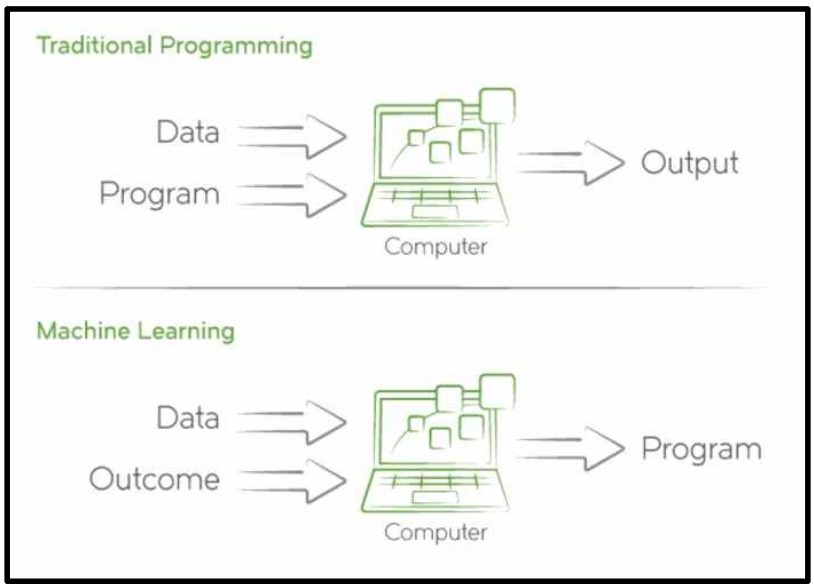
■ 간단한 예측자

1) 머신러닝 / 딥러닝과 다른 프로그래밍의 차이는?



- 기존 프로그래밍 '알고리즘'을 만들고 입력값을 넣어서 '출력'을 얻음
- 머신러닝은 '입력'과 '출력'을 빅데이터로 넣어줘서 학습시켜 '알고리즘'을 만들어 냄

2) 가장 기초적인 신경망의 이해(간단한 예측자 이해)

- 선형 관계의 예측자 만들기
- 킬로미터를 마일로 변환하는 예측자

마일 = 킬로미터 x c(c는 상수)

- 결국 상수 C를 구하는 방법
- 빅데이터(사실 스몰데이터)가 필요함

km	mi
0.1	0.0621371192
0.2	0.1242742384
0.3	0.1864113577
0.4	0.2485484769
0.5	0.3106855961
0.6	0.3728227153
0.7	0.4349598346
0.8	0.4970969538
0.9	0.5592340730
1	0.6213711922
1.1	0.6835083115
1.2	0.7456454307

- 만약 0.5를 대입해본다면

100 -> 100 x 0.5 -> 50
오차는 12.137...

- 만약 0.6을 대입한다면

100 -> 100 x 0.6 -> 60
오차는 2.137...

- 만약 0.7을 대입한다면

100 -> 100 x 0.7 -> 70
오차는 -7.863...

- 오버슈팅 발생
- 다시 0.61을 대입

100 -> 100 x 0.61 -> 61
오차는 1.137...

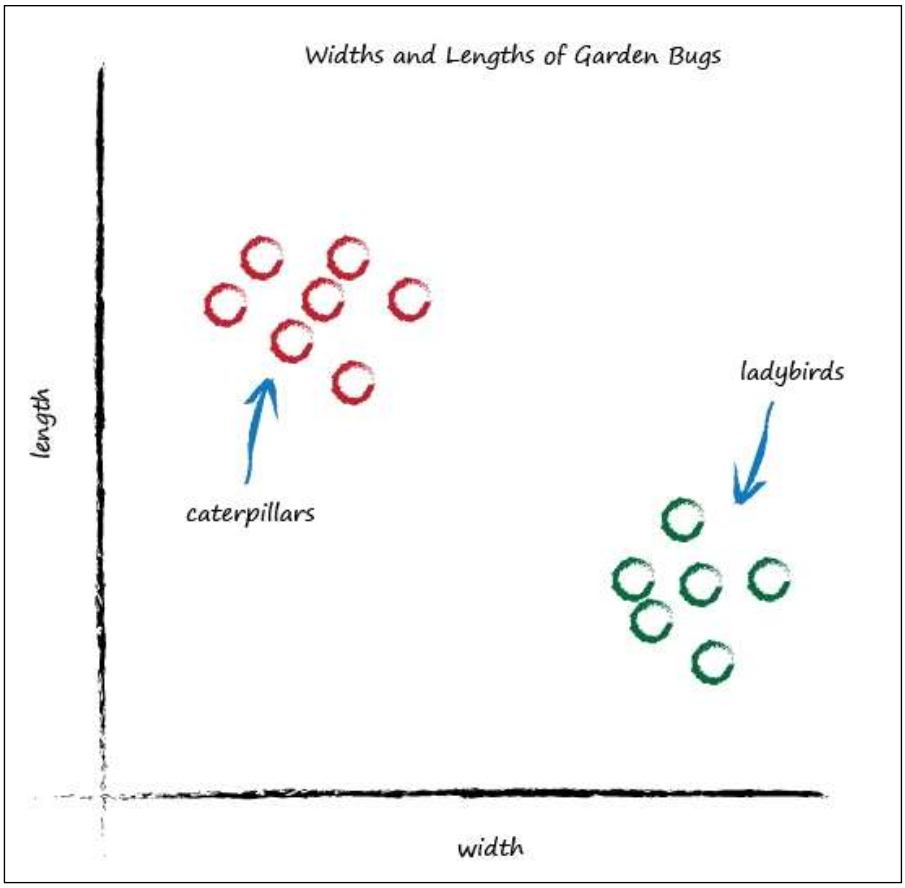
- 꽤 정확한 값을 얻음

3) 위의 방식으로 인공 신경망을 학습시키는 핵심과정 : 강화학습

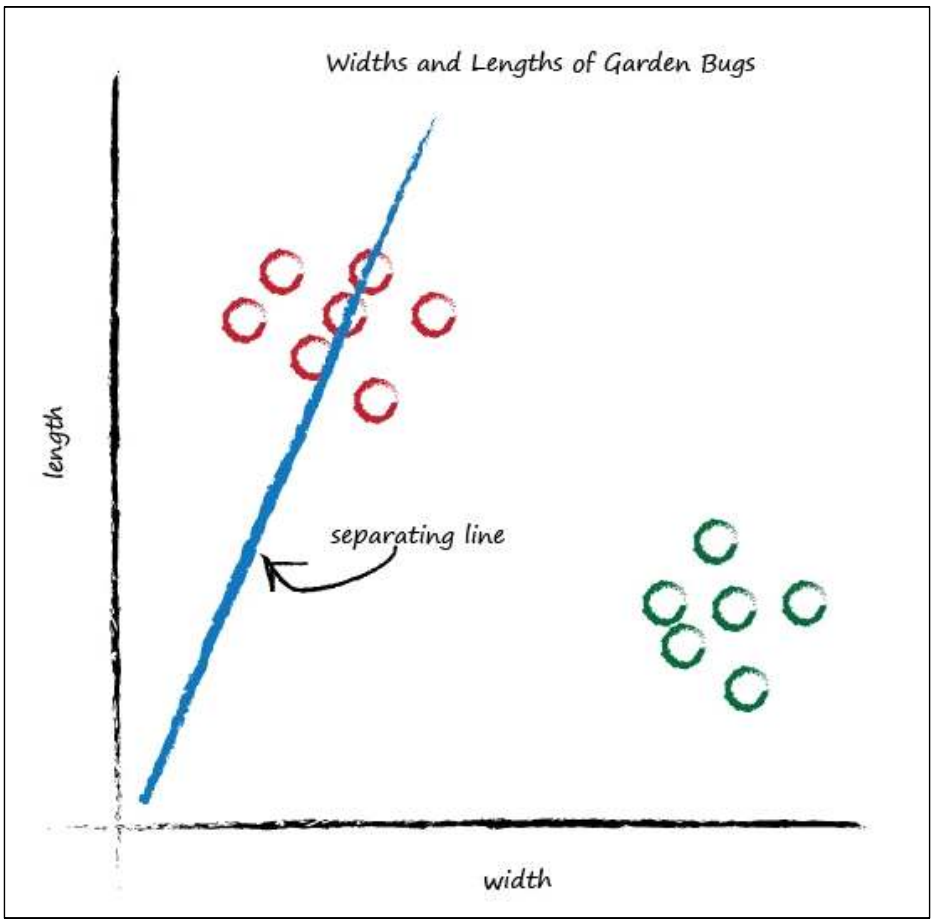
- 반복을 통해 점점 더 정답에 가까운 값을 스스로 얻음

■ 예측자의 이해

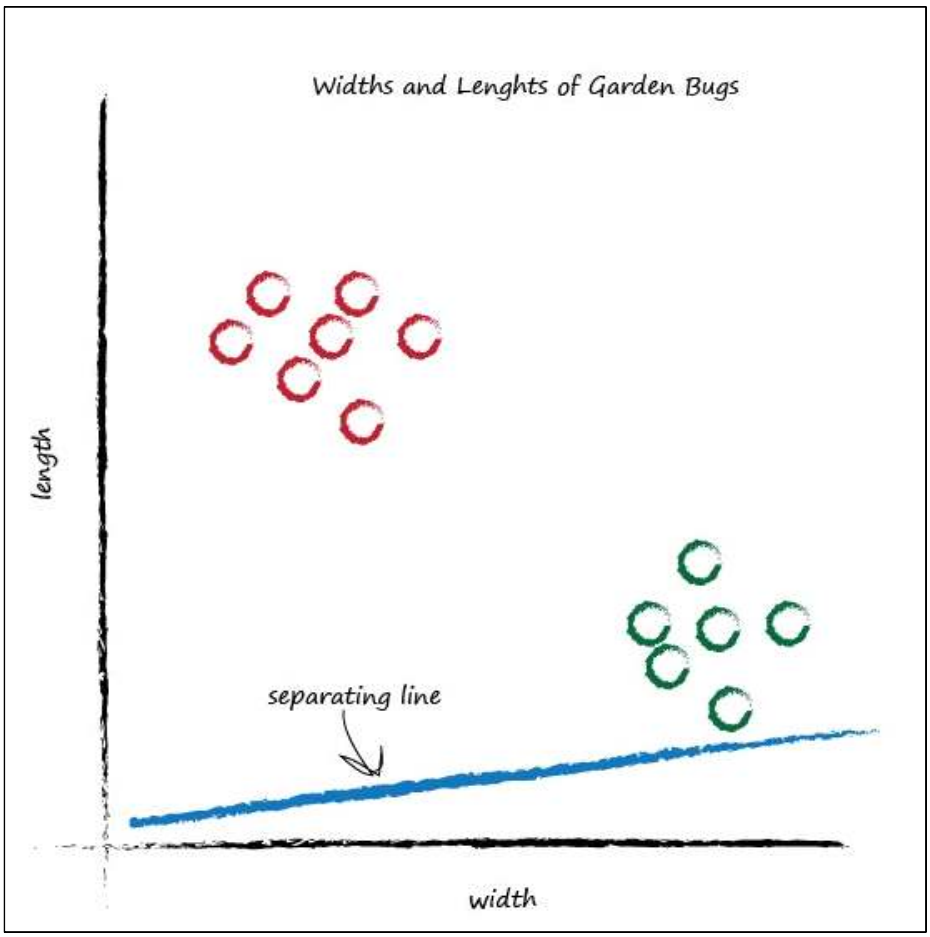
1) 곤충의 폭과 길이에 따른 그래프



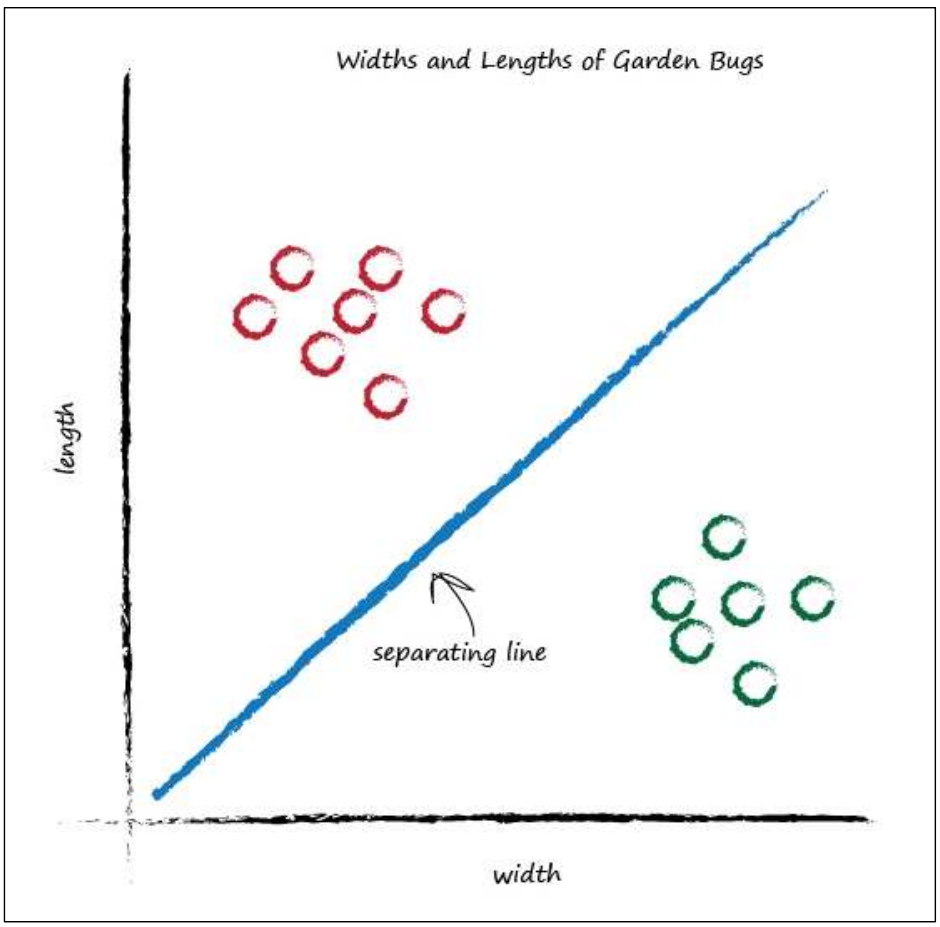
- 선을 그어보면 이 직선이 '분류'하는 용도로 사용이 가능함



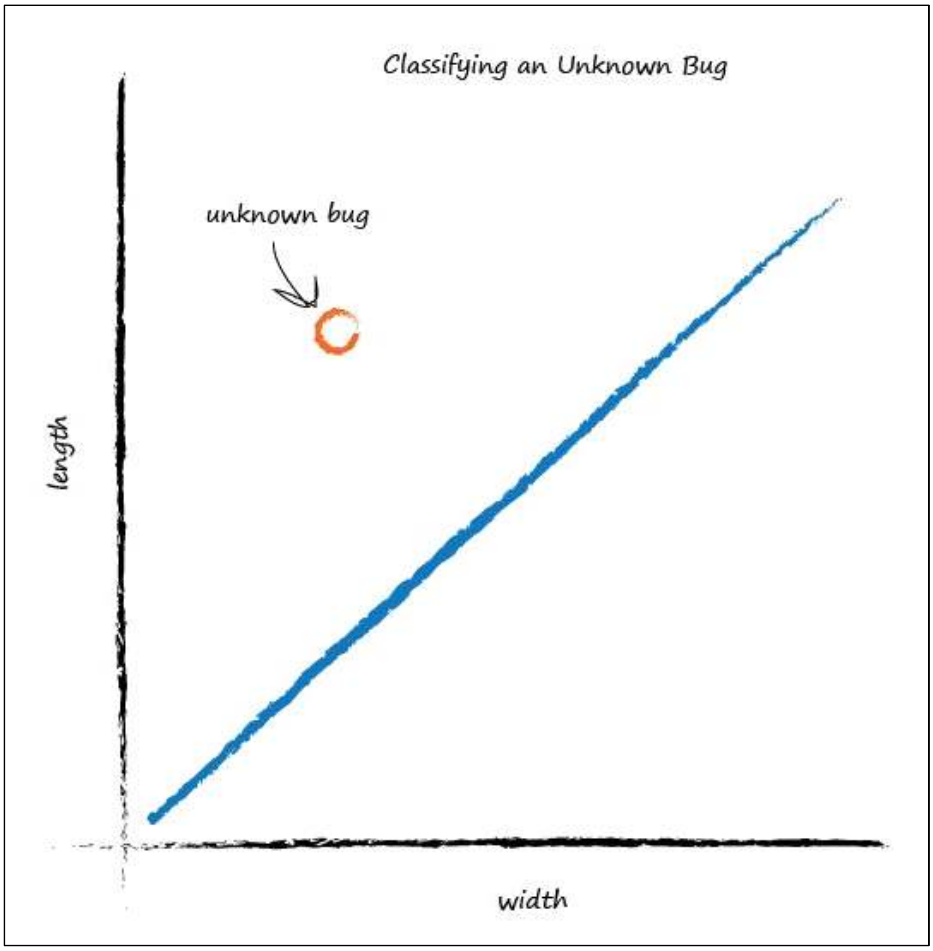
- 기울기를 다르게 변형



- 다시 한번 기울기를 변경



2) 이 직선은 곤충의 분류자로 이용가능

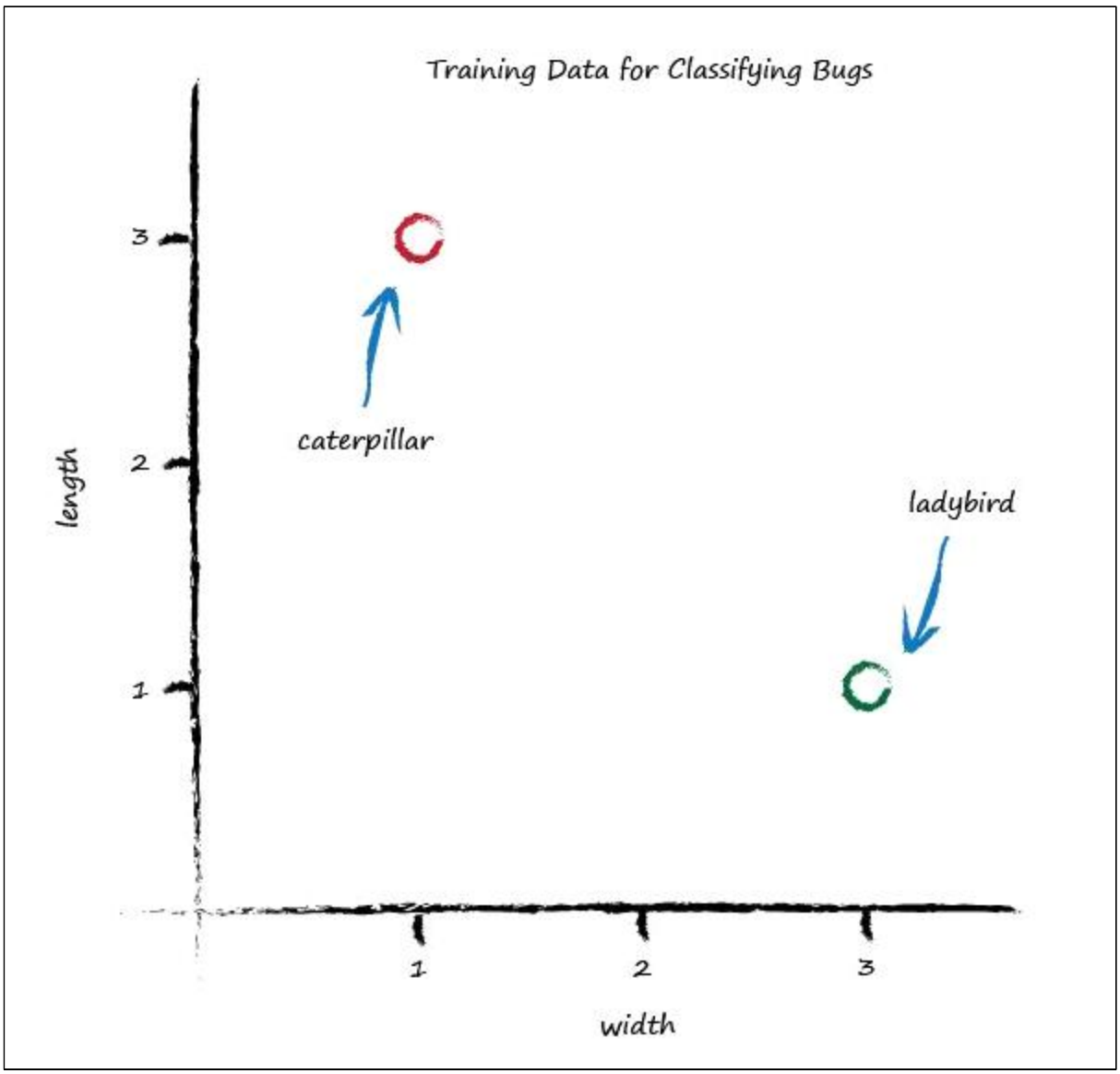


- 예측자의 선형함수가 미지의 데이터를 어떻게 분류했는지 알아보았음
- 하지만 어떻게 이 직선의 기울기를 구해야 할 것인가?
  어떻게 하면 신경망을 학습시킬 수 있을까?

■ 분류자 학습시키기

1) 정말 간단한 '학습데이터'를 예시로

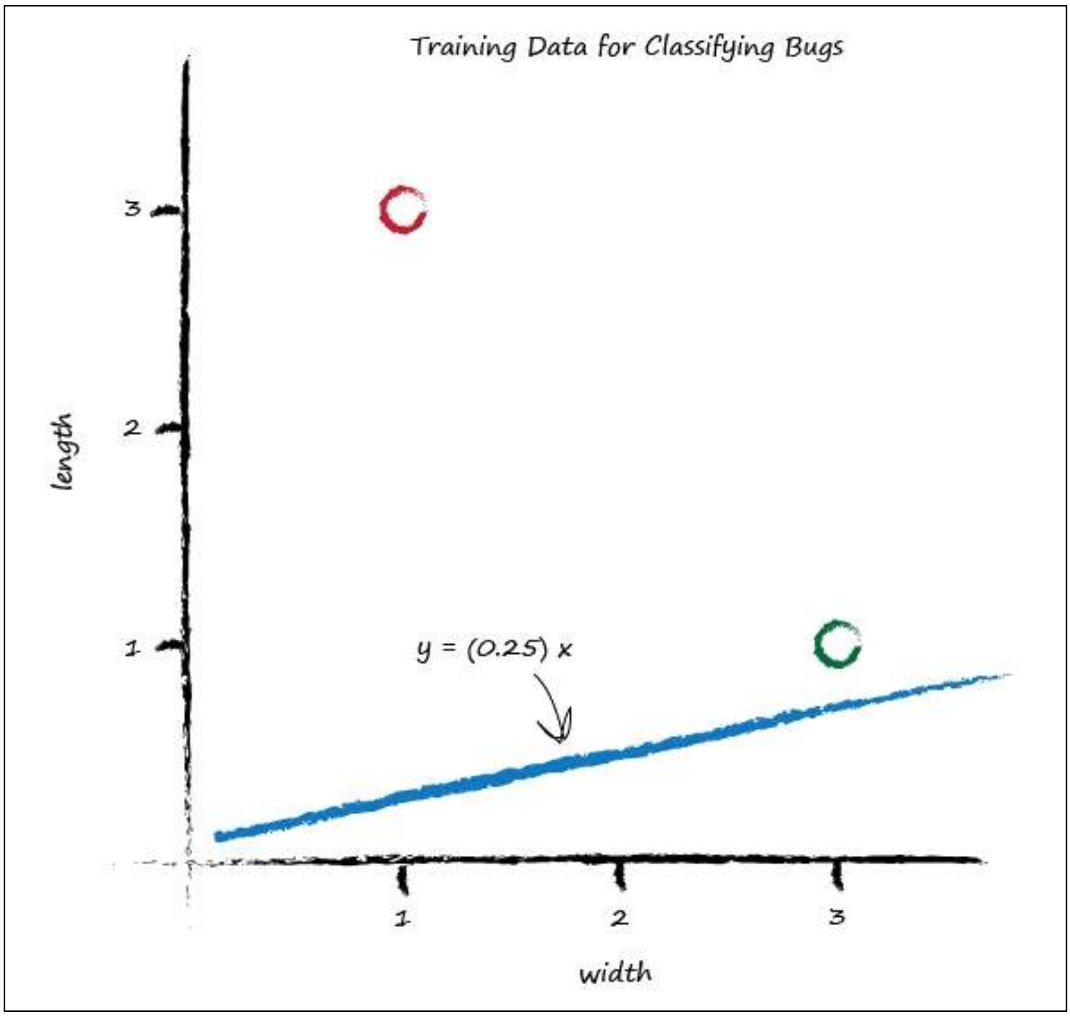
예 제	폭	길 이	곤 충
1	3.0	1.0	무당벌레
2	1.0	3.0	애벌레



2) 여기에 선형함수를 이용해 직선을 그림

- 원래 선형함수의 방정식은 'y = Ax + B'
- 여기서 B가 0이면 원점을 지나는 방정식

3) 우선 A = 0.25라고 학습함

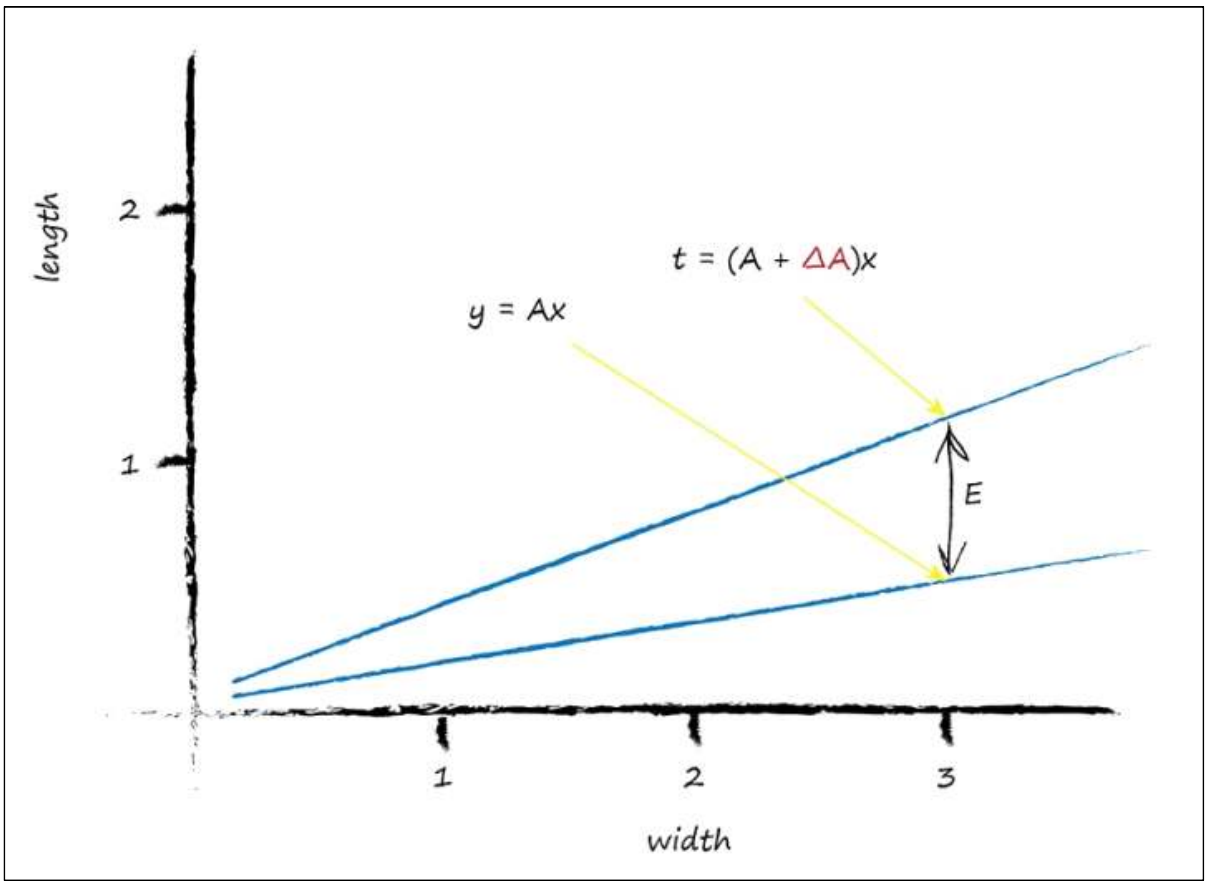


- 한쪽에는 2개가 다른 한쪽에는 전혀 없음
- 0.25는 전혀 좋은 분류자(예측자)가 되지 못함
- 여기서 학습데이터를 활용하면

```
Y = 0.25 * 3.0
즉 Y는 0.75가 예상됨 : 원래의 값인 1보다 작음
```

- 오차 발생
- 사실 Y는 1보다는 무조건 커야 함(만약 1이면 무당벌레와 겹침)

- 그래서 (대충) 가장 작은 수는 1.1이라고 가정한다면



- 1.1 - 0.73 = 0.35

- 이 0.35가 오차(그래프에서 E라고 표시된 숫자가 바로 오차)

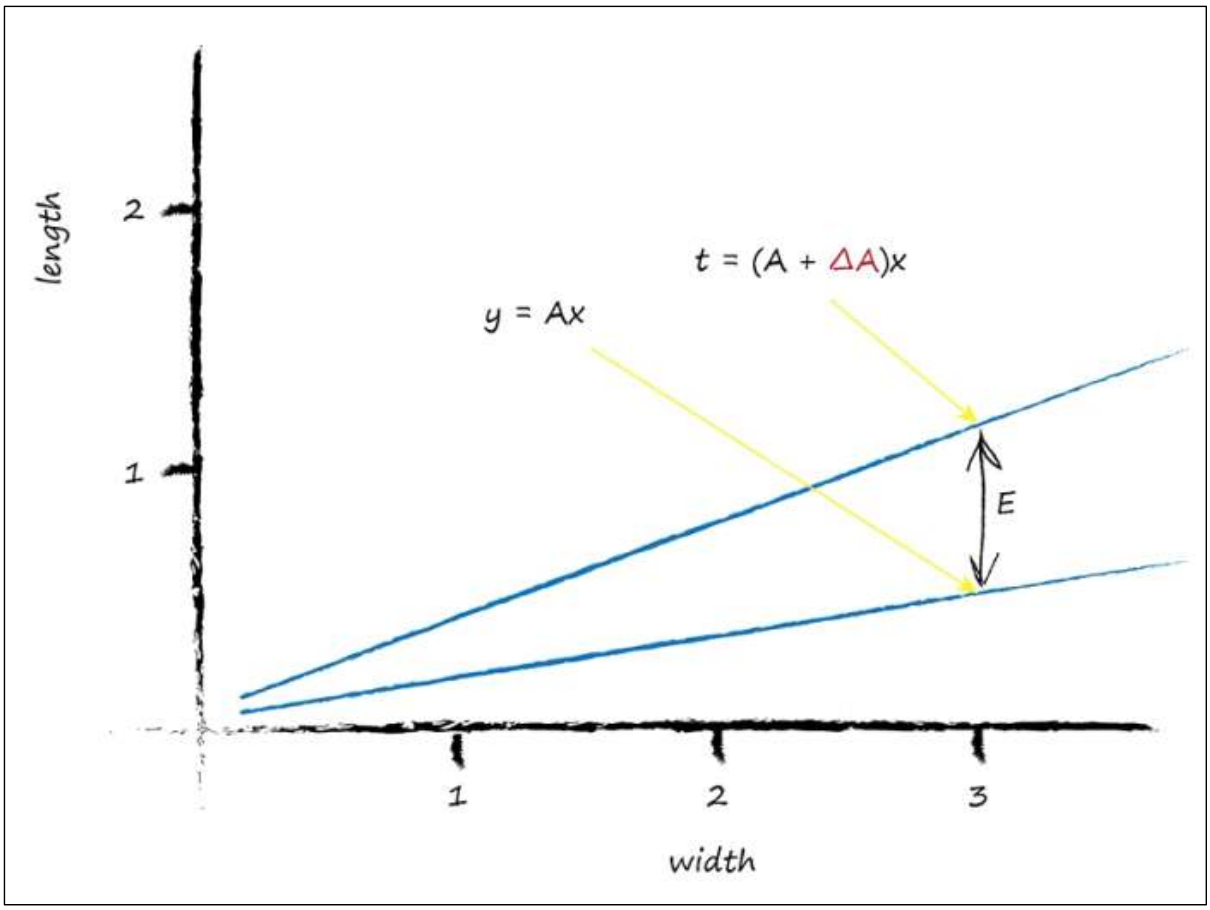
3) E(오차)를 어떻게 하면 정교화할 수 있을 것인가?

- 직선의 방정식

$$y = Ax$$

- A에 임의의 값을 넣으면 제대로 된 분류자를 만들 수 없음
- 목표값을 t라고 했을때 A를 조금씩 조정해나가야 함
- 작은 변화를 Δ(델타)라고 하고 'Δ' 이 기호를 사용함

$$t = (A + \Delta A)x$$



$$E = t - y$$

$$= Ax + (\Delta A)x - Ax$$

$$E = (\Delta A)x$$

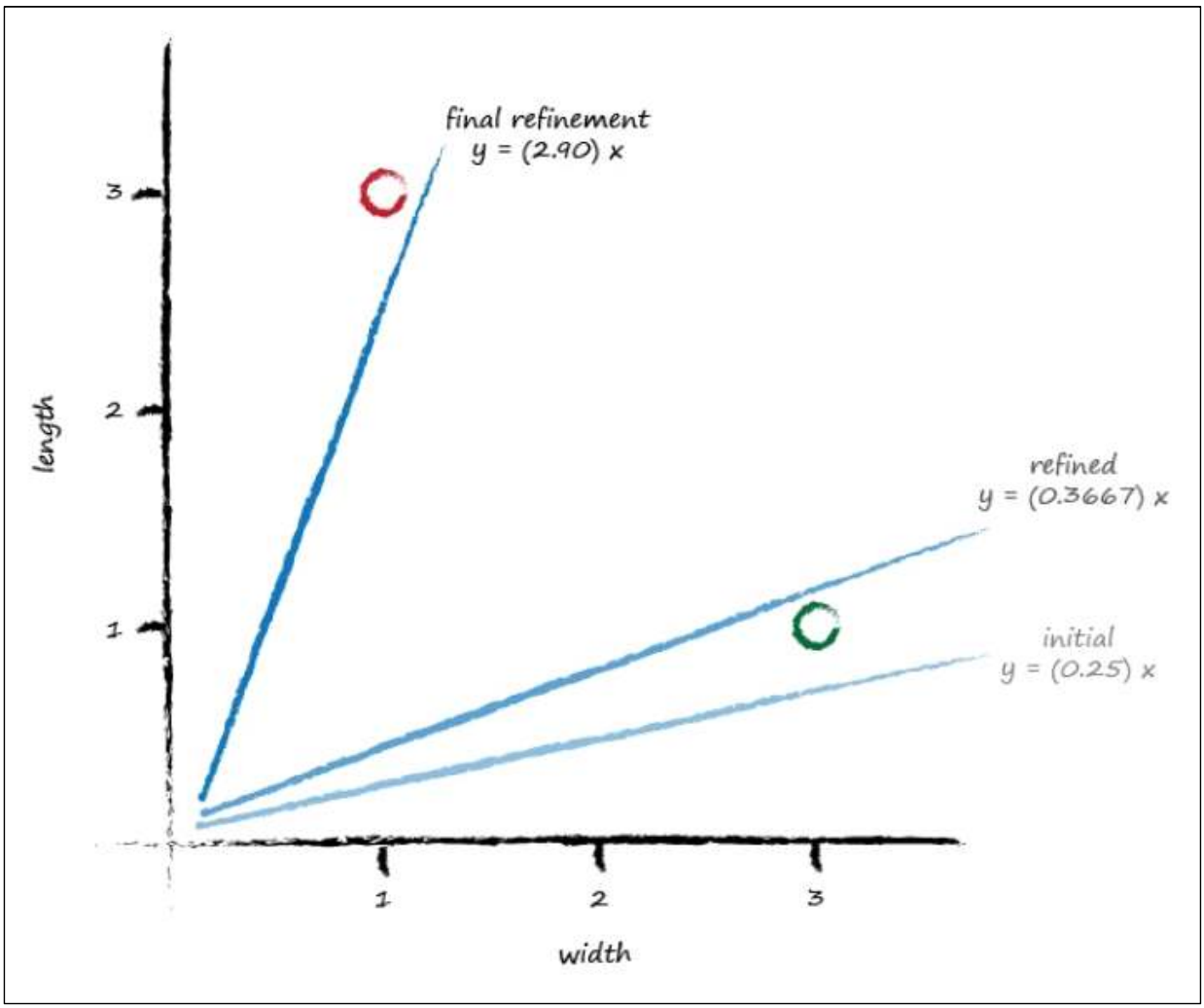
$$\Delta A = E/x$$

$$\Delta A = E/x = 0.35 / 3.0 = 0.1167$$

- 결론적으로 이전 0.25에 0.1167을 더한 0.3667이 계산됨

- 2번째 학습 데이터를 적용해 보면, 우선 3.0이 아니라 그 아래에 위치해야 하기 때문에 2.9로 계산

```
y = Ax
  = 2.9 = (0.3667) * 1.0 = 2.5333
  = 즉 0.3667 + 2.53333 인 2.9로 업데이트 됨
```



- 지금까지의 내용 정리

1) 최초의 직선(랜덤하게 아무 곳이나)
2) 첫번째 학습 데이터를 학습하여 1차 업데이트 직선
3) 두번째 학습 데이터를 학습하여 최종 업데이트 직선

- 문제는 기울기가 마지막 학습데이터에만 의존적임(원하는 분류자가 만들어진 것이 아님)
- 지금의 방법은 마지막 데이터 외에는 필요가 없음(강화학습이 아님)

4) 최종 방법 : 업데이트의 정도를 조금씩 조정

- L을 학습률이라고 함

```
ΔA = L(E / x )
```

- 예를 들어 학습률이 0.5일때, 첫번째 학습데이터 학습

```
0.25 ->

ΔA= L(E/ x) = 0.5 * 0.35 / 3.0 = 0.0583

A는 0.25 + 0.0583 = 0.3083
```

- 두번째 학습데이터로 학습하면

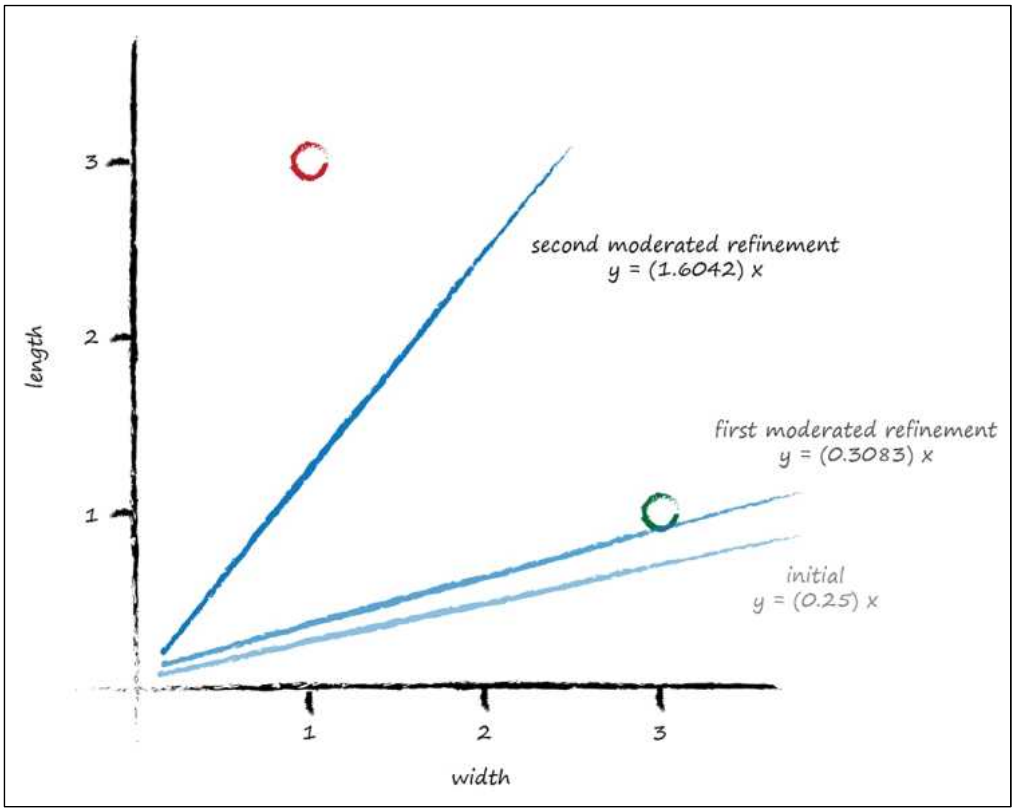
```
A= 0.3083 ->

2.9 = 0.3083 * 1.0 = 0.3083

(2.9 - 0.3083) = 2.5917

ΔA= L(E/ x) = 0.5 * 2.5917 / 1.0 = 1.2958

0.3083 + 1.2958 = 1.6042
```



- 단 2개의 학습데이터를 가지고 학습해서 나온 결과 선형함수는

```
y = 1.6042 * x
```

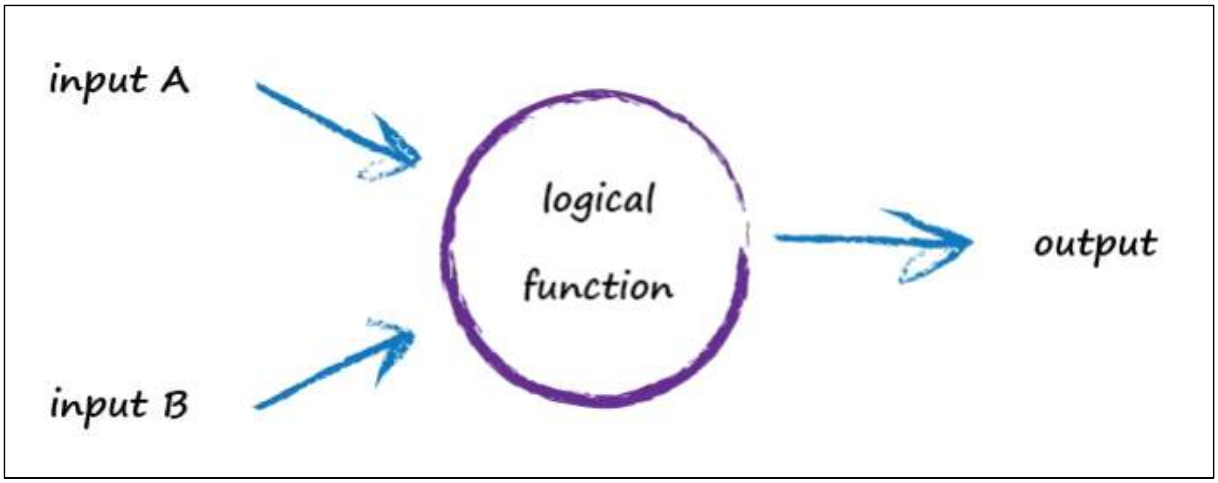
- 검증 : x 가 2이고 y가 2인 벌레는

```
1.6042 * 2 = 3.2084
3.2084보다 y가 크면 애벌레 / y가 작으면 무당벌레
y가 3.2084보다 작으므로 '무당벌레'임
```

■ 2개의 분류자

1) 1개의 분류자를 인공신경망에 적용하기에는 충분치 못함

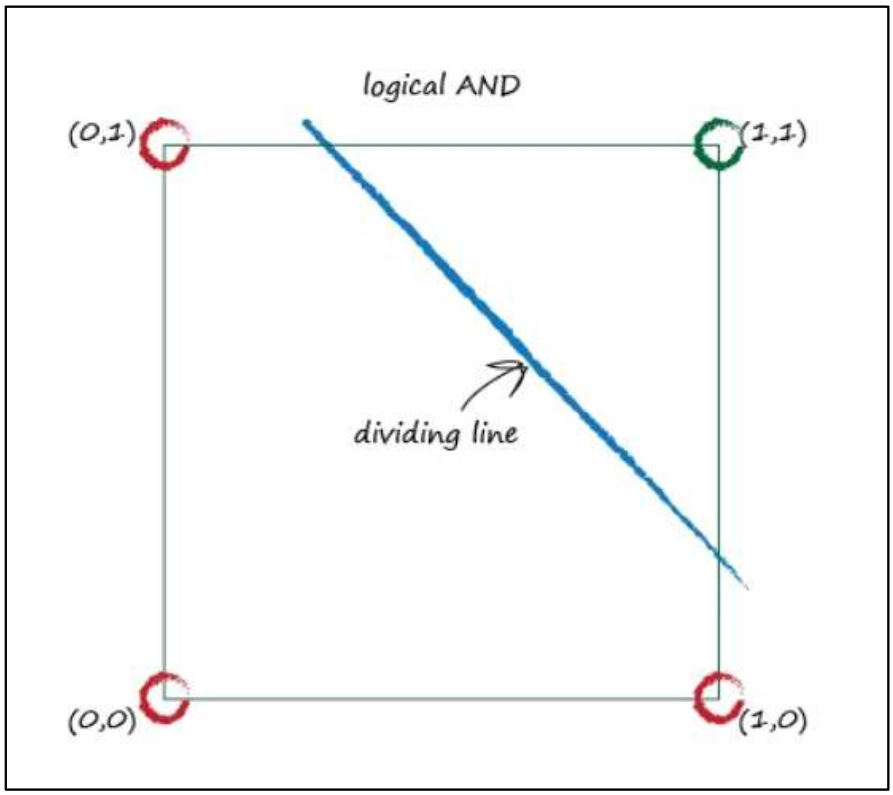
2) boolean의 경우



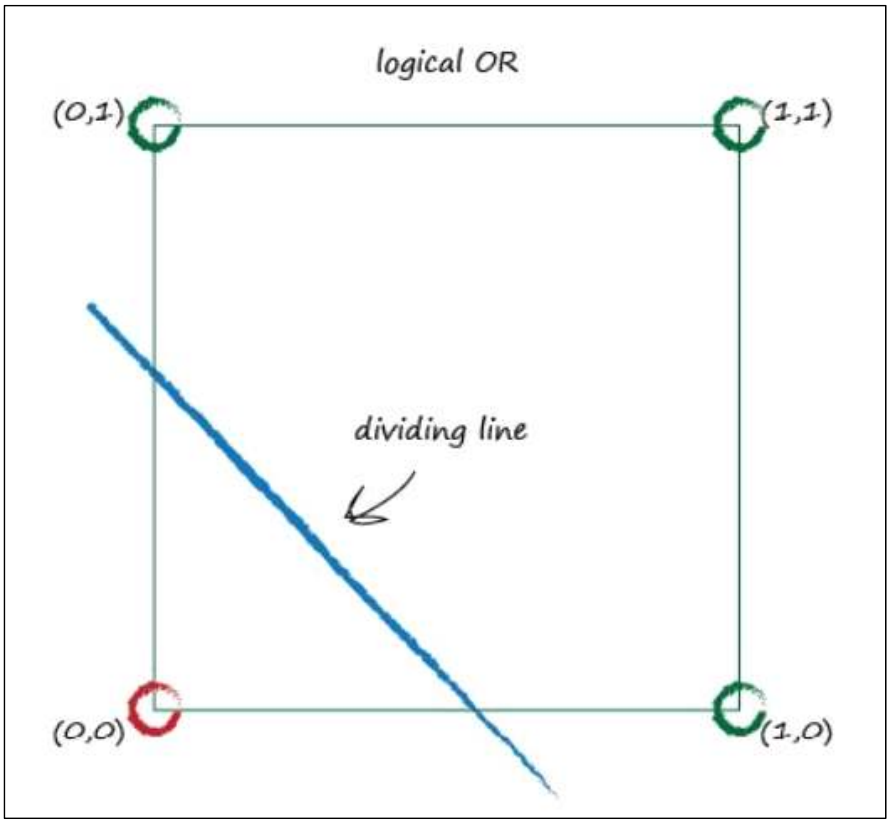
3) boolean의 논리합 / 논리곱

입력 A	입력 B	논리합	논리곱
1	1	1	1
1	0	1	0
0	1	1	0
0	0	0	0

4) 논리곱의 경우

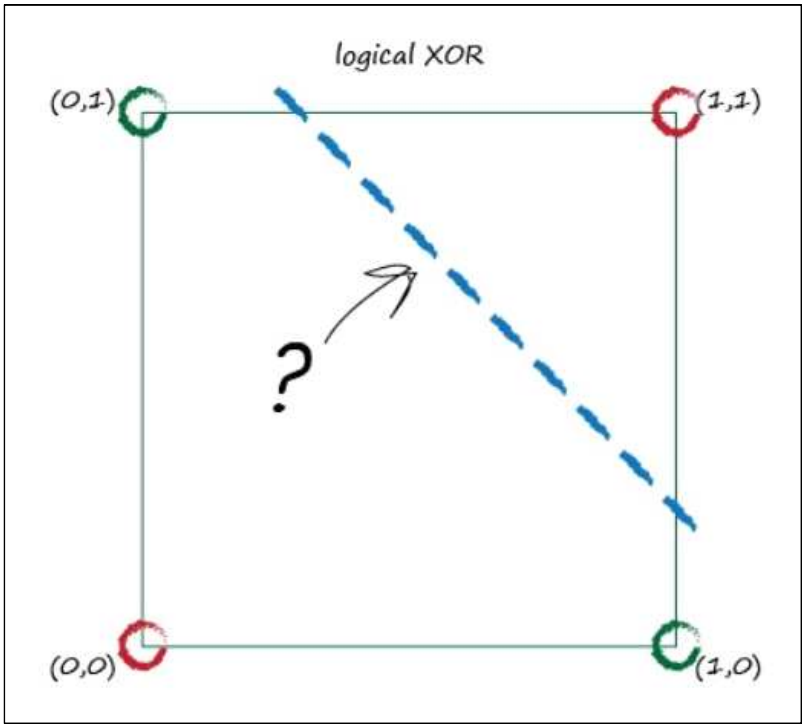


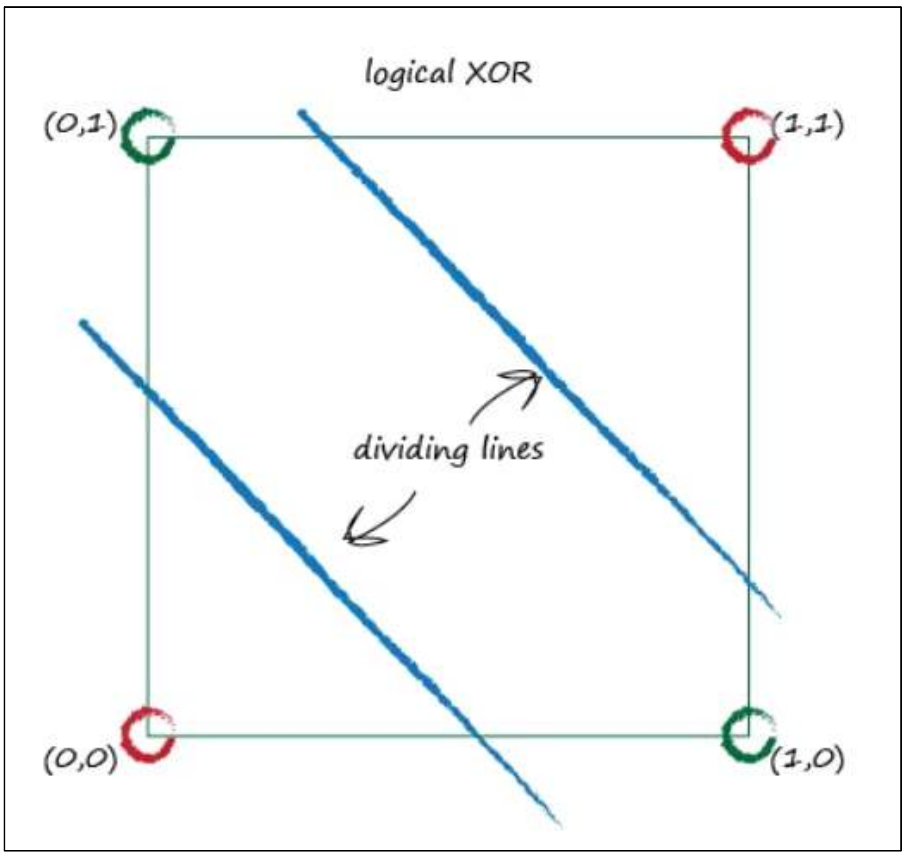
5) 논리합의 경우



6) 배타적 논리합(XOR)

입력 A	입력 B	배타적 논리합
0	0	0
0	1	1
1	0	1
1	1	0





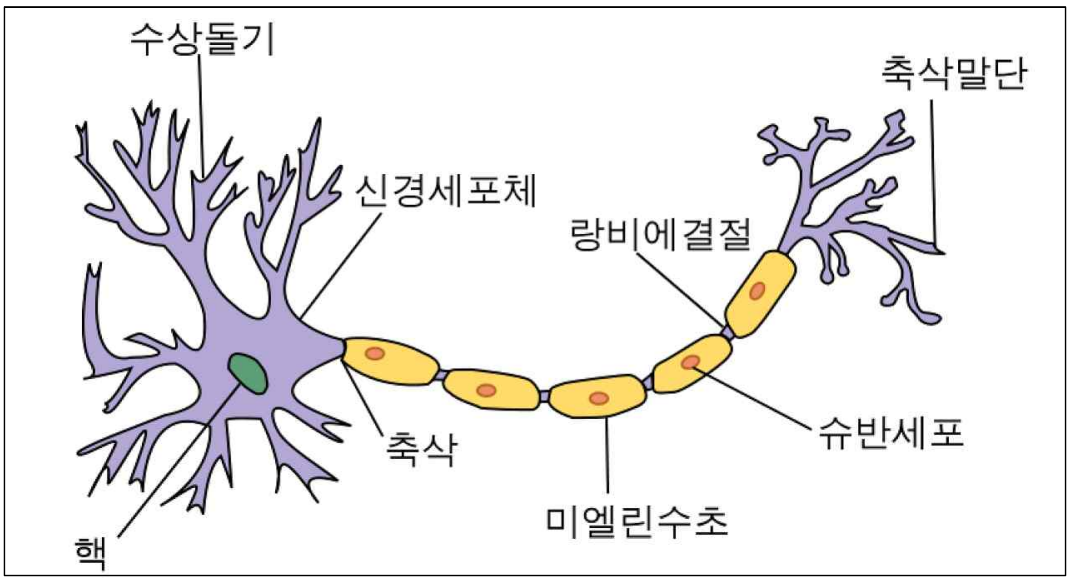
■ 대자연의 컴퓨터 : 뉴런

1) 뇌의 크기가 매우 작은 비둘기조차, 엄청난 자원을 가지고 빠른 속도로 연산하는 컴퓨터보다 훨씬 광범위한 능력을 가지고 있음

2) 전통적인 컴퓨터는 데이터를 순차적으로 처리

- 명확한 체계에 의해 정확히 수행
- 냉철하고 에누리가 없는 연산만이 존재함

3) 동물의 뇌는 비록 컴퓨터보다 느리게 작동하지만, 신호를 순차적이 아니라 병렬적으로 처리하며, 그 처리 특성 중 하나는 '불명확성'



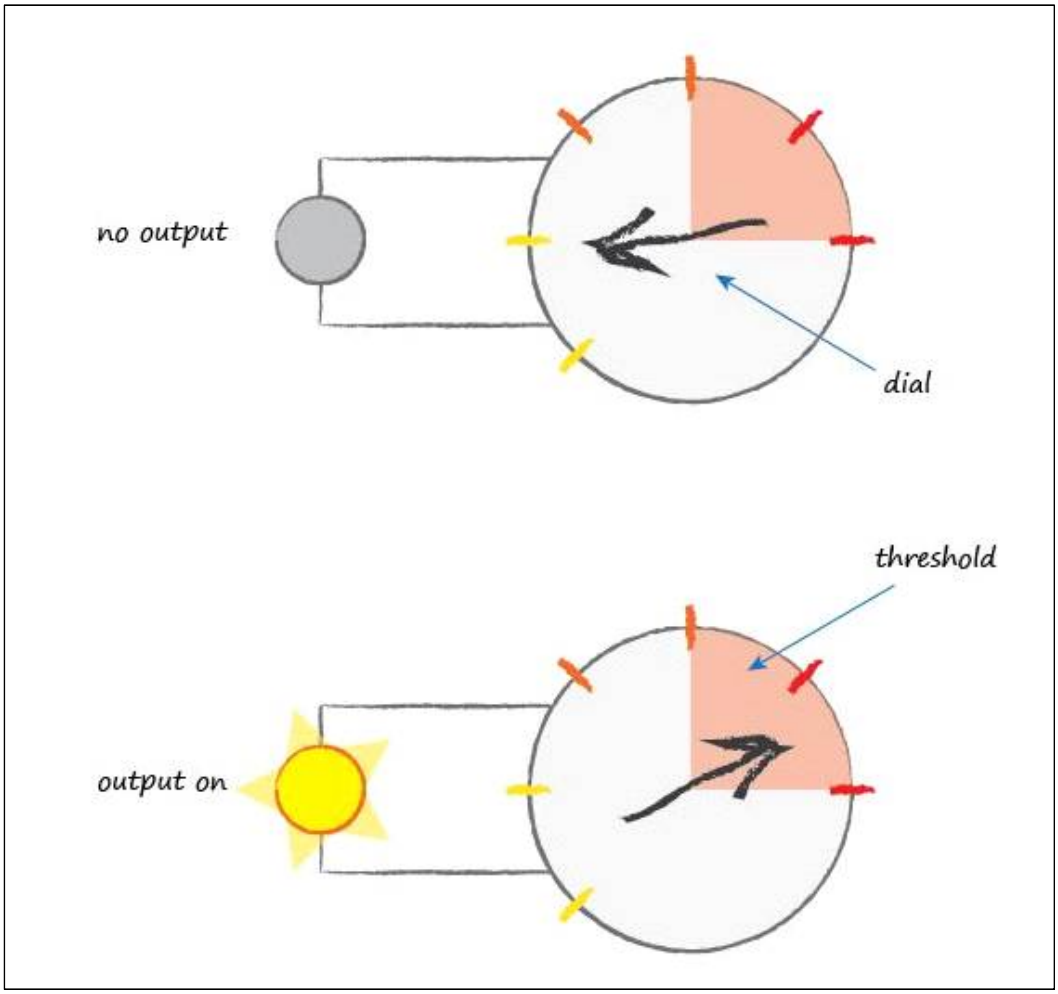
4) 인간의 뇌는 1천억 개의 뉴런을 가지고 있음

- 초파리는 겨우 10만 개의 뉴런을 가지고 있지만, 잘 날아다니고, 음식을 찾아 섭취하고, 위협을 피하는 등 상당히 복잡한 업무를 수행함 : 10만 개 정도라면 오늘날의 컴퓨터가 충분히 복제를 시도할 만한 수준

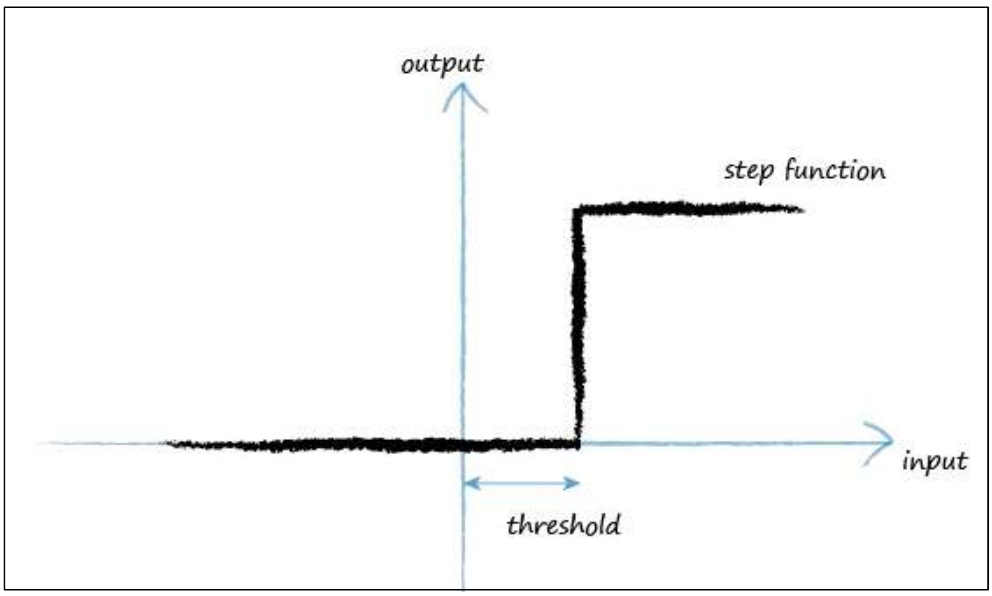
5) 뉴런의 동작 원리

- 뉴런은 전기 입력을 받아 또 다른 전기 신호를 발생시킴 : 예측자에서 입력을 받아 어떤 처리를 해서 결과를 출력하는 것과 매우 흡사함

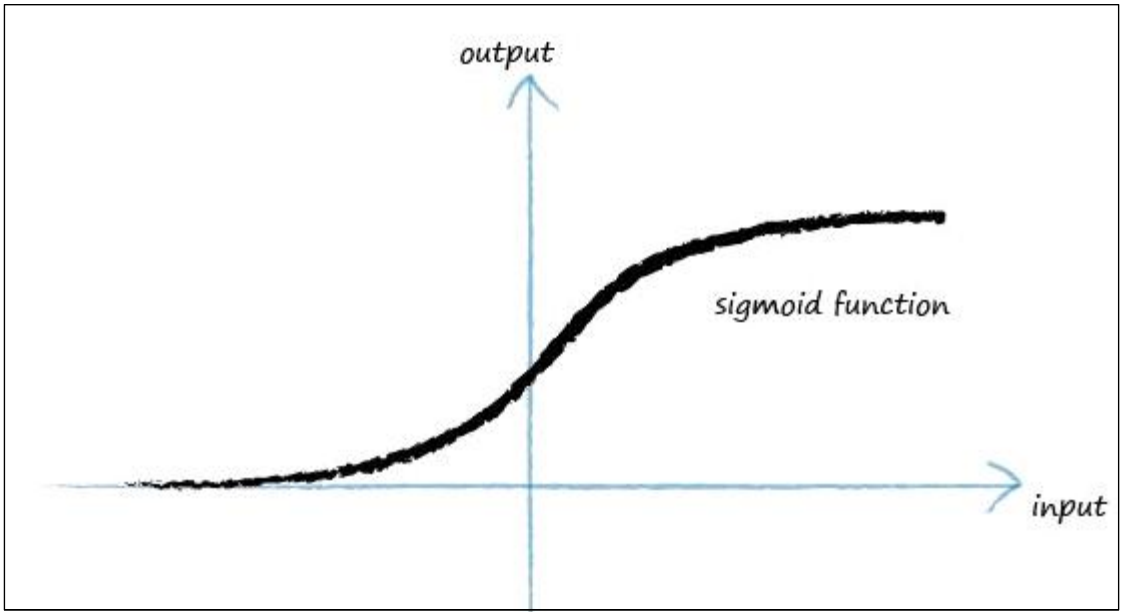
- 뉴런은 입력을 받았을 때 즉시 반응하지 않음
- 입력이 누적되어 어떤 수준으로 커진 경우에만 출력을 하게 됨 : 분계점에 도달해야 출력이 발생됨



- 입력신호를 받아 특정 분계점을 넘어서는 경우에 출력 신호를 생성해주는 함수를 활성화 함수라고 함
- 가장 간단한 형태는 '계단 함수'



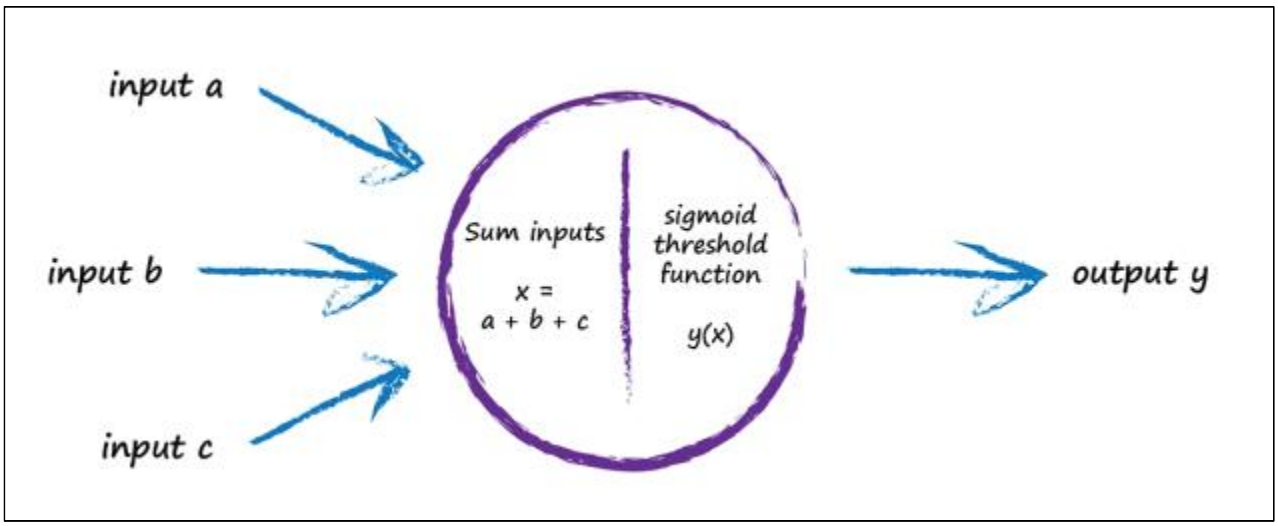
- 실제로 계산해보면 S자 모양의 함수로 자연스럽고 현실에 가까운 함수 : 시그모이드 함수



- 이 함수를 '로지스틱 함수'라고도 함

$$y = \frac{1}{1 + e^{-x}}$$

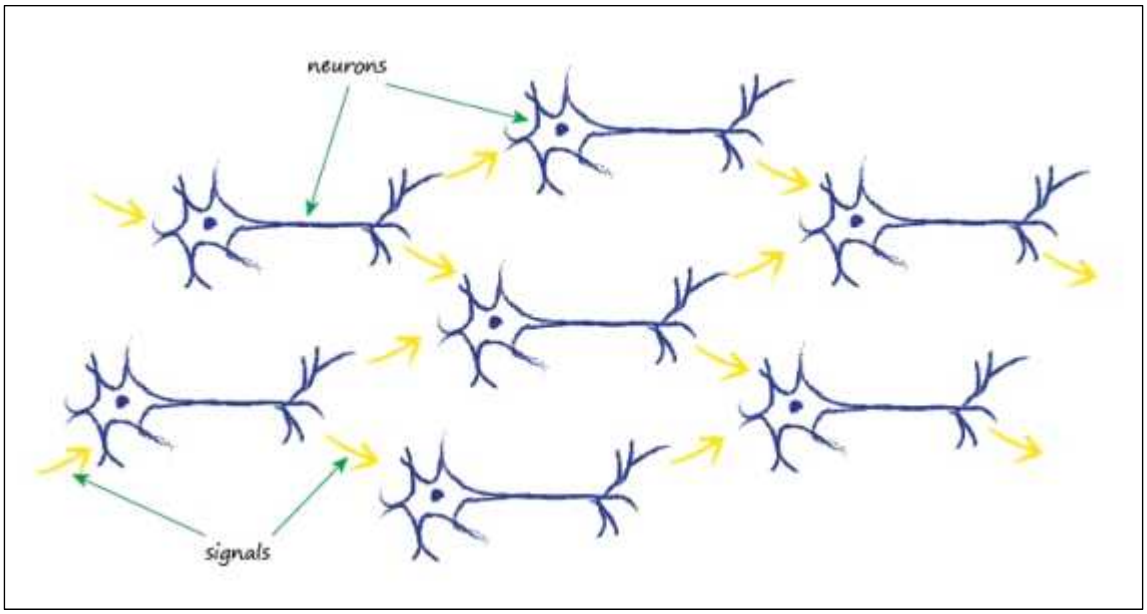
- 뉴런의 작동형태



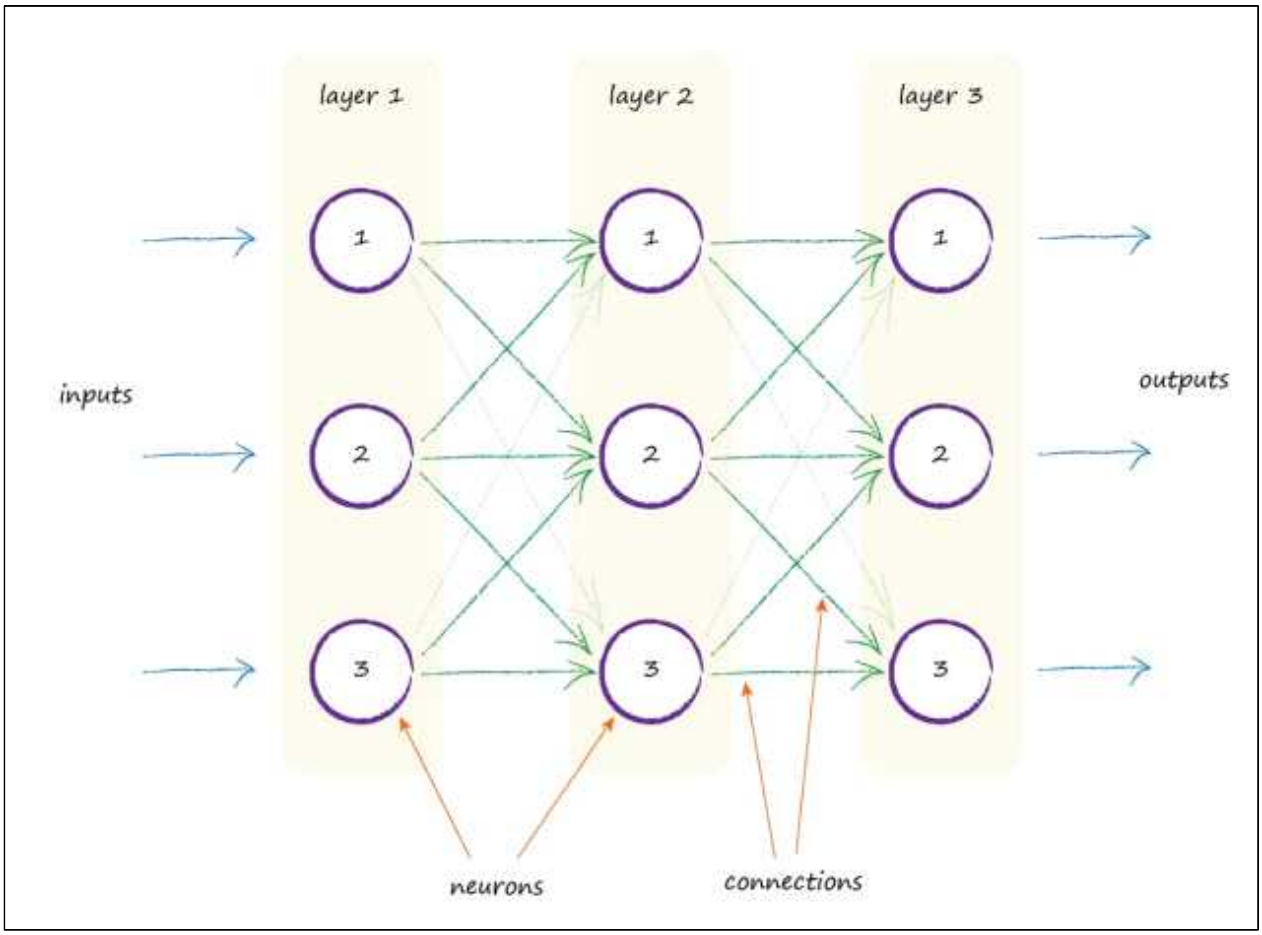
- 입력 a, b, c 의 합이 분계점인 x를 넘어서면 출력 / 아니면 출력하지 않음

■ 뉴런과 인공 신경망

1) 실제 뉴런의 모양



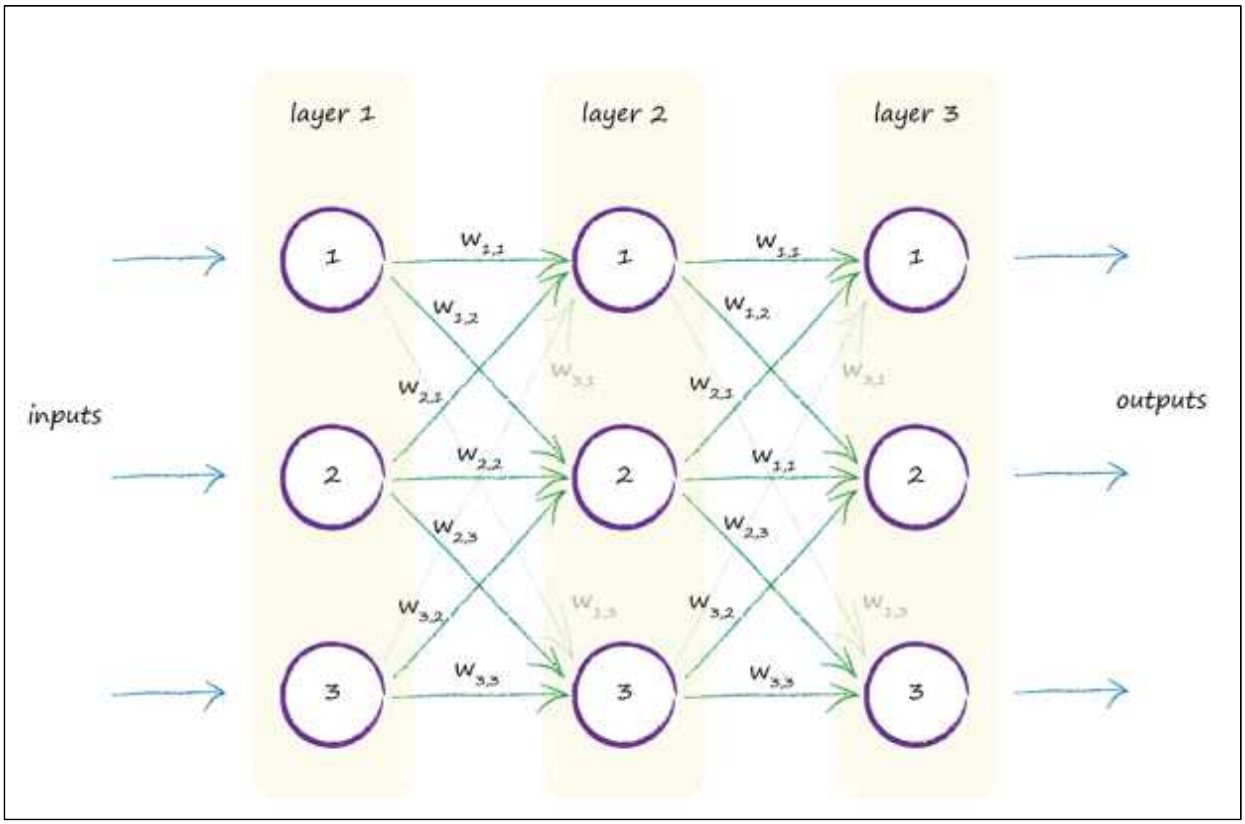
2) 인공신경망의 간단한 형태



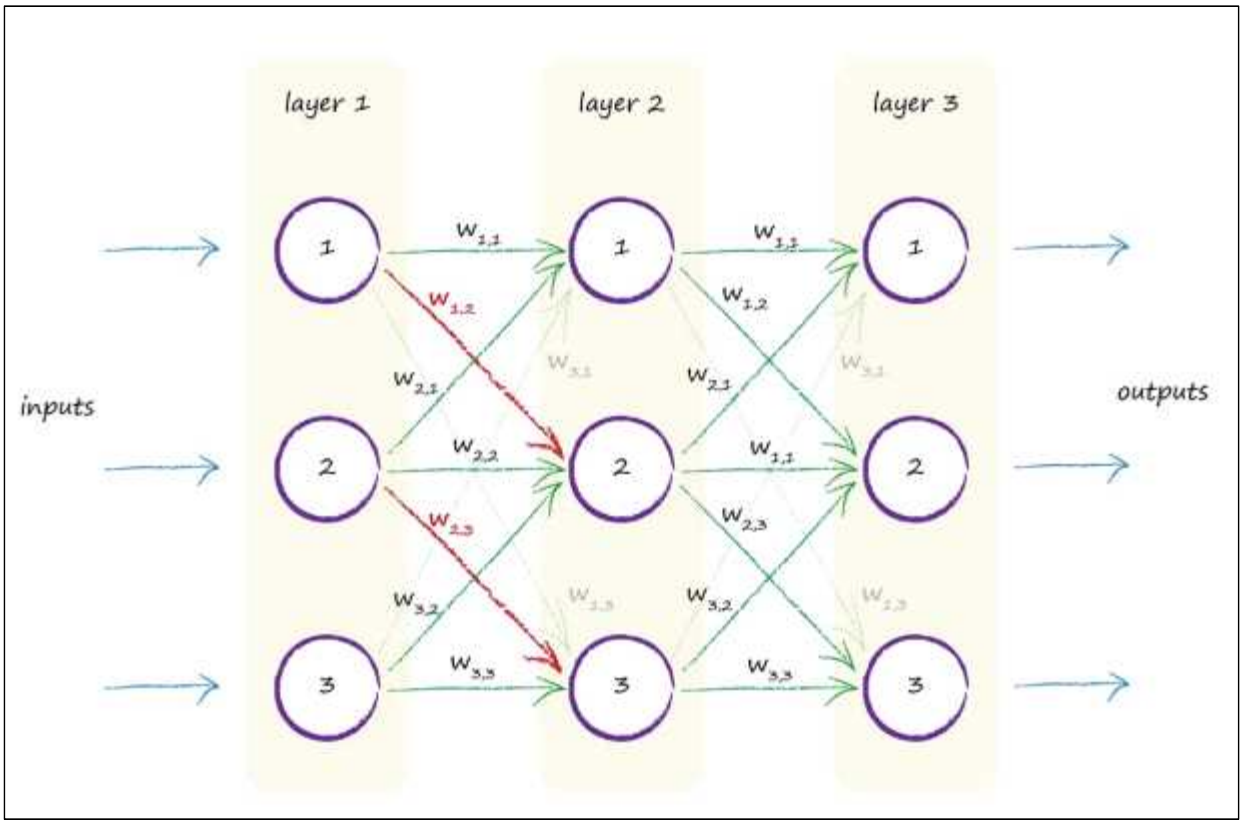
- 3개의 계층이 있고, 각 계층에는 뉴런이 3개씩 존재
- 각각의 인공뉴런을 '노드'라고 부름

3) 인공신경망 학습

- 노드간의 연결 강도를 조정해 나아감 : 가중치

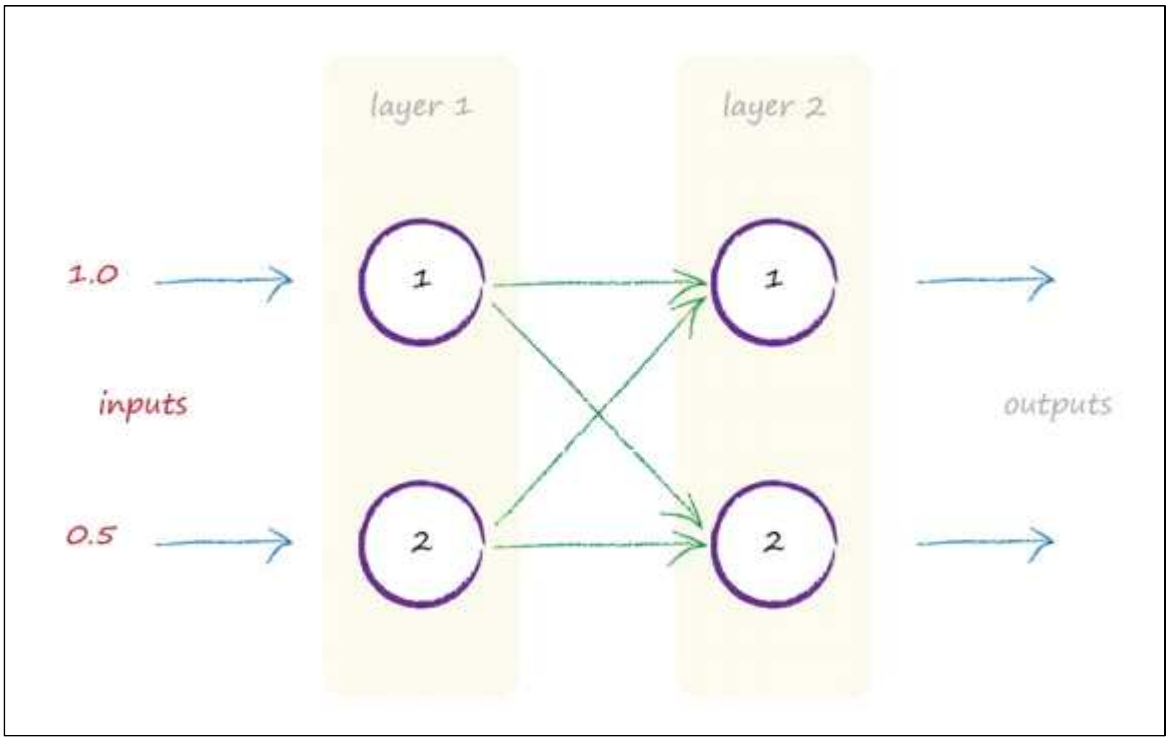


- 1->2 / 2->3 으로 가중치를 줘서 표현한 형태



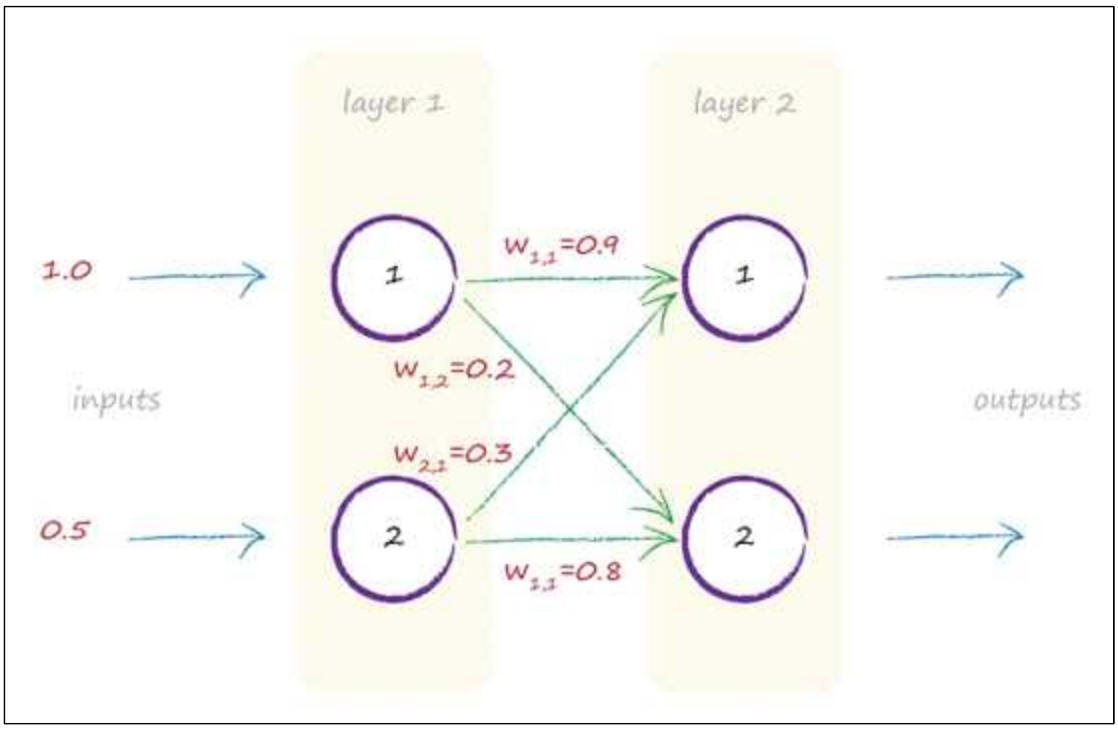
4) 간단한 계산해보기

- 만약 1과 0.5를 입력한다면,



- 가중치는 임의의 값으로 시작(학습하면서 점점 개선되므로 상관없음)

- w1,1= 0.9
- w1,2= 0.2
- w2,1= 0.3
- w2,2= 0.8



- 계층2의 1번 노드 계산

```
x = (노드1의 출력값 * 가중치) + (노드2의 출력값 * 가중치)
x = (1.0 * 0.9) + (0.5 * 0.3)
x = 0.9 + 0.15
x = 1.05
```

- 이제 시그모이드 함수(활성화 함수)에 이 입력값을 넣어 출력값을 계산

```
y = 1 / (1 + 0.3499) = 1 / 1.3499
y = 0.7408
```

- 계층2의 2번 노드 계산

```
x = (노드1의 출력값 * 가중치) + (노드2의 출력값 * 가중치)
x = (1.0 * 0.2) + (0.5 * 0.8)
x = 0.2 + 0.4
x = 0.6
```

- 이제 시그모이드 함수(활성화 함수)에 이 입력값을 넣어 출력값을 계산

```
y= 1/(1 + 0.5488) = 1/(1.5488)
y= 0.6457
```

- 결론