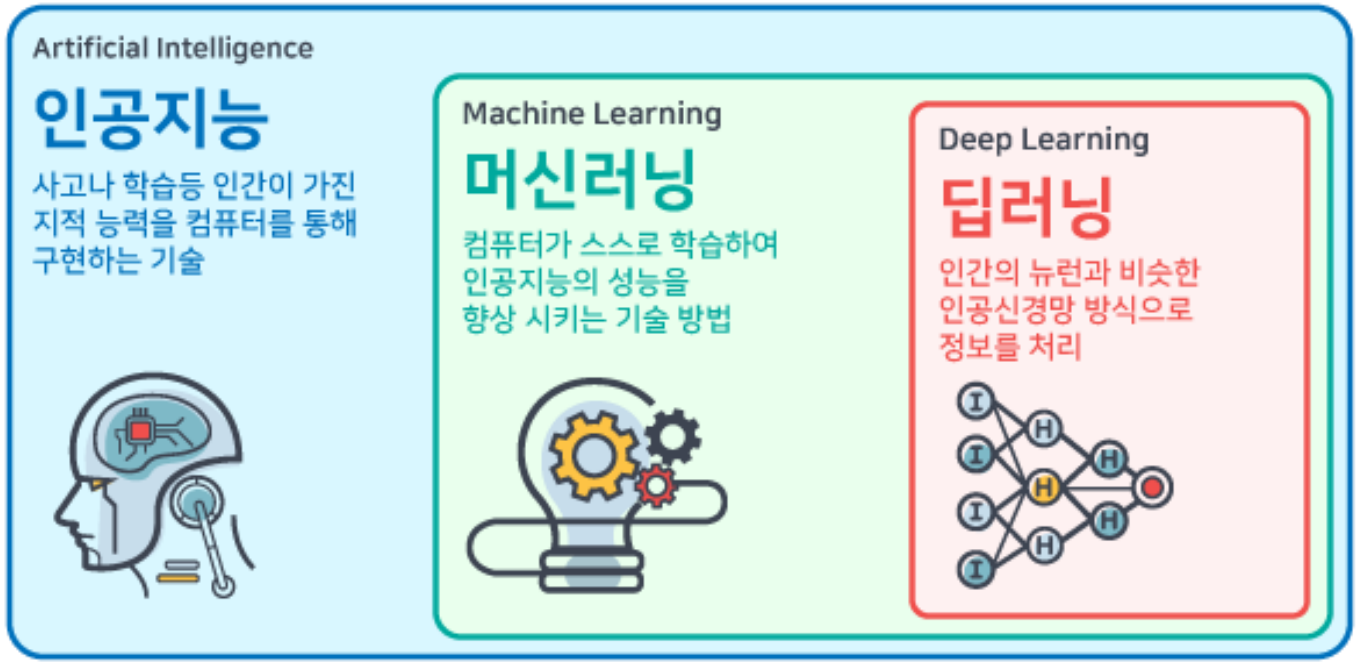
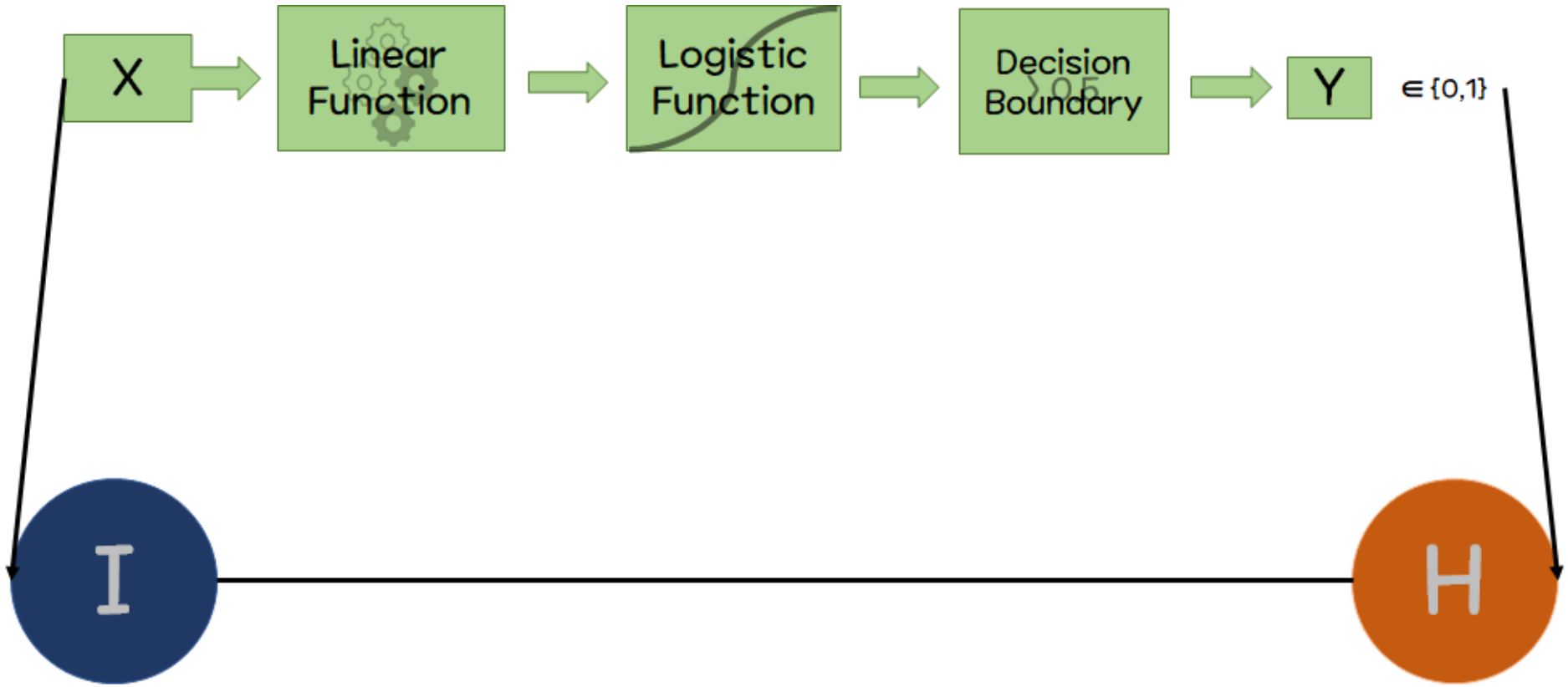
딥러닝 개념

- 사람은 기억에 대한 정보를 뉴런이라는 기본 단위에 저장한다.
- 또한 뉴런 여러 개가 쌓여 엄청난 정보를 처리한다.
- 딥러닝은 뉴런과 인간의 뇌구조를 형상화하여 인공신경망을 구현한다.
- 앞서 배운 머신러닝의 기법 중 로지스틱회귀 모델이 기본 단위(=뉴런)가 된다.



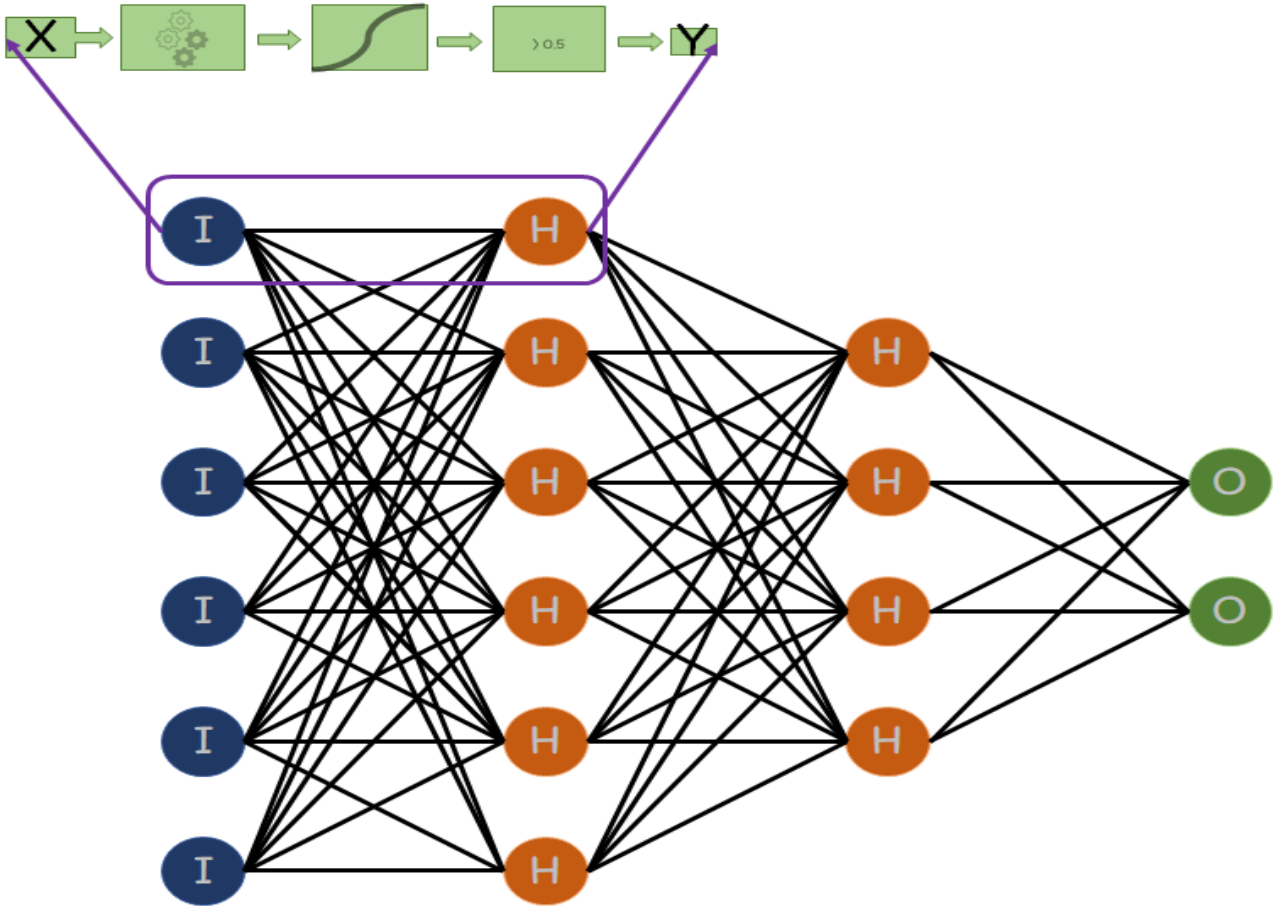
딥러닝 개념

- 머신러닝 기법 중(회귀 분류 = 로지스틱 회귀)를 딥러닝의 기본단위 뉴런으로 본다.
- 아래와 같은 모양

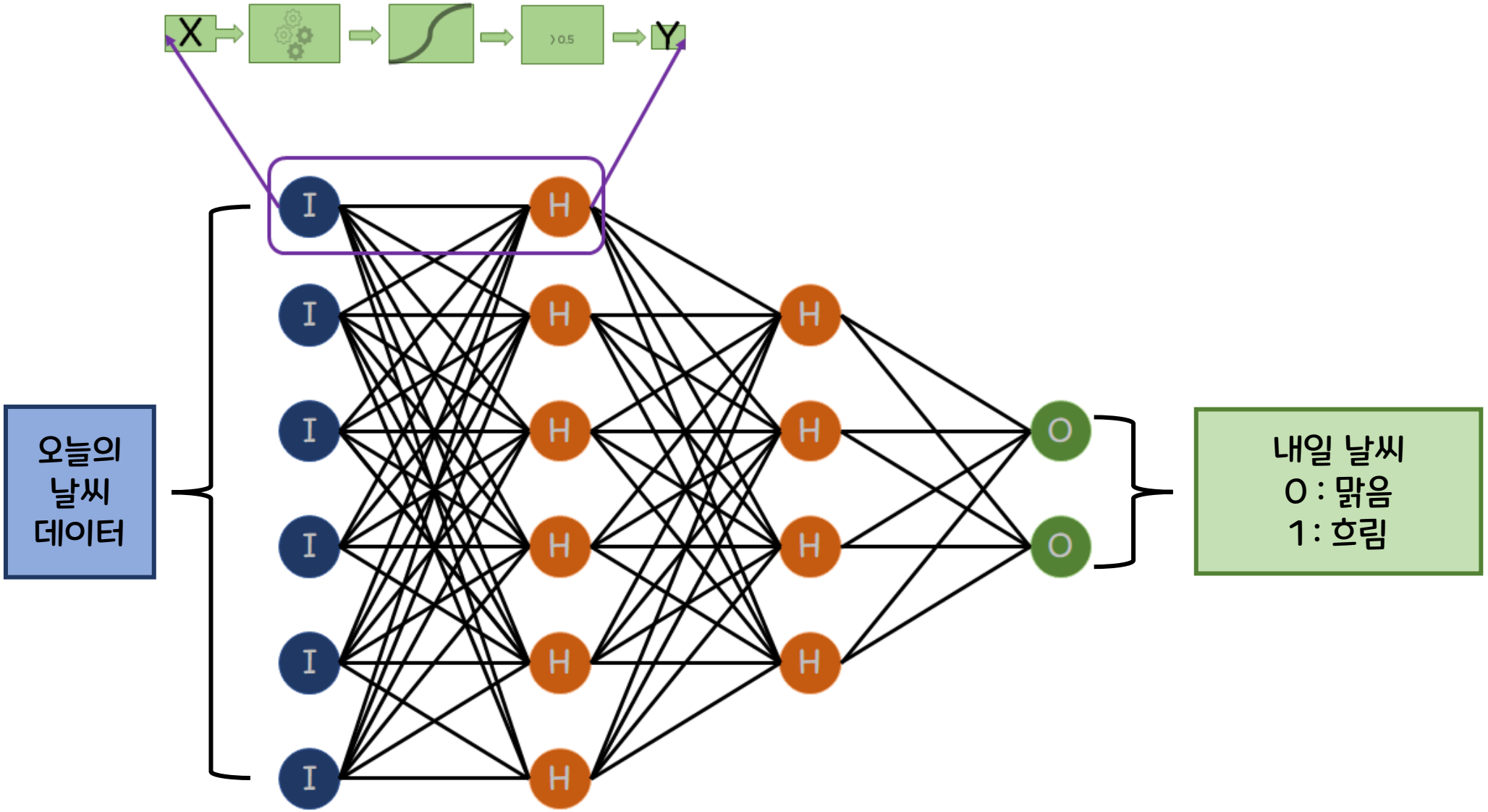


딥러닝 개념

- 뉴런을 여러 개를 사용하여 인공신경망을 구축한다.
- 이를 DL(Deep Learning)이라고 한다.

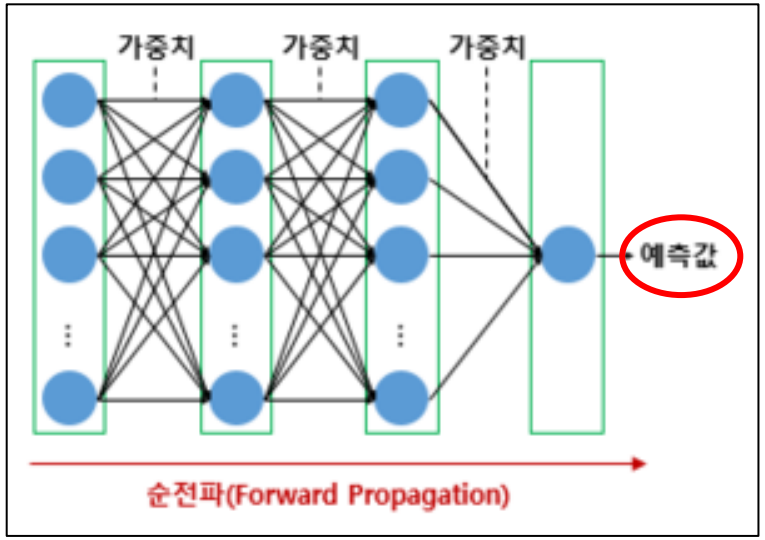


딥러닝 개념



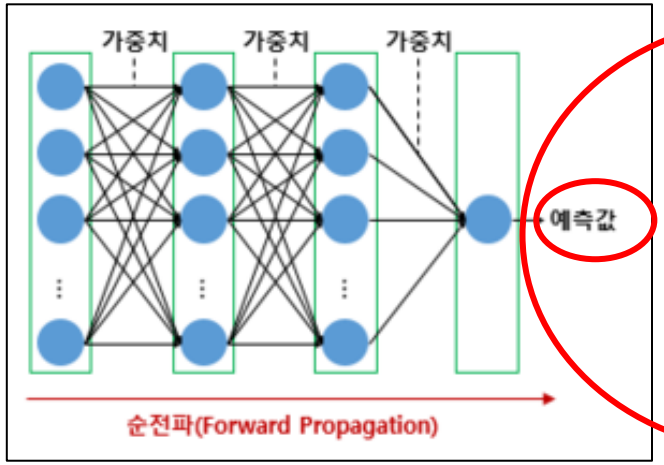
딥러닝 개념

- **순전파(전방연산, Forward-Propagation)**
  - 순전파는 모델이 구현한 입력 층부터 출력 층까지 순차적으로 접근하여 변수들을 계산하고 저장하는 것을 의미한다.
  - 즉, **입력 데이터(X)를 받아 예측 값(Y)를 얻는 연산(과정)**이다.
  - 모든 층(Hidden Layer)은 행렬로 표현할 때, $H_i = X * W_i$로 표현된다.
    - 이는 **행렬의 내적 연산을 의미**한다.



딥러닝 개념

- **오차(Error, Loss)**
  - 모델이 순전파를 통해 얻은 예측 값(output)과 데이터의 실제 값(Label)을 통해 오차를 계산함.
  - 오차를 구하는 오차함수(Loss_Func)는 여러가지가 있다.
    - 최소오차평균(Mean-Square-Error(MSE))
    - 크로스 엔트로피(Cross-Entropy) 등등
    - 오차함수는 개발자가 직접 정의 할 수도 있다.



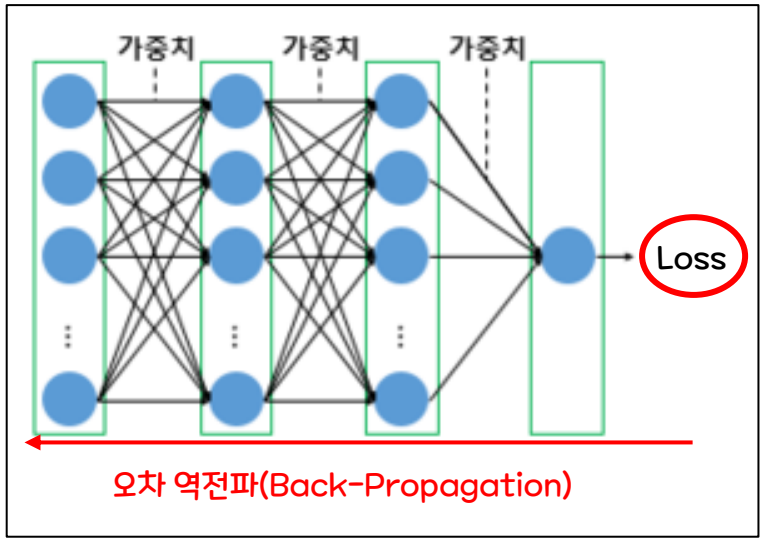
Label
10
32
.
.
.
-29

예측값 : output
실제값 : Label

Loss(오차) = **Loss_Function**(Label, output)

딥러닝 개념

- **역전파(오차 역전파, Back-Propagation)**
  - 가중치를 갱신하는 알고리즘이다.
  - 실제값(Label)과 예측값(output)을 오차 함수에 넣어 얻은 Loss값을 모델에 역으로 전파하여 가중치들을 갱신한다.
  - 가중치를 갱신하는 최적화 알고리즘은 여러 가지가 있다.
    - SGD, Adam 등등

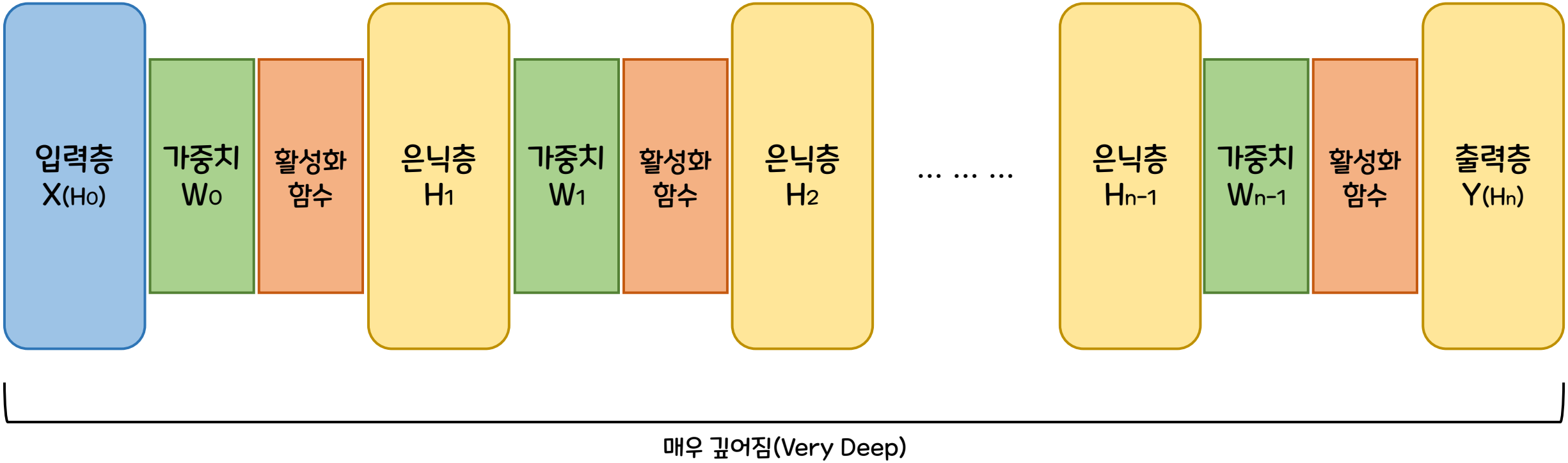


딥러닝 개념

- 한번의 **학습의 과정**을 나열해 보면,
  1. **순전파**(예측 값(Y) 계산)
  2. **Loss값 계산**(Loss_Fn(Label, output))
  3. **오차 역전파**(모델 내 가중치(W) 갱신)
- **모델에서 사용하는 가중치의 개수나 입력데이터의 크기 정도에 따라 학습의 시간이 결정된다.**
- **고속 연산 및 병렬 처리를 위해 GPU를 사용할 수 있다.**

딥러닝 개념

- 오차 역전파로인해 깊은-다층 퍼셉트론(DMLP)가 주목받기 시작하였다.



딥러닝 개념

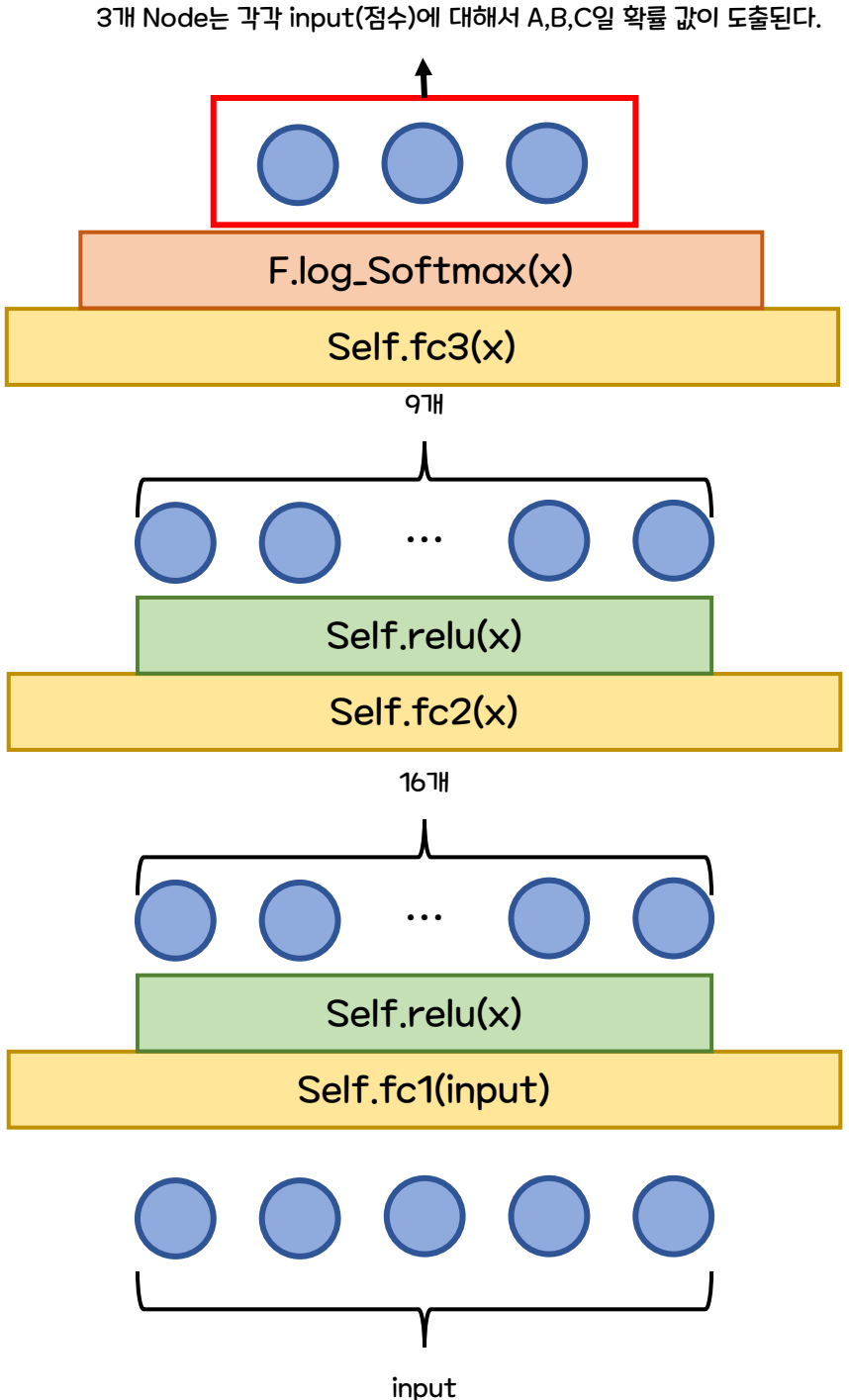
학습 설계

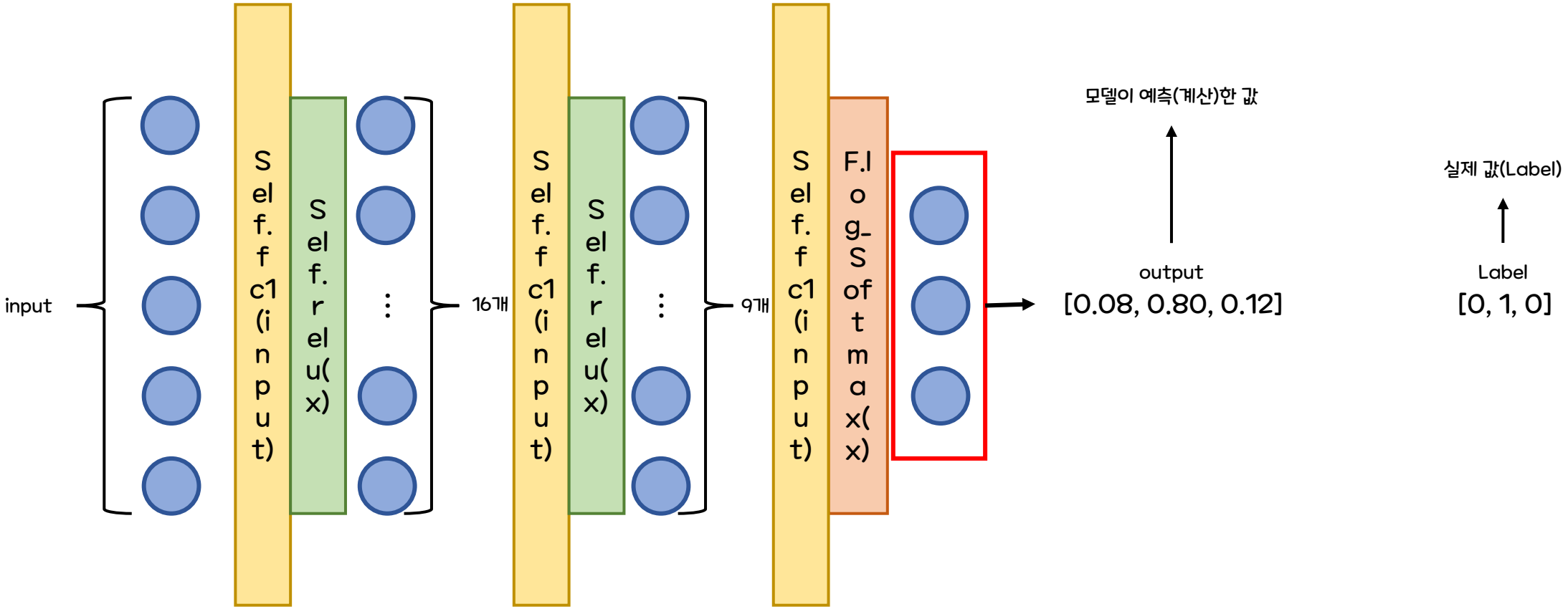
1. 순전파 (모델을 통해 예측값을 계산)
    - `output = model(input_data)`
2. Loss(Error) 계산
    - 실제 값(Label)과 모델이 예측한 값(output)을 비교함
    - Loss는 각 Task(문제)에 맞게 정의 해주어야함
        - 예측 : MSE
        - 분류 : CrossEntropy
        - 또한, 개발자가 직접 정의 할 수 있음
    - `Loss = Loss_fn(output, label)`
3. Loss를 모델 내부의 가중치(weight = parameters)에 적용(갱신)
    - 이 과정을 **오차역전파**라고 함.
    - 경사하강법을 적용하기 위한 가중치들의 미분값을 구하는 과정임
    - `Loss.backward()`
4. 오차역전파를 통해 구한 미분값을 통해 optimizer를 통해 가중치 갱신
    - **new_weight = old_weight - lr * old_weight'**
    - `opt.step()`
5. 1~4 과정을 EPOCH만큼 반복

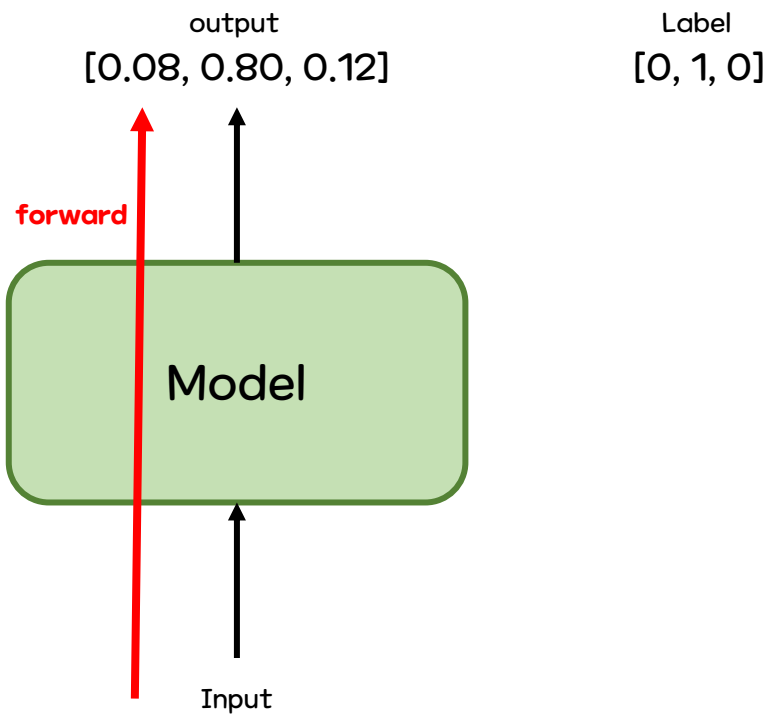
ID	중간	기말	과제1	과제2	프로젝트	학점
1120	80	89	97	92	88	A
1121	17	100	98	94	75	B
1124	37	42	68	57	72	C
1129	87	82	92	95	95	A
1131	69	78	90	87	90	B
1684	54	89	85	85	72	B
1893	61	76	75	95	87	B
1497	47	52	70	62	83	C
1622	60	75	77	70	40	C
1623	100	95	93	92	67	A
1732	91	88	85	93	79	A
1706	82	89	76	78	82	B
1724	97	76	98	97	86	A
1721	77	82	50	47	43	C
1918	81	83	95	72	86	B
1936	91	79	68	72	81	B
1937	29	81	71	62	57	C

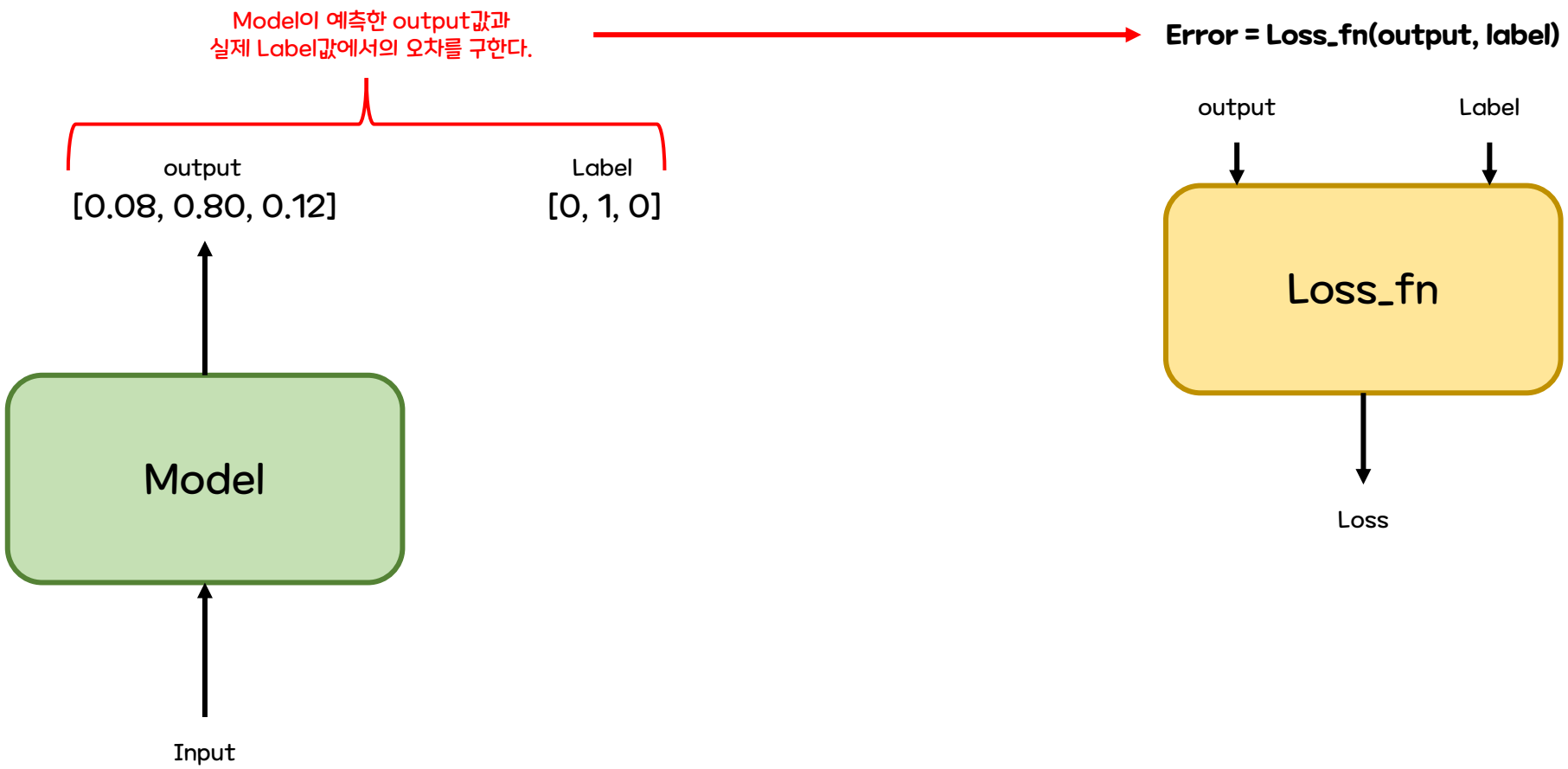
ID	중간	기말	과제1	과제2	프로젝트	학점
1865	81	72	100	89	95	A
2121	67	82	86	84	90	B
1832	88	57	94	72	98	B
1130	45	62	73	89	68	C
1219	100	72	79	45	50	C

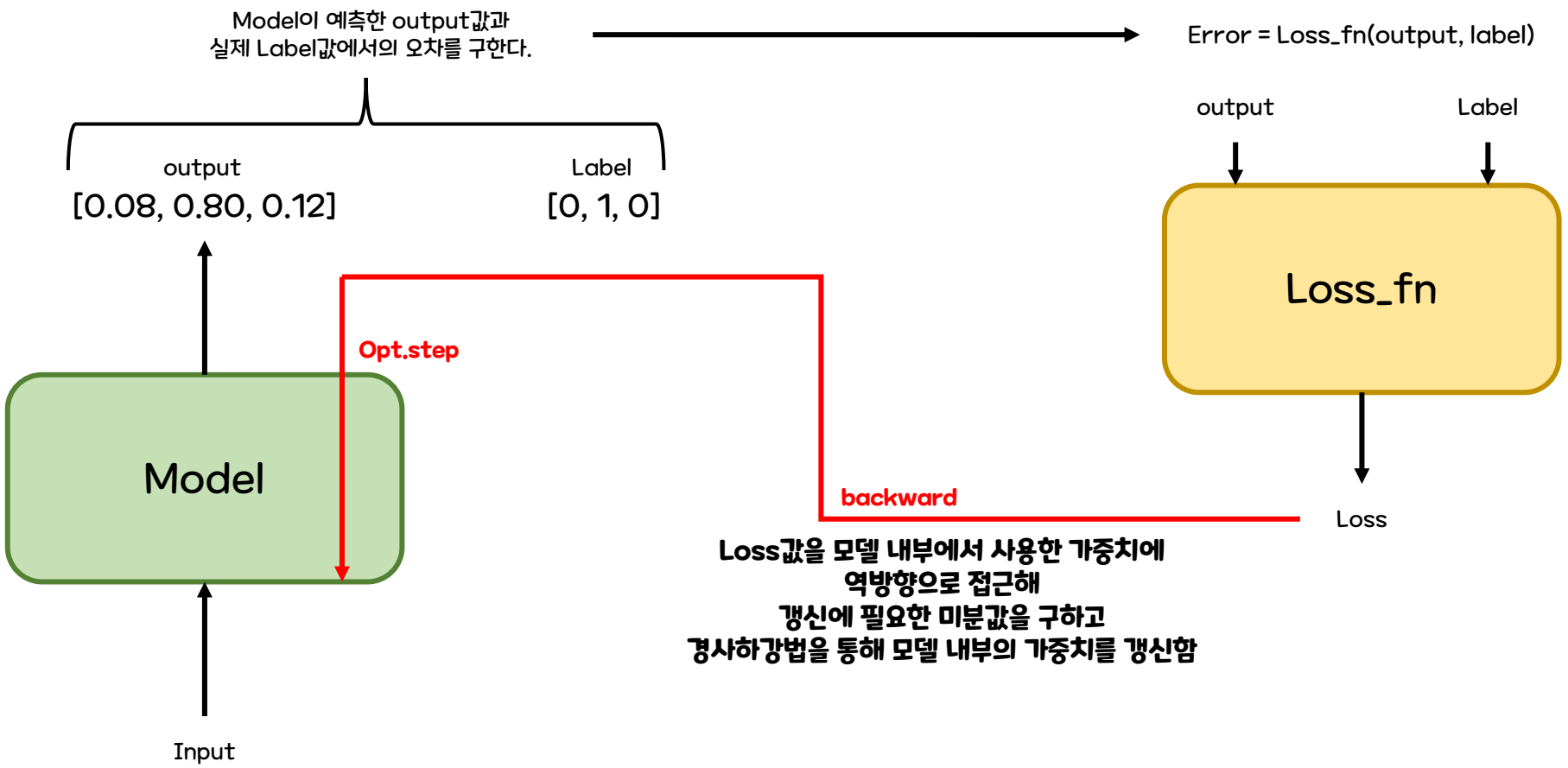
```python
class MyModel(nn.Module):
    def __init__(self):
        super(MyModel, self).__init__()

        self.fc1 = nn.Linear(5, 16)
        self.fc2 = nn.Linear(16 , 9)
        self.fc3 = nn.Linear(9, 3)

        self.act_fn = nn.ReLU()

    def forward(self, input):
        x = self.fc1(input)
        x = self.act_fn(x)

        x = self.fc2(x)
        x = self.act_fn(x)

        x = self.fc3(x)
        x = F.log_softmax(x, dim=1)
        return x
```




input
Self.fc1(input)
Self.relu(x)
16개
Self.fc1(input)
Self.relu(x)
9개
Self.fc1(input)
F.log_Softmax(x)
모델이 예측(계산)한 값
output
[0.08, 0.80, 0.12]
실제 값(Label)
Label
[0, 1, 0]


output
[0.08, 0.80, 0.12]
Label
[0, 1, 0]
forward
Model
Input


Model이 예측한 output값과
실제 Label값에서의 오차를 구한다.
Error = Loss_fn(output, label)
output
[0.08, 0.80, 0.12]
Label
[0, 1, 0]
Model
Input
output
Label
Loss_fn
Loss


Model이 예측한 output값과
실제 Label값에서의 오차를 구한다.
Error = Loss_fn(output, label)
output
[0.08, 0.80, 0.12]
Label
[0, 1, 0]
output
Label
Loss_fn
Opt.step
Model
backward
Loss
Loss값을 모델 내부에서 사용한 가중치에
역방향으로 접근해
갱신에 필요한 미분값을 구하고
경사하강법을 통해 모델 내부의 가중치를 갱신함
Input

아래 과정을 EPOCH만큼 반복