AIoT AutoCar Prime 으로 배우는

온디바이스 AI 프로그래밍

7. 인공지능

7.2 딥러닝

딥러닝

- 딥러닝 : 인공신경망을 기반으로 설계된 머신러닝 기법
  - 인공신경망
    - 인간의 신경망을 모방하여 만들어진 알고리즘
    - 퍼셉트론에서 시작
  - 대표적인 딥러닝 기법
    - 심층신경망, 합성곱 신경망, 순환신경망 등

퍼셉트론

- 다수의 데이터로 하나의 결과를 출력하도록 하는 복합 논리 회로
  - 하나 이상의 데이터 값을 입력받고 각 입력에 가중치를 곱함
  - 이 값들을 모두 합해 기준값보다 크면 활성화(True)
  - 기준값보다 작으면 비활성화(False)
  - 다양한 요인이 결과를 만들 때, 각 요인이 결과에 미치는 영향을 분석하기 위해 사용

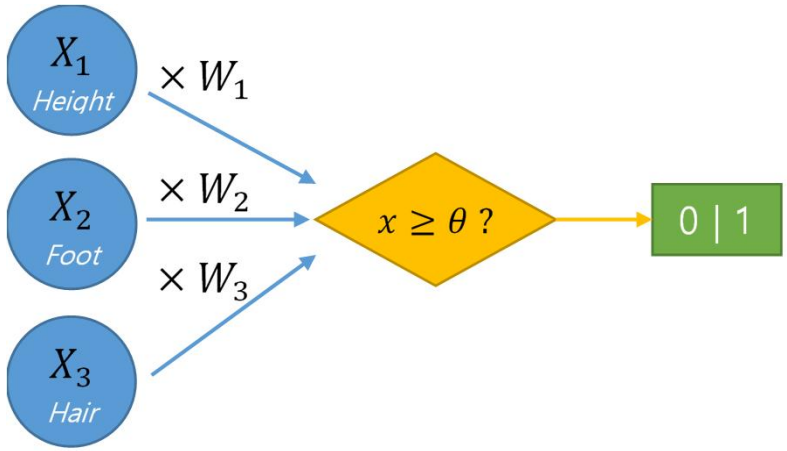
퍼셉트론

- 복합적 요인에 따른 성별 예측을 위한 퍼셉트론 모델 구하는 예제
  - 키, 발 크기, 머리카락 길이와 성별에 관한 데이터
  - 성별 값은 True(1)가 여성, False(0)가 남성일 경우로 설정

키(cm)	발 크기(mm)	머리카락 길이(cm)	성별
173	270	17	0
171	275	51	1
162	245	62	1
187	280	12	0
157	230	47	1
169	265	30	0
177	270	5	0
159	250	32	1
182	275	0	0

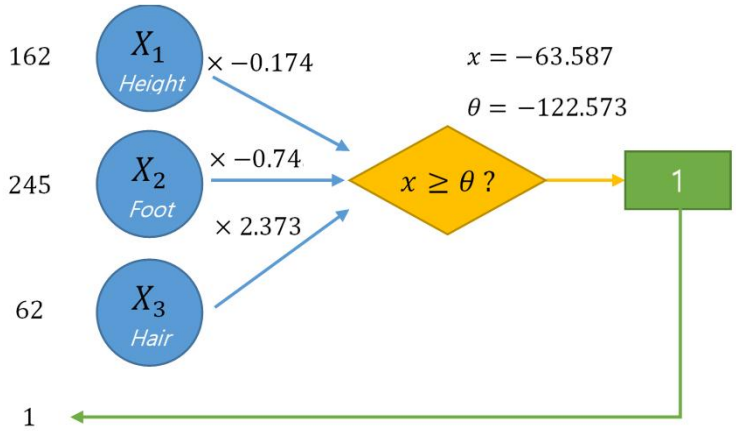
퍼셉트론

- 각 데이터에 가중치를 곱하고 모든 데이터를 합한 값 x과 기준값 θ 을 비교
- 가중치는 각 데이터가 결과에 가지는 영향력을 의미



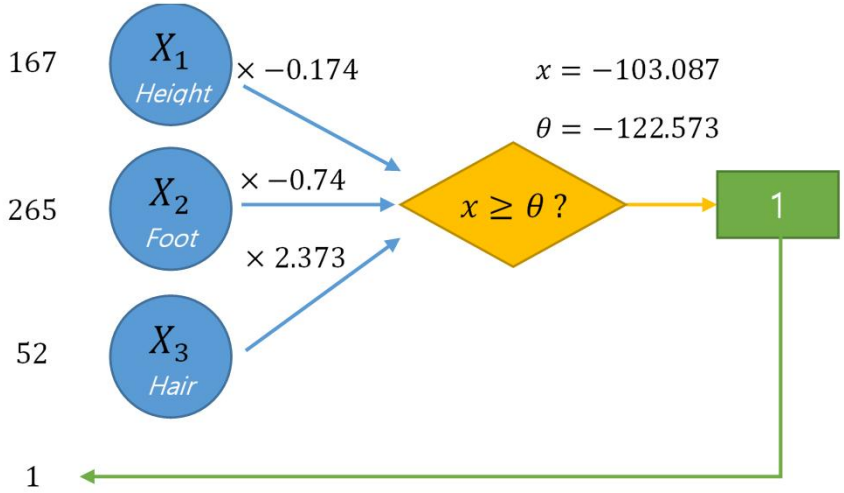
퍼셉트론

- 출력값과 실제값을 비교하고 가중치를 조절해 오차를 줄여나가도록 반복(최적화)
- 손실 함수와 비용 함수 : 모델의 출력값과 실제값을 비교하는 함수
  - 손실 함수 : 하나의 데이터셋을 대상으로 비교
  - 비용 함수 : 여러 데이터셋을 대상으로 통계를 내거나 성능을 평가



퍼셉트론

- 최적화에 성공한 퍼셉트론 모델
  - 새로운 복합 데이터가 주어졌을 때 여성인지 아닌지를 판단
  - 각 데이터에 대한 가중치를 조사해 성별에 대한 데이터의 영향력 분석 가능



퍼셉트론

- 성별을 예측하는 모델 실습
  - 키, 발 크기, 머리카락 길이, 성별 데이터를 입력받아 학습
  - 새로운 데이터를 입력하면 성별을 예측
  - Perceptron 객체의 파라미터
    - input_size: 입력 데이터의 크기를 의미하며 최하위 차원의 크기 입력 (필수 입력)
    - output_size: 결과 데이터의 크기를 의미 (기본값: 1)
    - restore: 최근 모델에 이어서 학습할지에 대한 여부를 Boolean으로 입력 (기본값: False)
    - ckpt_name: 저장 및 불러올 모델 파일의 이름 (기본값: perceptron)
    - softmax: 모델의 예측 결과에 총합이 1이 되도록 할지에 대한 여부를 입력 (기본값: True)
      - 결과 데이터의 크기가 2 이상일 때

퍼셉트론

- input_size는 사용자 입력이 필수
- input_size 외 각 기본값은 1, False, perceptron, True
- Pop.AI라이브러리 import, Perceptron 객체의 input_size 파라미터 3으로 설정
- Perc 변수 생성

```
01:     from pop import AI
02:
03:     Perc = AI.Perceptron(input_size=3)
```

퍼셉트론

- Perceptron 객체의 속성
  - X_data : 입력 데이터
  - Y_data : 입력에 대한 결과값 데이터

```
04:     Perc.X_data = [[173,270,17], [171,275,51], [162,245,62], [187,280,12],
                       [157,230,47], [169,265,30], [177,270,5], [159,250,32], [182,275,0]]
05:     Perc.Y_data = [[0],[1],[1],[0],[1],[0],[0],[1],[0]]
```

퍼셉트론

- Perceptron 객체의 train() 메소드
  - 퍼셉트론의 학습 시작
  - 파라미터 times : 학습할 횟수 (기본값은 100)
  - 파라미터 print_every : 학습 상황을 몇 번째마다 출력할지를 의미 (기본값은 10)
  - train 메소드를 실행하면 10회마다 퍼셉트론 모델의 오차 출력

```
06:          Perc.train()
```

퍼셉트론

- Perceptron 객체의 run() 메소드
  - 학습된 모델을 사용
  - 파라미터 inputs
    - 기본값은 X_data를 사용
  - run 메소드를 실행하면 입력에 대한 퍼셉트론 모델의 성별 확률 출력

```
07:          Perc.run()
```

퍼셉트론

전체 코드

```
01:     from pop import AI
02:
03:     Perc = AI.Perceptron(input_size=3)
04:
05:     Perc.X_data = [[173,270,17], [171,275,51], [162,245,62], [187,280,12], [157,230,47],
                       [169,265,30], [177,270,5], [159,250,32], [182,275,0]]
06:     Perc.Y_data = [[0],[1],[1],[0],[1],[0],[0],[1],[0]]
07:
08:     Perc.train()
09:     Perc.run()
```

퍼셉트론

- 추가 학습
  - train 메소드의 times 파라미터를 1000으로 설정하고 학습

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

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

```
11:          Perc.run()
```

퍼셉트론

- run 메소드의 파라미터로 새로운 데이터를 입력하여 출력 확인

```
12:     Perc.run([[174,265,6], [152,230,30], [162,255,10]])
```

- 학습 횟수를 과도하게 늘려 100,000회를 학습시켰을 때 결과 확인

```
13:     Perc.train(times=100000, print_every=10000)
```

- run 메소드의 파라미터로 새로운 데이터를 입력하여 출력 확인

```
14:     Perc.run([[152,230,28], [152,230,30]])
```

퍼셉트론

- 두 표본 데이터의 차이는 2cm 정도의 머리카락 길이임에도 극단적인 결과 출력
- 이 현상이 소프트맥스 회귀에서 언급된 과적합(Overfit) 현상
- 최적의 학습 모델은 무조건 학습 오차를 낮추는 것이 아니라 적절한 수준을 유지하는 것

퍼셉트론

- Perceptron 객체의 restore 파라미터를 True로 설정
  - 최근 사용한 학습 모델을 불러와 다시 사용 가능

```
15:          Perc = AI.Perceptron(input_size=3, restore=True, ckpt_name="model_1")
```

인공신경망

- 인공신경망은 퍼셉트론을 기반으로 고안
- 다수의 데이터로 하나 이상의 결과를 출력하도록 하는 알고리즘
  - 하나 이상의 데이터 값을 입력받고 각 입력에 가중치를 곱함
  - 이 값들을 모두 합해 활성화 함수에 입력
  - 0~1사이값으로 결과 출력

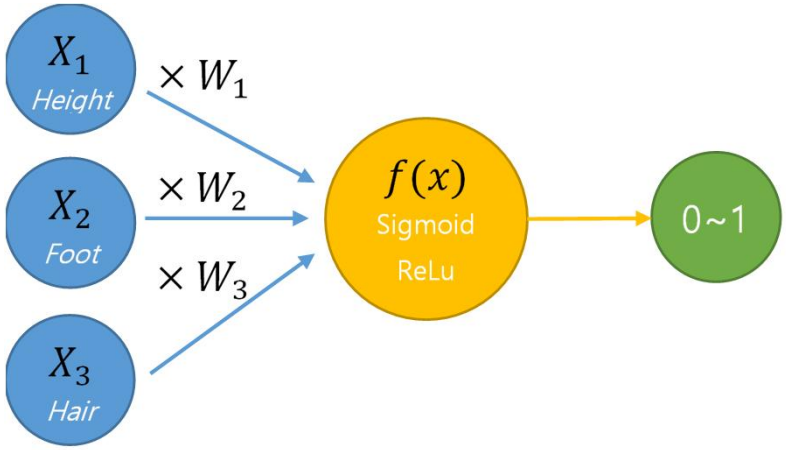
인공신경망

- 퍼셉트론 형태를 기반으로 기초적인 형태의 인공신경망 모델 설명
- 복합적 요인에 따른 성별을 예측하기 위한 인공신경망 모델 구하기
  - 키, 발 크기, 머리카락 길이와 성별에 관한 데이터가 주어졌을 때
  - 성별 값은 True(1)가 여성, False(0)가 남성일 경우로 설정

키(cm)	발 크기(mm)	머리카락 길이(cm)	성별
173	270	17	0
171	275	51	1
162	245	62	1
187	280	12	0
157	230	47	1
169	265	30	0
177	270	5	0
159	250	32	1
182	275	0	0

인공신경망

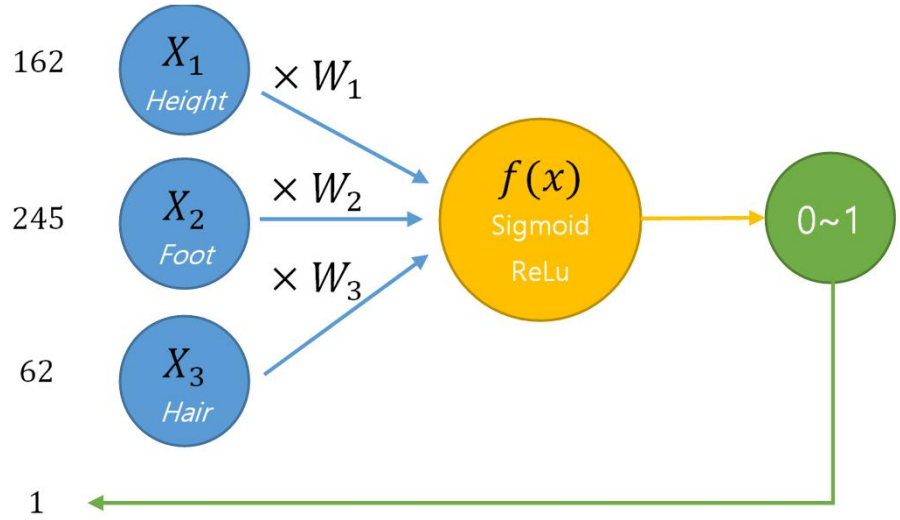
- 주어진 데이터셋을 나열하고 각 데이터에 임의의 가중치를 곱함
- 모든 데이터를 합한 값 x을 활성화 함수 에 입력하면 0~1사이값으로 출력
- 활성화 함수에는 주로 시그모이드 함수와 ReLu 함수가 사용됨



인공신경망

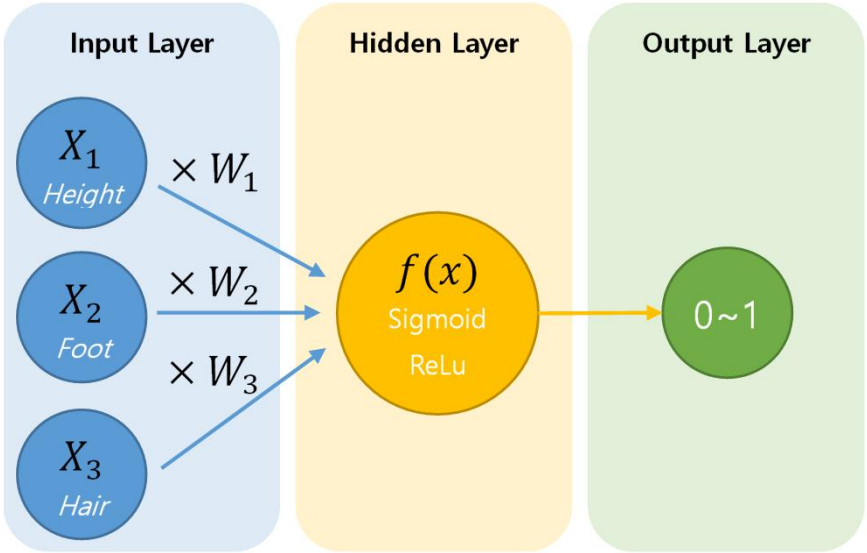
- 활성화 함수의 출력값과 실제값을 비교하고 가중치를 조절하는 과정을 반복

  ->모델의 최적화



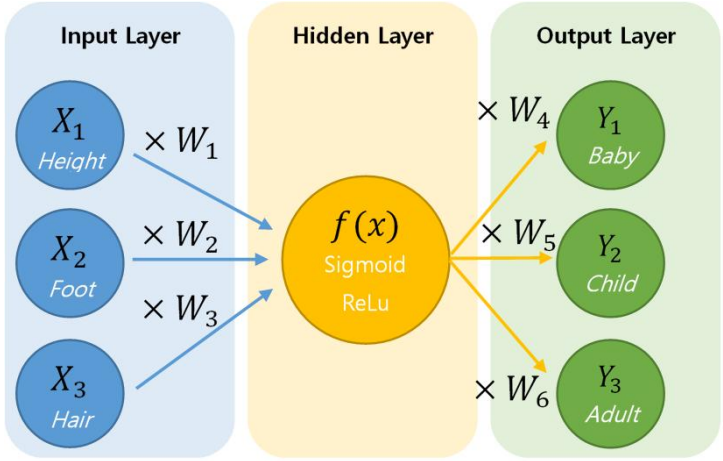
인공신경망

- 기초 인공신경망 모델
  - 역할에 따라 입력층, 은닉층, 출력층으로 구분
  - 노드 : 각 계층에 속해 있는 하나의 변수 또는 함수



인공신경망

- 인공신경망
  - 은닉층과 출력층 노드의 개수 조절 가능
  - 다수의 입력을 받아 다수의 출력 가능
  - 다수의 활성화 함수를 둬서 복잡한 학습 가능

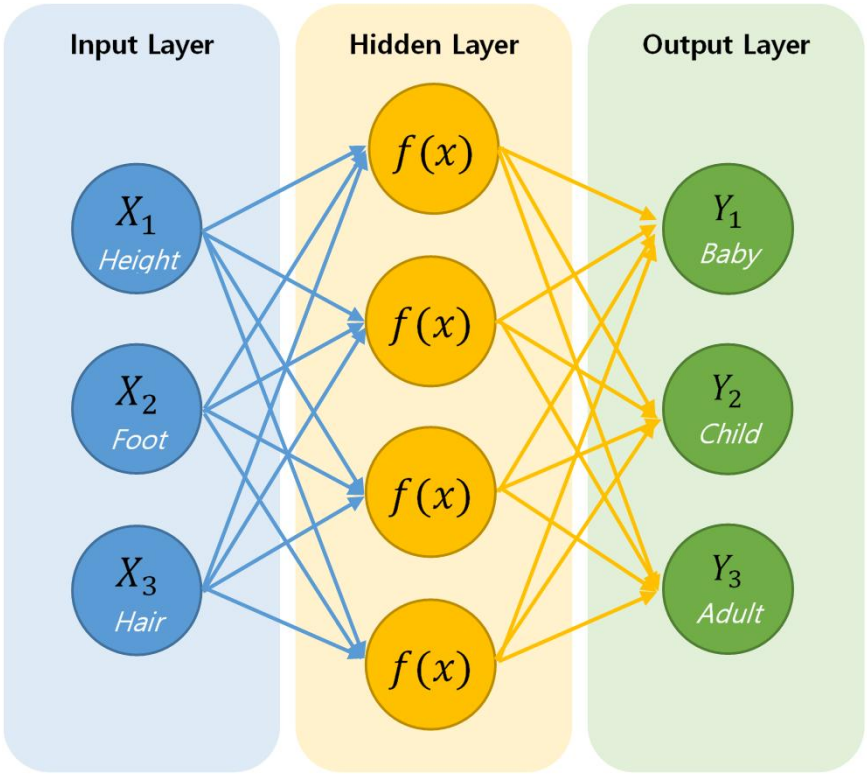


키, 발 크기, 머리카락 길이와 연령층 데이터가 주어졌을 때
복합적 요인에 따른 연령층을 예측하기 위한 인공신경망 모델

인공신경망

- 유아, 어린이, 성인에 대한 출력층 노드를 3개로 늘림
- 은닉 노드 : 출력 노드들에 대해 가중치를 갖음
  - 은닉 노드가 출력에 대한 가중치를 주지 않으면 모든 출력 노드가 같은 값을 갖음
  - 은닉층에서 1개 노드만으로 입력을 연산
    - 입력 데이터의 활용 한계가 낮고 잘못된 학습을 하게 될 가능성이 큼
    - 은닉층의 노드를 여러 개로 늘립니다.

인공신경망

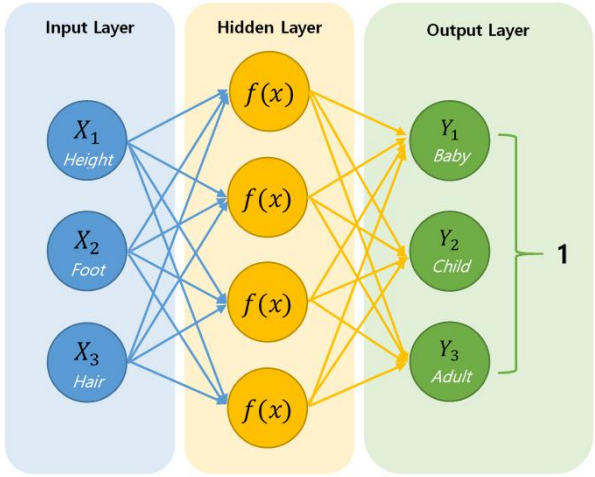

Input Layer
Hidden Layer
Output Layer
X_1
Height
X_2
Foot
X_3
Hair
$f(x)$
$f(x)$
$f(x)$
$f(x)$
Y_1
Baby
Y_2
Child
Y_3
Adult

인공신경망

- 은닉층 노드가 여러 개일 경우
  - 노드들이 각각 받아들이는 입력에 대한 가중치가 다르게 적용
  - 출력에 대한 은닉 노드의 영향력을 분산
  - 한 노드가 잘못된 학습을 하더라도 나머지 3개의 노드에서 정상적인 값을 출력
    - 모델의 오차를 획기적으로 감소가능
- 예시 모델에서는 4개의 은닉 노드의 가중치가 각각 다름
  - 입력-은닉 구간에서 12개의 가중치와 은닉-출력 구간에서 12개의 가중치를 이용

인공신경망

- 은닉층에서 나온 3개의 출력은 실수 범위
  - 합산 범위를 알 수 없음
  - 결과를 활용하기 위해선 정규화 필요
  - 소프트맥스 함수를 이용해 출력 노드들의 합이 1이 되도록 조정



인공신경망

- 연령층을 예측하는 모델 실습
  - 키, 발 크기, 머리카락 길이, 연령층에 대한 데이터를 입력받아 학습
  - 새로운 데이터를 입력하면 연령층을 예측

인공신경망

- ANN 객체의 파라미터
  - input_size: 입력 데이터의 크기. 최하위 차원의 크기 입력 (필수 입력)
  - hidden_size: 은닉층의 노드 수. (기본값: 10)
    - hidden_size를 조절하여 더 복잡한 학습이 가능하지만 크기가 커질수록 학습 속도는 느려짐
  - output_size: 결과 데이터의 크기. 최하위 차원의 크기 입력 (기본값: 1)
  - restore: 최근 모델에 이어서 학습할지에 대한 여부를 Boolean으로 입력 (기본값: False)
  - ckpt_name: 저장 및 불러올 모델 파일의 이름 (기본값: ANN)
  - softmax: 총합이 1이 되도록 할지에 대한 여부를 Boolean으로 입력 (기본값: True)
    - 결과 데이터의 크기가 2 이상일 때, 모델의 예측 결과에 소프트맥스 함수를 적용

인공신경망

- Pop.AI라이브러리 import
- ANN이라는 변수에 생성
  - ANN 객체의 input_size 파라미터를 3, output_size 파라미터를 3으로 설정

```
01:     from pop import AI
02:
03:     ANN = AI.ANN(input_size=3, output_size=3)
```

인공신경망

- ANN 객체의 속성
  - X_data : 입력 데이터
  - Y_data : 입력에 대한 결괏값 데이터
  - 입력할 데이터들은 유아, 어린이, 성인 각 분류별 3개의 데이터를 사용

```
04:    ANN.X_data = [[73,90,1], [62,70,0], [83,100,2], [110,150,7], [139,220,15],
                     [123,190,10], [177,275,7], [159,240,35], [182,280,15]]
05:    ANN.Y_data = [[1,0,0], [1,0,0], [1,0,0], [0,1,0], [0,1,0], [0,1,0], [0,0,1], [0,0,1], [0,0,1]]
```

인공신경망

- ANN 객체의 train() 메소드
  - 인공신경망 학습 시작
  - 파라미터 times : 학습할 횟수 (기본값은 100)
  - 파라미터 print_every : 학습 상황을 몇 번째마다 출력할지를 의미 (기본값은 10)
  - train 메소드를 실행하면 10회마다 인공신경망 모델의 오차 출력

```
06:             ANN.train()
```

인공신경망

- run 메소드
  - 입력에 대한 인공신경망 모델의 세대 분류 결과 출력

```
07:             ANN.run()
```

인공신경망

- 추가 학습
  - train 메소드의 times 파라미터를 1000으로 설정하고 학습

```
01:          ANN.train(times=1000, print_every=100)
```

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

```
02:          ANN.run()
```

인공신경망

- run 메소드의 파라미터로 새로운 데이터를 입력하여 출력 확인

```
03:        ANN.run([[174,270,10], [57,70,1], [140,220,10]])
```

- 학습 횟수를 과도하게 늘려 100,000회를 학습시켰을 때 결과 확인

```
04:        ANN.train(times=100000, print_every=10000)
```

- run 메소드의 파라미터로 [[140,220,10], [140,200,11]]을 입력 출력 확인

```
05:        ANN.run([[140,220,10],[140,200,11]])
```

인공신경망

- ANN 객체의 restore 파라미터를 True로 설정
  - 최근 사용한 학습 모델을 불러와 다시 사용 가능

```
06:          ANN = AI.ANN(input_size=3, output_size=3, restore=True, ckpt_name="model_1")
```

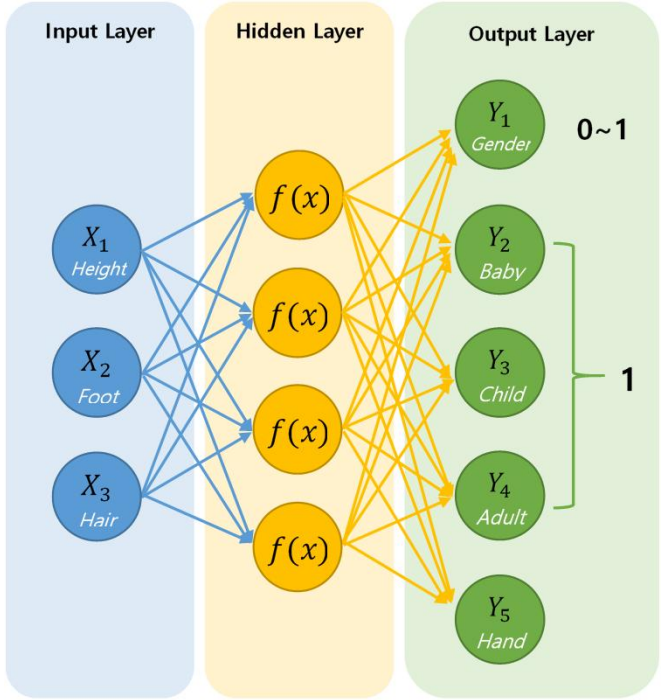
심층신경망

- 심층신경망
  - 복잡한 모델링을 위해 여러 개의 은닉층을 가지고 있는 인공신경망
  - 많은 은닉 노드를 거치기 때문에 더 많은 클래스를 출력 가능
  - 입력 데이터와 출력 데이터 사이에서 비선형적 관계 파악 가능

심층신경망

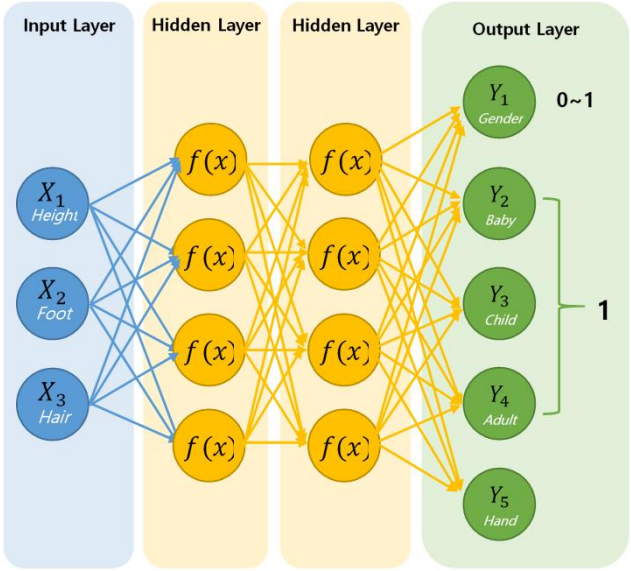
- 복합적 요인에 따른 성별, 연령층, 손 크기 예측 심층신경망 모델
  - 키, 발 크기, 머리카락 길이와 성별, 연령층, 손 크기에 관한 데이터 주어졌을 때
  - 성별 값은 True(1)가 여성, False(0)가 남성일 경우로 설정
  - 인공신경망과 비슷한 구조
  - 출력에 성별과 손 크기만 추가
  - 성별 값은 0~1 사이로 출력되도록 시그모이드 또는 ReLu 함수를 사용
  - 연령층 값들은 그 합이 1이 되도록 소프트맥스 함수를 사용

심층신경망


Input Layer
Hidden Layer
Output Layer
X_1 Height
X_2 Foot
X_3 Hair
$f(x)$
$f(x)$
$f(x)$
$f(x)$
Y_1 Gender
Y_2 Baby
Y_3 Child
Y_4 Adult
Y_5 Hand
0~1
1

심층신경망

- 은닉층을 여러 개로 늘려 연결하면 은닉층 간 다양한 네트워크를 구성
- 복잡한 모델링에 적합



심층신경망

- 최적화에 성공한 심층신경망 모델
  - 새로운 복합 데이터가 주어졌을 때 여성일 확률, 연령층, 손 크기 예측

심층신경망 실습

- 퍼셉트론 실습의 데이터 활용
  - 퍼셉트론과 비교해 얼마나 적은 시간으로 더 효율적인 학습을 하는지에 비교

심층신경망 실습

- DNN 객체의 파라미터
  - input_size: 입력 데이터의 크기. 최하위 차원의 크기를 입력 (필수 입력)
  - hidden_size: 은닉층의 노드 수 (기본값: 10)
    - hidden_size를 조절하여 더 복잡한 학습이 가능하지만 크기가 커질수록 학습 속도는 느려집니다.
  - output_size: 결과 데이터의 크기. 최하위 차원의 크기를 입력 (기본값: 1)
  - layer_level: 은닉층의 수 (기본값: 3)
    - layer_level을 조절하여 더 깊은 신경망을 만들어 복잡한 학습이 가능
    - 은닉층 차원이 커질수록 학습 속도는 느려지고 과적합 현상이 쉽게 발생할 가능성이 커짐
  - restore: 최근 모델에 이어서 학습할지에 대한 여부를 Boolean으로 입력 (기본값: False)
  - ckpt_name: 저장 및 불러올 모델 파일의 이름 (기본값: DNN)
  - softmax: 총합이 1이 되도록 할지에 대한 여부를 Boolean으로 입력 (기본값: True)
    - 결과 데이터의 크기가 2 이상일 때, 모델의 예측 결과에 소프트맥스 함수를 적용

심층신경망 실습

- Pop.AI라이브러리 import
- DNN이라는 변수에 생성
  - input_size 파라미터를 3, output_size 파라미터를 2, layer_level 파라미터를 5로 설정

```
01:     from pop import AI
02:
03:     DNN=AI.DNN(input_size=3, output_size=2,layer_level=5)
```

심층신경망 실습

- DNN 객체의 속성
  - X_data : 입력 데이터
  - Y_data : 입력에 대한 결괏값 데이터
- 입력할 데이터들은 키, 발 크기, 머리카락 길이 데이터를 사용
- 결과 데이터로는 성별 데이터를 사용
- 성별 데이터는 [1, 0] 이면 여성, [0, 1] 이면 남성으로 설정

```
04:    DNN.X_data = [[173,270,17], [171,275,27], [162,245,42], [187,280,12], [157,230,47],
                     [169,265,30], [177,270,5], [159,250,32], [182,275,0]]
05:    DNN.Y_data = [[0,1],[1,0],[1,0],[0,1],[1,0],[0,1],[0,1],[1,0],[0,1]]
```

심층신경망 실습

- DNN 객체의 train() 메소드
  - 심층신경망 학습 시작
  - 파라미터 times : 학습할 횟수 (기본값은 100)
  - 파라미터 print_every : 학습 상황을 몇 번째마다 출력할지를 의미 (기본값은 10)
  - train 메소드를 실행하면 10회마다 심층신경망 모델의 오차 출력

```
06:          DNN.train()
```

심층신경망

- DNN 객체의 run 메소드
  - 학습된 모델 사용 가능
  - 파라미터 inputs
  - 기본값은 X_data 사용
  - 입력에 대한 심층신경망 모델의 성별 분류 결과 출력

```
07:             DNN.run()
```

심층신경망

전체 코드

```
01:     from pop import AI
02:
03:     DNN=AI.DNN(input_size=3, output_size=2,layer_level=5)
04:     DNN.X_data = [[173,270,17], [171,275,27], [162,245,42], [187,280,12],
                      [157,230,47], [169,265,30], [177,270,5], [159,250,32], [182,275,0]]
05:     DNN.Y_data = [[0,1],[1,0],[1,0],[0,1],[1,0],[0,1],[0,1],[1,0],[0,1]]
06:     DNN.train()
07:     DNN.run()
```

심층신경망

- run 메소드의 파라미터로 새로운 데이터를 입력하여 출력 확인

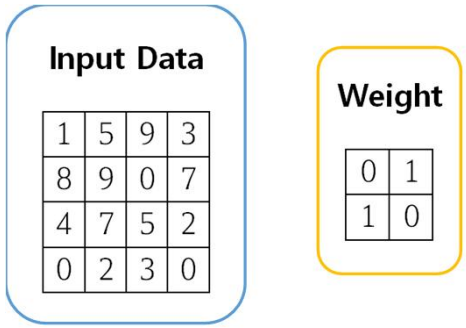
```
08:             DNN.run([[174,265,6], [152,230,30], [162,255,10]])
```

합성곱 신경망

- 합성곱 신경망
  - 입력 데이터의 양이 많을 때 입력 데이터를 압축하여 모델링하는 인공신경망
  - 필요한 수준으로 압축하여 성능과 속도를 모두 확보 가능
  - 입력 데이터를 압축하는 과정에서 슬라이드 윈도우 방식 사용
    - 슬라이드 윈도우 방식 : 입력 데이터보다 작은 사이즈의 배열을 순서대로 옮겨가며 축소연산
    - 필터링 또는 마스킹 : 옮겨가며 연산하는 과정
    - 커널 또는 윈도우 : 필터링에 사용되는 배열

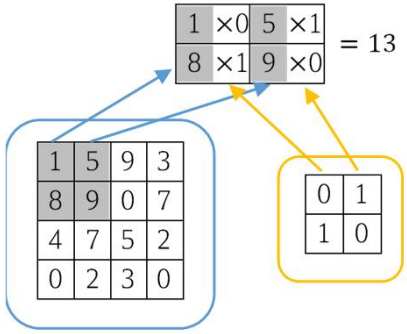
합성곱 신경망

- 4x4 텐서 데이터가 주어졌을 때 임의로 2x2 가중치 커널을 생성



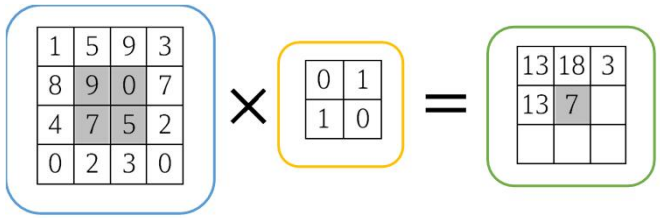
합성곱 신경망

- 데이터의 왼쪽 위부터 값을 곱한 후 모두 합해 1개 값으로 반환
- 옆으로 한 칸 옮겨 반복
- 이 과정을 필터링 또는 마스킹이라고 함

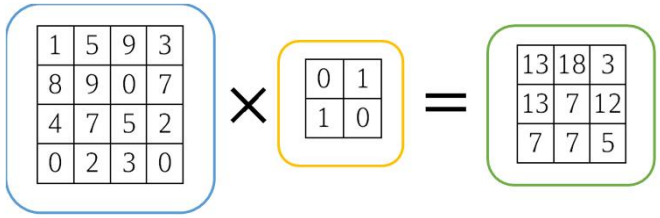


합성곱 신경망

- 필터링 중 중간 과정

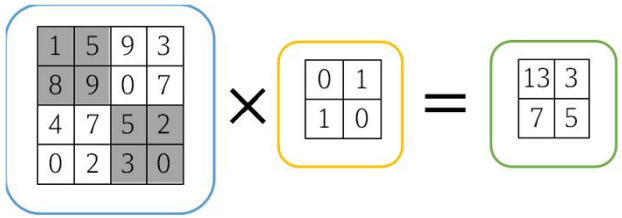


- 필터링이 완료되면 3x3 사이즈의 압축 데이터가 반환
  - 이 데이터를 심층신경망에 연결하면 보다 빠른 속도를 기대할 수 있음



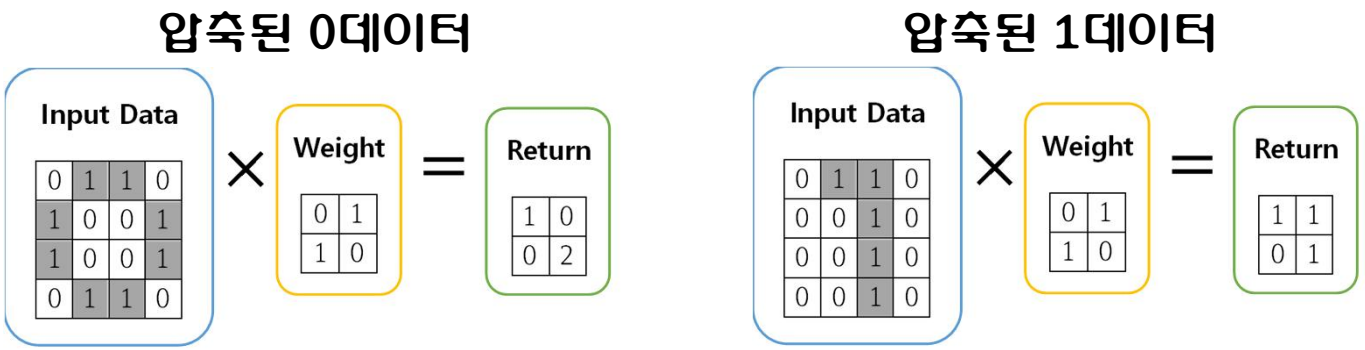
합성곱 신경망

- 스트라이드 : 커널이 슬라이드 할 때, 한 번에 넘어갈 칸수
  - 스트라이드를 2로 설정하면 2칸씩 건너뛰어 필터링 함



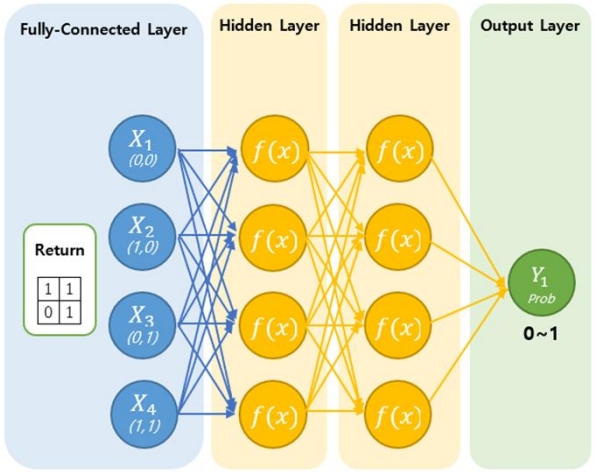
합성곱 신경망

- 0 또는 1을 그린 4x4 이미지를 분류하는 합성곱 신경망 모델
  - 데이터에서 흰색을 0, 검은색을 1로 설정
  - 스트라이드가 2인 2x2 커널을 이용해 압축 데이터를 구함



합성곱 신경망

- 반환된 값을 나열하여 심층신경망에 입력
  - 평탄화 : 압축 데이터를 나열하는 과정
  - 완전 연결 계층 : 압축 데이터와 심층신경망을 연결하는 은닉층



합성곱 신경망

- 숫자 식별 모델 실습
  - 손글씨로 쓴 숫자 이미지를 입력받아 학습
  - 새로운 이미지 데이터 입력시 숫자 식별

합성곱 신경망 실습

- CNN 객체의 파라미터
  - input_size: 2차원 데이터를 입력 데이터로 사용. 리스트로 입력 (기본값: [28, 28])
  - input_level: RGB 이미지인 경우 3, 흑백 이미지인 경우 1을 입력 (기본값: 1)
  - kernel_size: 합성곱 계층에서 사용할 커널의 크기. 리스트로 입력 (기본값: [3, 3])
  - kernel_count: 하나의 입력 데이터에 사용할 가중치 커널의 개수 (기본값: 32)
    - 커널의 개수가 많아지면 더 다양한 특징을 찾아내지만, 속도는 급격히 느려집니다.
  - strides: 커널의 스트라이드를 설정. 리스트로 입력 (기본값: [1, 1])
  - hidden_size: 은닉층의 노드 수 (기본값: 128)
    - hidden_size를 조절하여 더 복잡한 학습이 가능하지만 크기가 커질수록 학습 속도는 느려짐

합성곱 신경망 실습

- output_size: 결과 데이터의 크기. 최하위 차원의 크기를 입력 (기본값: 1)
- conv_level: 합성곱 계층의 개수를 설정. 값이 커질수록 합성곱 계층은 깊어짐 (기본값: 2)
- layer_level: 은닉층의 수 (기본값: 1)
  - layer_level을 조절하여 더 깊은 신경망을 만들어 복잡한 학습이 가능
  - 은닉층 차원이 커질수록 학습 속도는 느려지고 과적합 현상이 쉽게 발생할 가능성 커짐
- restore: 최근 모델에 이어서 학습할지에 대한 여부를 Boolean으로 입력 (기본값: False)
- ckpt_name: 저장 및 불러올 모델 파일의 이름 (기본값: CNN)
- softmax: 총합이 1이 되도록 할지에 대한 여부를 Boolean으로 입력 (기본값: True)
  - 결과 데이터의 크기가 2 이상일 때, 모델의 예측 결과에 소프트맥스 함수를 적용

합성곱 신경망 실습

- Pop.AI라이브러리 import
- CNN이라는 변수에 생성
  - output_size 파라미터를 10으로 설정

```
01:     from pop import AI
02:
03:     CNN = AI.CNN(output_size=10)
```

합성곱 신경망 실습

- MNIST 데이터셋을 다운로드 받아 사용
  - CNN 객체 속성에 X_data와 Y_data가 있지만 이미지 데이터는 직접 입력하기에 부적합
- MNIST는 손으로 쓴 숫자 이미지 데이터 베이스
- CNN 객체의 load_MNIST() 메소드
  - 자동으로 데이터셋을 불러와 X_data와 Y_data에 입력
  - 만약 장치에 데이터셋이 없다면 자동으로 다운로드 (인터넷 연결이 필요)

```
04:          CNN.load_MNIST()
```

합성곱 신경망 실습

- CNN 객체의 show_img 메소드
  - 이미지를 출력
  - 파라미터 input : x, y, color를 담는 3차원 배열 입력

```
05:         CNN.show_img(CNN.X_data[501])
```

합성곱 신경망 실습

- CNN 객체의 train() 메소드
  - 합성곱 신경망 학습 시작
  - 파라미터 times : 학습할 횟수 (기본값은 100)
  - 파라미터 print_every : 학습 상황을 몇 번째마다 출력할지를 의미 (기본값은 10)
  - Train 메소드를 실행하면 10회마다 합성곱 신경망 모델의 오차 출력

```
06:          CNN.train()
```

합성곱 신경망 실습

- CNN 객체의 run 메소드
  - 학습된 모델 사용 가능
  - 파라미터 inputs
  - 기본값은 X_data 사용
    - X_data의 크기가 매우 크므로 X_data의 일부 입력
  - run 메소드를 실행하면 입력에 사용된 이미지를 표시
    - 입력에 대한 합성곱 신경망 모델의 숫자 분류 결과를 출력 배열의 순서대로 0~9일 확률을 출력

```
07:         X = [CNN.X_data[10]]
08:         CNN.run(X)
```

합성곱 신경망 실습

전체 코드

```
01:     from pop import AI
02:
03:     CNN = AI.CNN(output_size=10)
04:     CNN.load_MNIST()
05:
06:     CNN.train()
07:
08:     X = [CNN.X_data[10]]
09:     CNN.run(X)
```

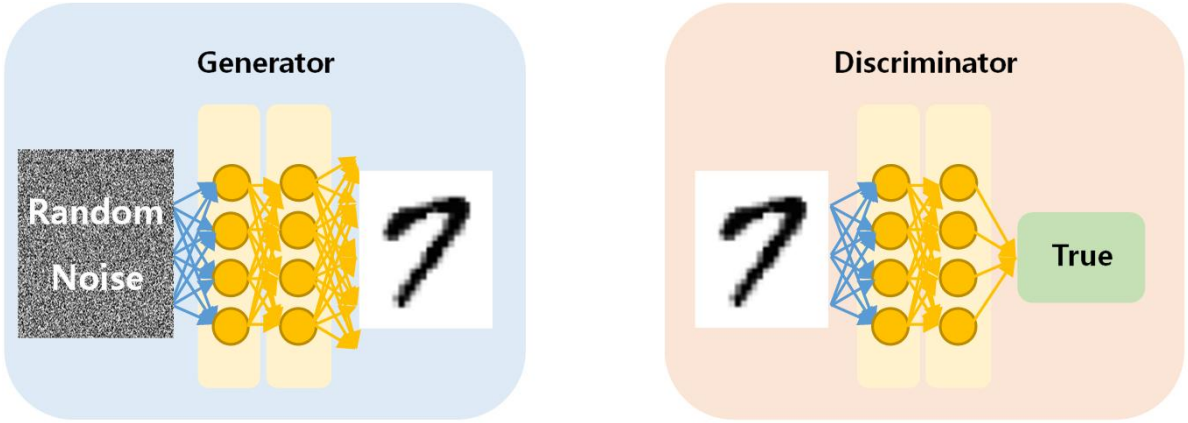
restore 파라미터를 True로 설정

- 최근 사용한 학습 모델을 불러와 다시 사용 가능

```
01:     CNN=AI.CNN(output_size=10, restore=True)
```

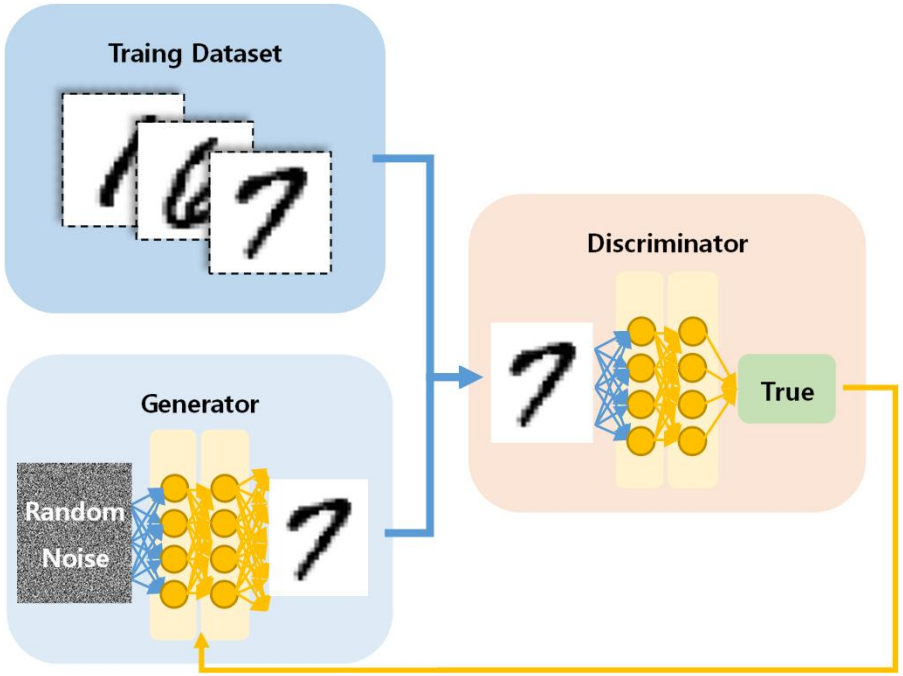
생산적 적대 신경망

- 판별 모델과 위조 모델을 학습시켜 고수준의 위조 데이터를 생성
  - 생성자 : 위조 데이터를 생성하는 인공신경망 모델
  - 판별자 : 데이터의 진위를 판별하는 인공신경망 모델



생산적 적대 신경망

- 위조와 판별을 반복하며 두 모델을 계속해서 발전시킴



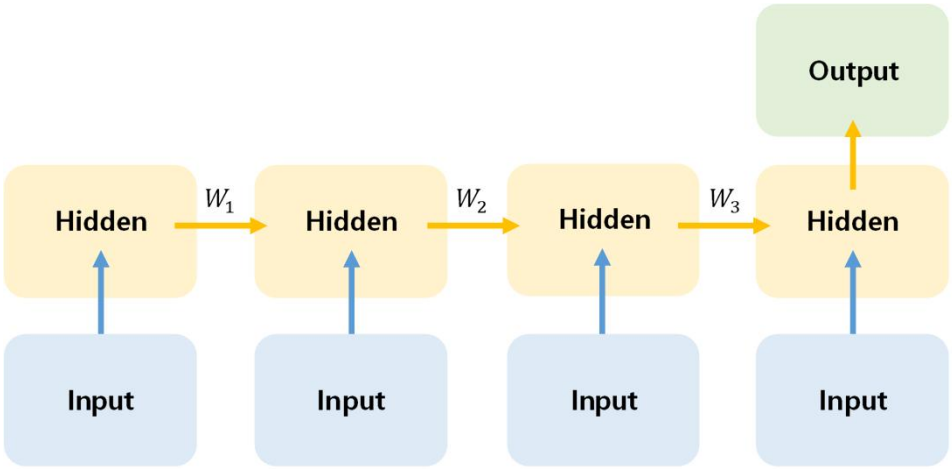
생산적 적대 신경망

- 성공적인 생산적 적대 신경망 모델 학습
  - 두 모델의 수준이 비슷하도록 학습율 조절 및 유지
  - 생성자 모델이 너무 뛰어날 경우
    - 판별자 모델은 진짜 데이터를 가짜라고 판별하게 학습할 가능성이 큼
  - 판별자 모델이 너무 뛰어날 경우
    - 생성자 모델은 잘 만든 위조 데이터도 False 피드백만 받음
    - 진짜 같은 데이터를 생성하기 위한 가중치를 찾을 수 없음

순환신경망

- 과거 입력 데이터를 이후 처리에도 반영하여 사용하는 인공신경망
  - 은닉층의 출력을 다음 스텝에서 입력 데이터와 함께 다시 입력
- 기존 딥러닝 알고리즘들의 최대 결점은 시간 개념이 없다는 것
  - 여러 데이터셋들 간의 관계나 순서를 반영하지 않음
- 순환신경망은 문장의 전후 관계, 이미지의 사물 관계를 파악 가능
  - 과거 데이터값을 요약하여 갖고 있음

순환신경망


Output
Hidden
W_1
Hidden
W_2
Hidden
W_3
Hidden
Input
Input
Input
Input

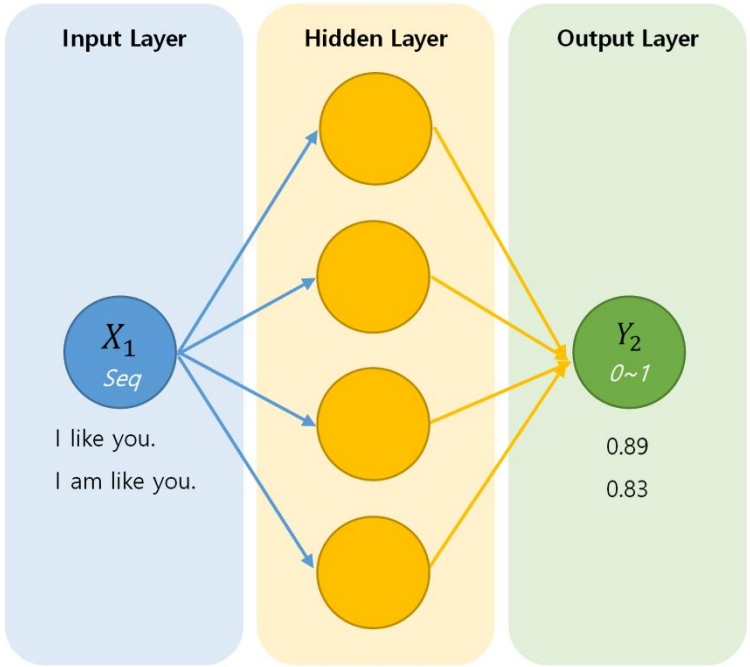
순환신경망

- ‘Like’의 동사와 전치사를 구분하는 순환신경망 모델과 심층신경망 모델 비교
  - 다음과 같은 두 데이터셋이 입력으로 주어졌을 때
  - 전치사를 True(1), 동사를 False(0)이라고 설정

문장	구분
I like you.	0
I am like you.	1

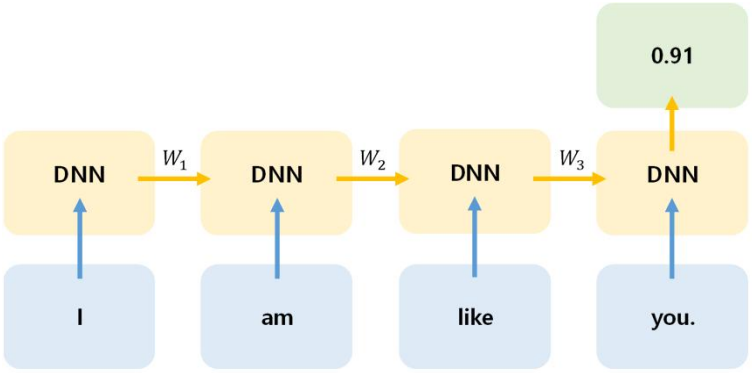
순환신경망

- 심층신경망 모델에서는 문장 전체를 입력으로 받아 처리

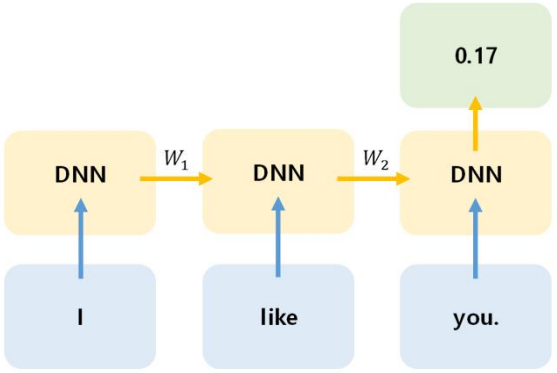


순환신경망

- 심층신경망은 전체적인 형태가 매우 비슷한 두 데이터셋을 똑같이 구분
  - 각 단어의 순서와 상호 관계를 알 수 없음
- 순환신경망은 'am'의 문장 결정력을 구분 가능
  - 문장에 포함된 단어를 순서대로 입력받고 다음 처리에 연결



순환신경망의 'I am like you' 구분



순환신경망의 'I like you' 구분

내용 정리

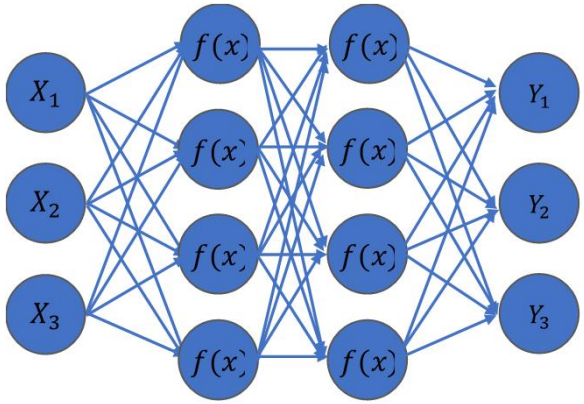
- 딥러닝 : 인공신경망 이론을 기반으로 설계된 머신러닝 기법
- 퍼셉트론 : 다수의 데이터로 하나의 결과를 출력하는 복합 논리 회로
  - 각 요인이 결과에 미치는 영향을 분석할 때 사용
- 인공신경망 : 퍼셉트론을 기반으로 고안
  - 다수의 데이터로 하나 이상의 결과를 출력하는 알고리즘
- 심층신경망 : 여러 은닉층을 가지고 있는 인공신경망
  - 입력과 결과의 복잡한 관계 분석

내용 정리

- 합성곱 신경망 : 대량의 입력 데이터를 압축하여 분석하는 인공신경망
- 생산적 적대 신경망 : 데이터의 진위를 구별하는 판별 모델과 가짜 데이터를 생성하는 위조 모델을 경쟁시켜 모방 데이터를 생성하도록 모델링하는 인공신경망
- 순환신경망 : 이전 입력을 이후 처리에도 반영하여 데이터의 연속 관계를 분석하는 인공신경망

연습문제

- 문제 39. 다음 신경망 그림을 보고 질문에 답해보세요.



  - A. 신경망의 입력과 출력의 크기를 답해보세요.
  - B. 신경망의 은닉층 수를 답해보세요.

연습문제

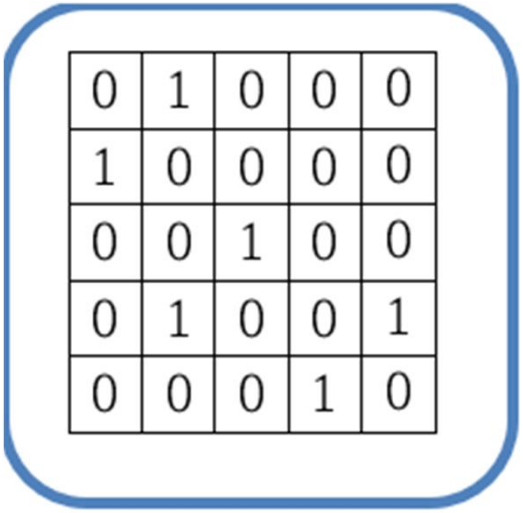
- 문제 40. 다음 문장들을 읽고 빈 칸을 채워보세요.
  - A.딥러닝은 [ ]을 기반으로 설계된 머신러닝 기법들이다.
  - B.퍼셉트론은 [ ]를 [ ]로 출력하는 복합 논리 회로이다.
  - C.심층신경망은 여러개의 [ ]층을 가지고 있다.
  - D.합성곱 신경망은 [ ] 방식으로 대용량 데이터를 압축한다.

연습문제

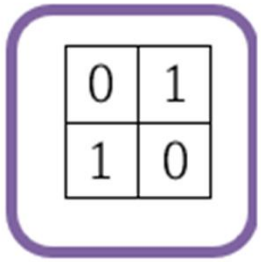
- E.합성곱 신경망의 데이터 압축 과정 중 필터링에 사용되는 배열을 [ ]이라 하고, 이것이 한 번에 이동할 칸 수를 [ ]라고 합니다.
- F.생산적 적대 신경망은 [ ]와 [ ]를 경쟁시키는 방법으로 학습하여 위조 데이터를 생성합니다.

연습문제

- 문제 41. 다음 그림의 배열(좌)과 커널(우)을 이용해 질문에 답해보세요.



0	1	0	0	0
1	0	0	0	0
0	0	1	0	0
0	1	0	0	1
0	0	0	1	0



0	1
1	0

연습문제

- A. 스트라이드가 3 (또는 [3, 3]) 일 때 출력을 작성하세요.
- B. A의 출력이 완전 연결 계층에 입력될 때 완전 연결 계층의 노드 수를 답하세요.

연습문제

- 문제 42. 다음 코드는 Pop.AI 라이브러리를 이용하여 2개의 입력을 받아 2개의 출력을 하는 심층신경망를 구현한 코드입니다. 질문을 읽고 답해보세요.

```
01:     from pop import AI
02:
03:     DNN = AI.DNN(input_size=2, output_size=2, softmax=True)
04:
05:     DNN.X_data = X
06:     DNN.Y_data = Y
07:
08:     DNN.train(times=1000, print_every=100)
09:     DNN.run()
```

연습문제

- A.두 입력에 대해 크기를 비교하여 큰 쪽을 1, 작은 쪽을 0으로 출력할 수 있도록 빈 칸 X, Y에 들어갈 학습 데이터셋을 작성해보세요.
- B.A에서 작성한 데이터셋으로 학습시키고, [1,0]과 [-1,0]을 입력했을 때 출력을 작성하세요.

연습문제

- 문제 43. 다음 코드는 AIoT AutoCar의 카메라 영상을 BGR 값으로 받아 이미지로 출력하는 코드입니다. 이 코드를 응용해 다음 문제들을 해결해보세요.

```
01:     from pop import Camera, Util
02:
03:     cam = Camera(width=50, height=50)
04:
05:     value = Camera.value
06:     Util.imshow("Title", value)
```

연습문제

- A. 다음 코드를 실행해 카메라 BGR 데이터의 형태를 확인해보세요.

```
07:             print(value.shape)
```

- B. 다음 사진처럼 손바닥을 카메라에 비출 때 BGR 값을 받아 이미지로 확인해보세요.

연습문제

- C. 빈 배열을 생성하고, A와 같은 방법으로 다양한 손바닥 데이터를 20개 이상 추가해보세요.
- D. C의 배열에 손바닥이 없는 카메라 데이터를 20개 이상 추가해보세요.
- E. 빈 배열을 생성하고, 손바닥이 있는 사진과 없는 사진을 각각 1과 0으로 하여 C에서 추가한 개수만큼 1을 추가하고 D에서 추가한 개수만큼 0을 추가하세요.

연습문제

- F. Pop.AI 라이브러리의 CNN 클래스를 사용해 D에서 완성된 배열을 X 데이터, E에서 완성된 배열을 Y 데이터로 하는 합성곱 신경망 코드를 작성하세요.
- G. F에서 작성한 합성곱 신경망 모델을 학습시켜 손실율을 최소화해보세요.
- H. G에서 학습한 모델에 새로운 카메라 BGR 데이터를 입력하여 결과를 확인해보세요.