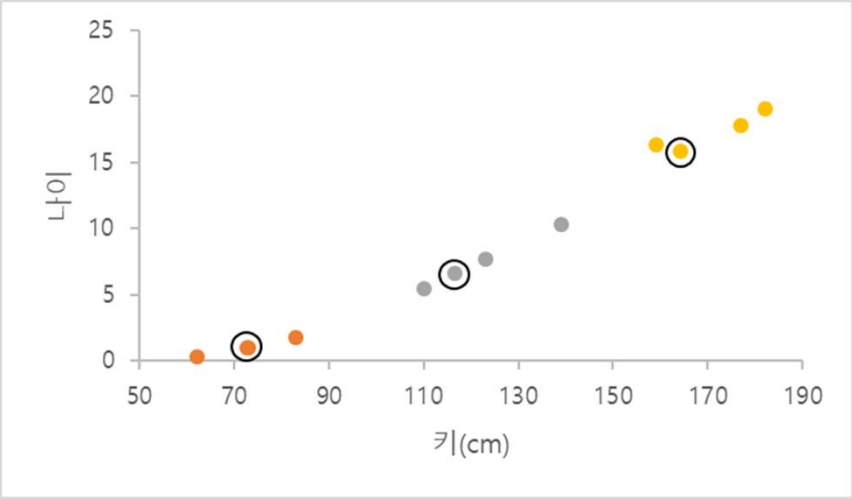
K-평균 군집화

- 묶인 데이터들의 중심점을 찾고 이 점을 군집점으로 재설정



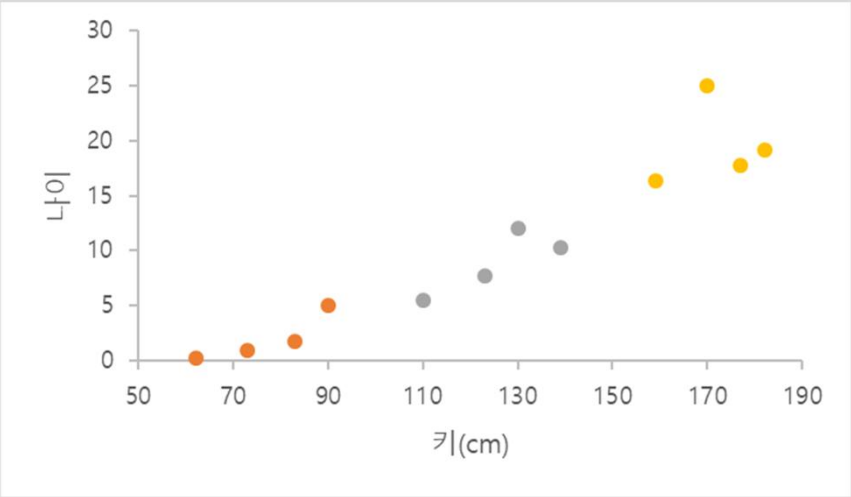
K-평균 군집화

- 이 과정을 군집점과 중심점의 오차가 최솟값이 될 때까지 반복 (최적화)



K-평균 군집화

- 최적화에 성공한 K-평균 군집화 모델
  - 새로운 키 데이터가 주어졌을 때 가장 가까운 군집에 포함
  - 필요에 따라 군집점을 재설정



내용 정리

- 인공지능
  - 약인공지능 : 특정 분야에서 우수한 능력을 가진 인공지능
  - 강인공지능 : 모든 분야에서 인간과 비슷한 능력을 가진 인공지능
  - 초인공지능 : 모든 분야에서 인간을 초월한 능력을 가진 인공지능
- 머신러닝 : 컴퓨터 시스템이 데이터를 학습하는 과정

내용 정리

- 지도 학습
  - 컴퓨터가 출력한 결과와 비교하여 머신러닝 모델을 최적화하는 방법
    - 학습 결과가 어떻게 출력되어야 하는지 알려준 상태
  - 분류 : 분류 기준을 학습하고 새로운 데이터를 어떤 결과로 분류할지 선택
    - 입력과 분류 결과가 주어진 상태
  - 회귀 : 두 값의 수학적 관계를 학습하고 새로운 데이터의 결과의 출력 예측
    - 입력과 결과가 주어진 상태

내용 정리

- 비지도 학습
  - 컴퓨터가 입력 데이터의 특징을 추론하고 특징의 기준을 구체화하는 방법
    - 입력 데이터만 주어진 상태
  - 군집화
    - 입력 데이터 값이 비슷한 것끼리 군집시켜 군집 기준을 구체화
    - 새로운 데이터에 대한 군집 그룹을 정하는 기법
- 강화 학습
  - 더 나은 결과를 출력할 때 보상을 주며 오차를 줄여나가는 학습 방법

내용 정리

- 머신러닝 기법
  - 선형 회귀 : 입력과 결과의 선형 관계를 분석하는 머신러닝 모델
  - 로지스틱 회귀 : 입력과 결과에 대해 이항 관계를 분석하는 머신러닝 모델
  - 소프트맥스 회귀 : 입력과 결과에 대해 다항 관계를 분석하는 머신러닝 모델
  - K-평균 군집화 : 가까운 입력 데이터끼리 군집시키는 방법
    - 군집 기준을 추론하는 머신러닝 모델

연습문제

- 문제 35. 다음 문장들을 읽고 빈 칸을 채워보세요.
  - A. 특정 분야에서 우수한 능력을 가진 인공지능을 [ ]이라 한다.
  - B. 컴퓨터 시스템이 주어진 데이터를 학습하는 과정을 [ ]이라 한다.
  - C. 머신러닝은 학습 방식에 따라 [ ], [ ], [ ] 으로 나눌 수 있다.
  - D. 대표적 머신러닝 기법은 [ ], [ ], [ ], [ ]화 등이 있다.

연습문제

- 문제 36. 다음 코드는 Pop.AI 라이브러리를 이용하여 선형 회귀를 구현한 코드입니다. 질문을 읽고 답해보세요.

```
01:     from pop import AI
02:
03:     LR = AI.Linear_Regression()
04:
05:     LR.X_data = [X]
06:     LR.Y_data = [Y]
07:
08:     LR.train(times=1000, print_every=100)
09:     LR.run()
```

연습문제

- A. 수식 y=2x 를 회귀할 수 있도록 빈 칸 X, Y에 들어갈 학습 데이터셋을 작성해 보세요.
- B. A에서 작성한 데이터셋으로 학습시키고, 100을 입력했을 때 출력을 작성하세요.

연습문제

- C. 다음과 같은 주가 그래프와 표가 주어졌을 때 선형 회귀하여 15시 30분 예측 값과 학습 횟수(Step)을 작성하세요.

9	9.5	10	10.5	11	11.5	12
48.7	48.45	48.45	48.6	48.7	48.9	49.05
12.5	13	13.5	14	14.5	15	
49.05	48.95	49.05	48.9	49.35	49.65	

연습문제

- 문제 37. 다음 코드는 Pop.AI 라이브러리를 이용하여 선형 회귀를 구현한 코드입니다. 질문을 읽고 답해보세요.

```
01:     from pop import AI
02:
03:     LR = AI.Logistic_Regression()
04:
05:     LR.X_data = [ X ]
06:     LR.Y_data = [ Y ]
07:
08:     LR.train()
09:     LR.run()
```

연습문제

- A. 양수면 1, 음수면 0을 출력하도록 회귀하고자 할 때 빈 칸 X, Y에 들어갈 학습 데이터셋을 작성하세요.
- B. 로지스틱 회귀 모델에 −1, 100, −0.01를 입력했을 때의 출력을 확인해보고, −0.01을 입력했을 때 오차가 0.1 이하인 모델을 만들어보고 모델의 손실율을 작성하세요.

연습문제

- 문제 38. 아래의 코드는 AIoT AutoCar의 Cds센서를 이용하여 밝기 값을 출력하는 코드입니다. 조도계를 사용하거나, 스마트폰에서 '조도계' 애플리케이션을 다운 받아 다음 문제를 해결해보세요. (단, 조도계의 단위는 Lux로 합니다.)

```
01:     from pop import Cds
02:
03:     cds = Cds(7)
04:
05:     value = cds.readAverage()
06:     print(value)
```

연습문제

- A. 조도계를 옆에 두고 코드를 실행시킨 후 조도계 값과 출력 값의 차이를 확인해보세요.
- B. 빈 배열을 생성하고, Cds 값을 0.5초 간격으로 10회 이상 추가하는 코드를 작성하세요.
- C. 빈 배열 2개를 생성하고, Cds 값과 조도계 값을 동시에 측정하여 Cds 값 배열과 조도계 값 배열을 만들어보세요
- D. Pop.AI 라이브러리의 Linear_Regression 클래스를 사용해 C에서 만든 두 배열을 각각 X, Y 데이터로 하고 선형 회귀하는 코드를 작성하세요.

연습문제

- E. 회귀 모델의 출력과 실제 조도계의 값을 비교해보고 데이터셋 추가 수집, 추가 학습 등 방법으로 ±30 lux 미만의 오차 범위를 갖는 회귀 모델을 만들어보세요.