Deep Learning from Scratch

조용군

iceman4u@naver.com

역전파 알고리즘 BACKPROPAGATION

역전파 알고리즘

- ✓ **Supervised Learning**

- ✓ **전파(propagation) 단계**
  - 학습 데이터로부터 실제 출력값을 구하고 목적값(정답)과 실제 출력값의 차이인 오차를 계산하여 각층에 대해 역순으로 전달
- ✓ **가중치 갱신/수정 단계**
  - 전파된 오차를 이용하여 가중치를 갱신

- ✓ **은닉층이 포함된 다층 신경망을 학습할 수 있는 알고리즘**
  - 역전파 이전에는 은닉층의 목적값(정답)을 결정할 수 없었기 때문에 2층 이상의 신경망을 학습시킬 수 없었음

  - http://untitledtblog.tistory.com/90
  - http://llnntms.tistory.com/31

계산 그래프

✓ Computation graph

- 계산 과정을 그래프로 나타낸 것
  - 노드(node): 연산 내용
  - 에지(edge): 값, 계산 결과

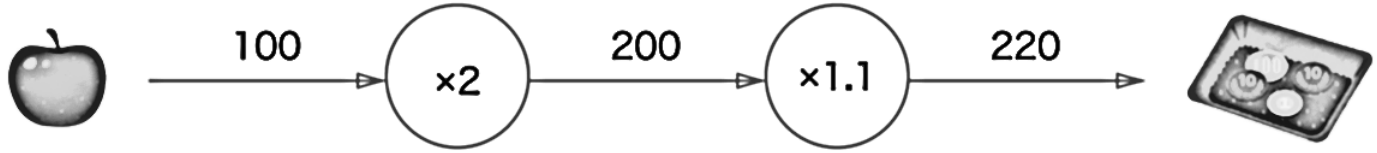
- 계산 그래프를 이용한 문제 풀이
  - 계산 그래프 구성
  - 그래프에서 계산을 왼쪽에서 오른쪽으로 진행
    - 순전파(forward propagation)
  - 오른쪽에서 왼쪽으로 계산 진행
    - 역전파(backpropagation)

계산 그래프에 익숙해지기

- ✓ 문제 1: 맹구는 슈퍼에서 1개에 100원인 사과를 2개 샀습니다. 이때 지불 금액을 구하세요. 단, 소비세가 10% 부과됩니다.

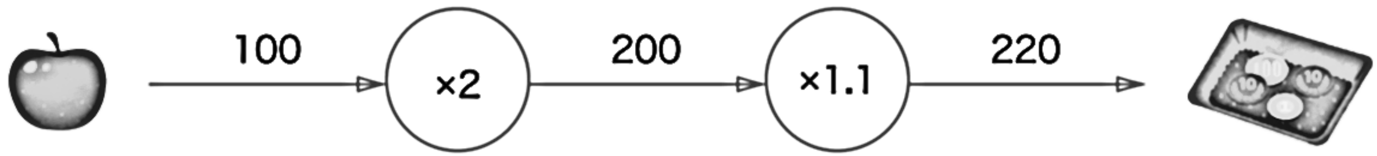
계산 그래프에 익숙해지기

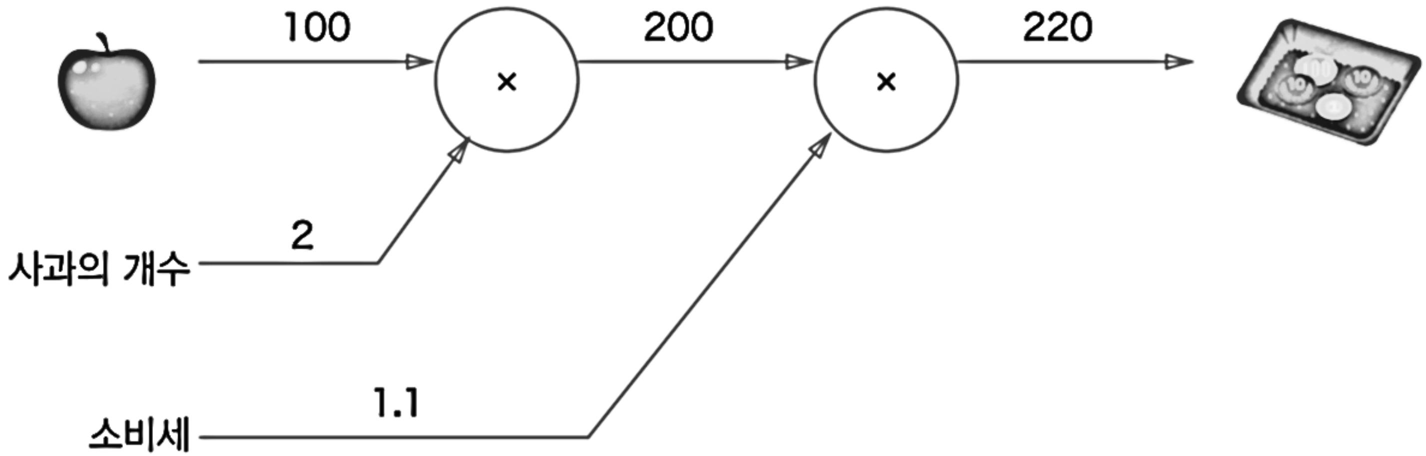
- ✓ 문제 1: 맹구는 슈퍼에서 1개에 100원인 사과를 2개 샀습니다. 이때 지불 금액을 구하세요. 단, 소비세가 10% 부과됩니다.



계산 그래프에 익숙해지기

- ✓ **문제 1: 맹구는 슈퍼에서 1개에 100원인 사과를 2개 샀습니다. 이때 지불 금액을 구하세요. 단, 소비세가 10% 부과됩니다.**



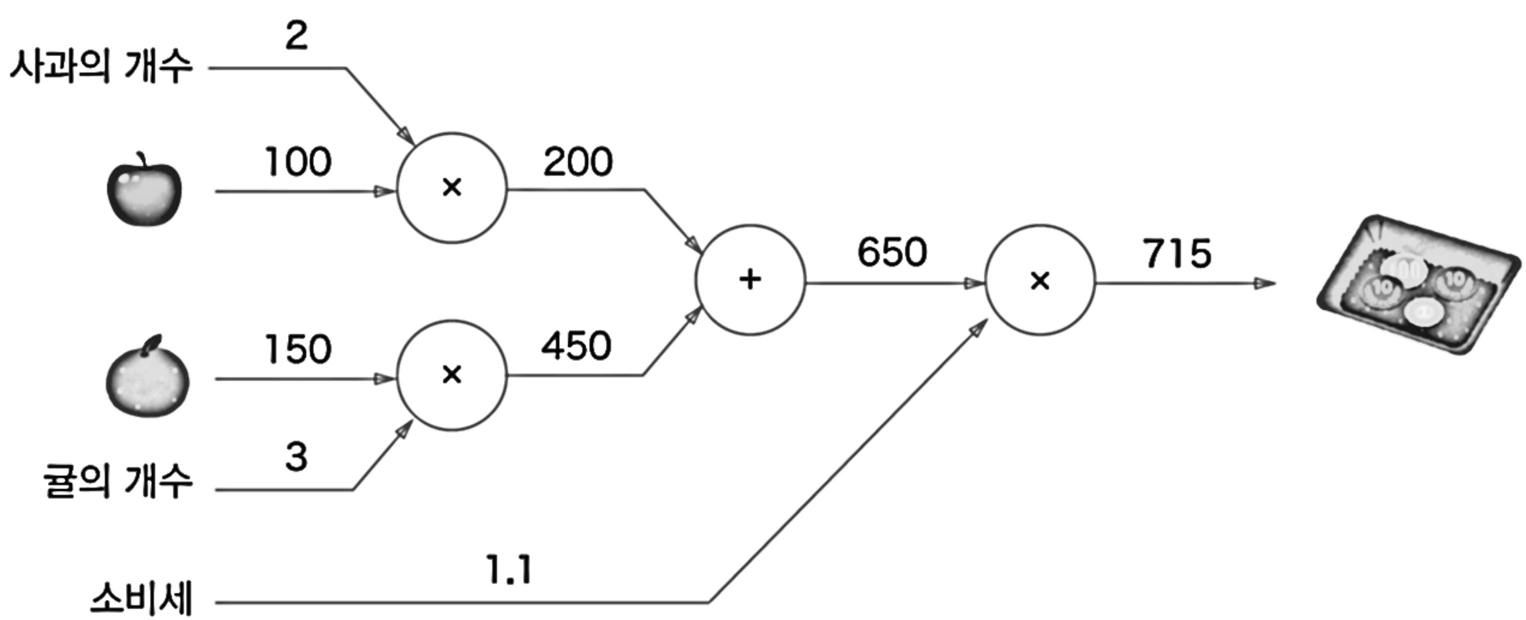


계산 그래프에 익숙해지기

- ✓ 문제 2: 맹구는 슈퍼에서 사과를 2개, 귤을 3개 샀습니다. 사과는 1개에 100원, 귤은 1개에 150원입니다. 소비세가 10%일 때 치를 금액을 구하세요.

계산 그래프에 익숙해지기

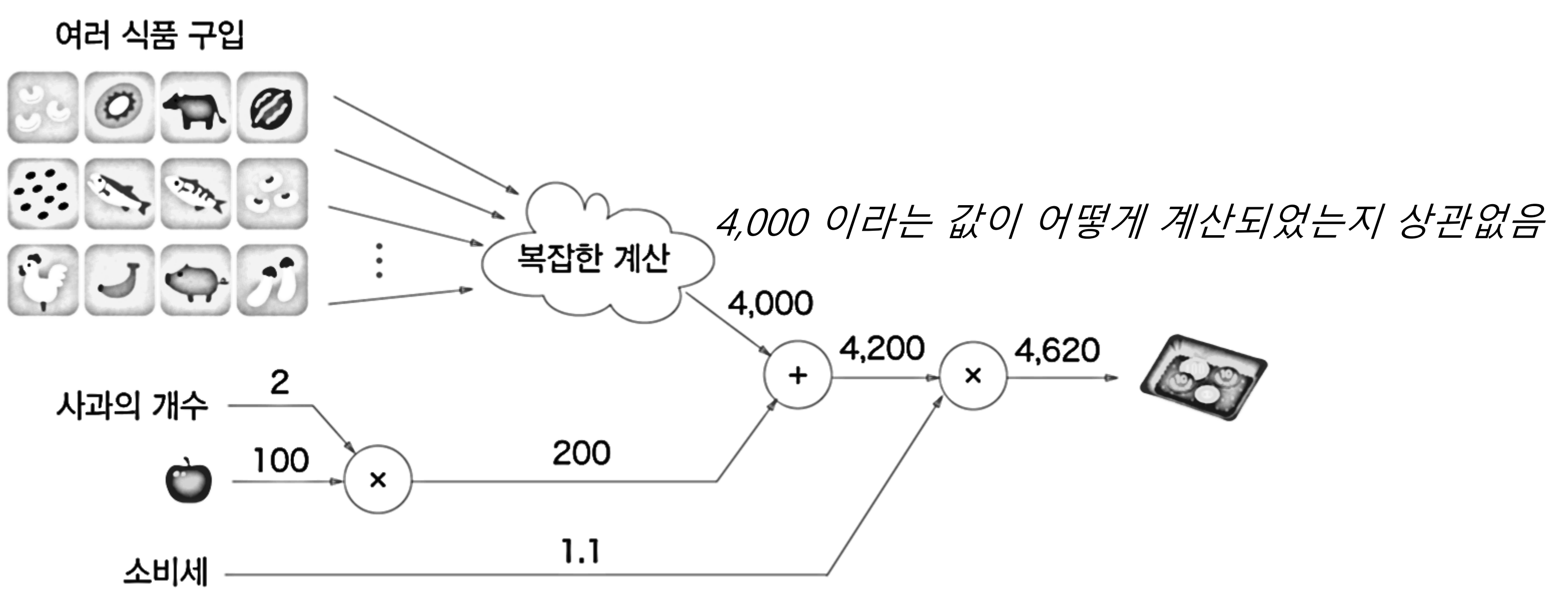
- ✓ **문제 2: 맹구는 슈퍼에서 사과를 2개, 귤을 3개 샀습니다. 사과는 1개에 100원, 귤은 1개에 150원입니다. 소비세가 10%일 때 치를 금액을 구하세요.**



국소적 계산

✓ 계산 그래프

- 국소적 계산 결과를 전파함으로써 최종 결과를 얻음
    - 전체에서 어떤 일이 벌어지든 상관없이 자신과 관계된 정보만으로 충분
- 국소적
    - 자신과 직접 관련된 작은 범위



왜 계산 그래프인가?

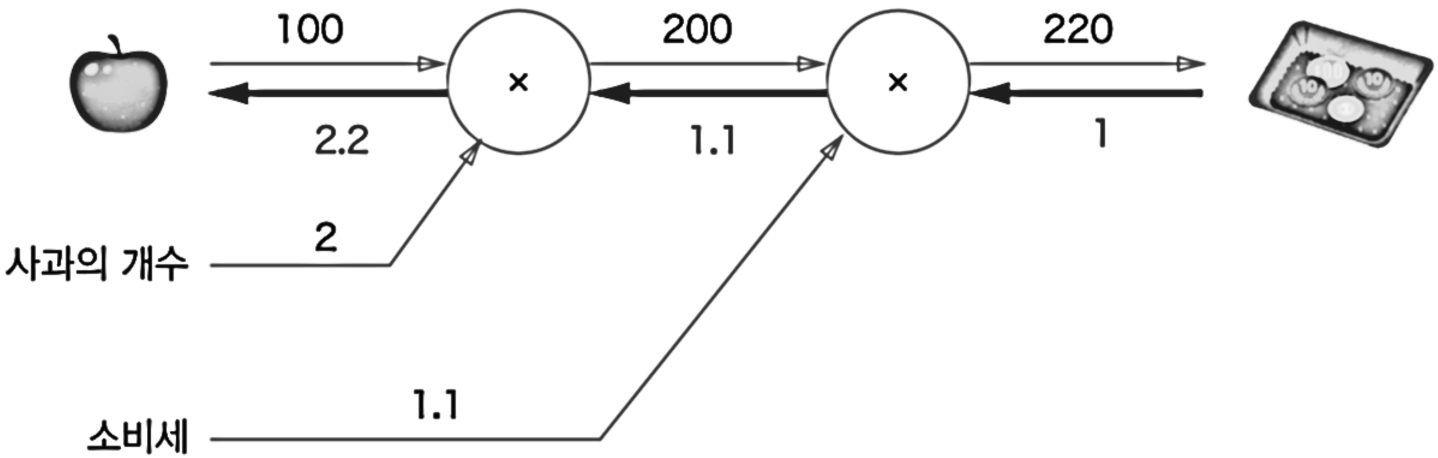
✓ **계산 그래프의 이점**

- 국소적 계산
- 중간 계산 결과 보관
- **역전파**를 통해 ‘**미분**’을 효율적으로 계산

왜 계산 그래프인가?

✓ 역전파에 의한 미분 값의 전달

- '국소적 미분'을 전달



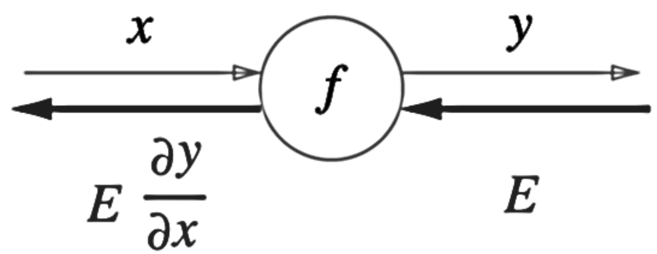
- 사과가 1원 오르면 ➔ 총 금액은 2.2원 오른다
  - 사과 값이 아주 조금 오르면 총 금액은 그 아주 작은 값의 2.2배만큼 증가

연쇄 법칙(Chain Rule)

✓ **계산 그래프에서 국소적 미분이 역전파로 전달되는 원리**

✓ **계산 그래프의 역전파**

- 순방향과 반대 방향으로 **국소적 미분**을 **곱**한다



- 연쇄 법칙에 의해 가능

연쇄 법칙

✓ 합성 함수

- 여러 함수로 구성된 함수

$$z = (x+y)^2 \quad \Rightarrow \quad \begin{aligned} z &= t^2 \\ t &= x+y \end{aligned}$$

✓ 연쇄 법칙

- **합성 함수의 미분**은 합성 함수를 구성하는 **각 함수의 미분의 곱**으로 나타낼 수 있다.

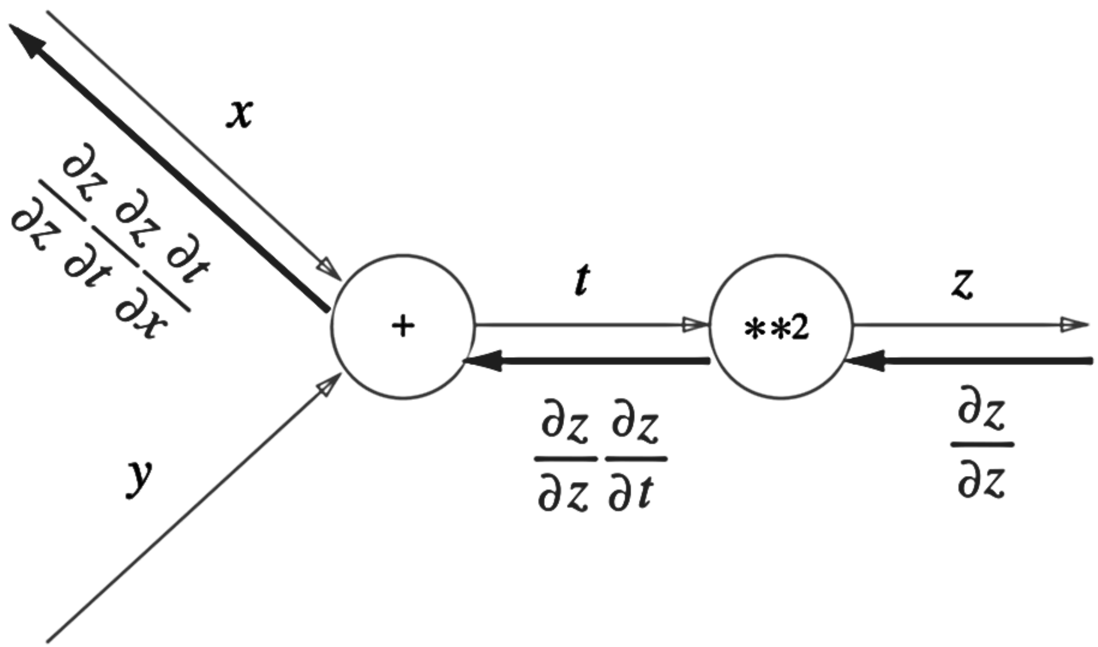
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial t}\frac{\partial t}{\partial x} \quad \Rightarrow \quad \begin{aligned} \frac{\partial z}{\partial t} &= 2t \\ \frac{\partial t}{\partial x} &= 1 \end{aligned}$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial t}\frac{\partial t}{\partial x} = 2t \cdot 1 = 2(x+y)$$

연쇄 법칙

✓ **연쇄 법칙과 계산 그래프**

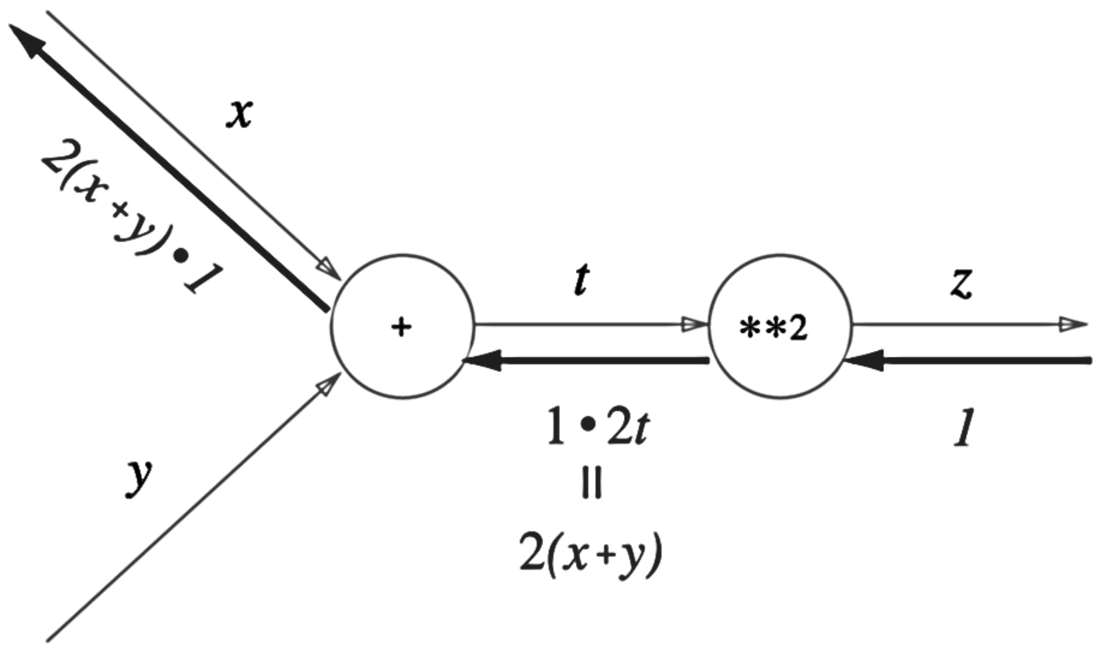
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial t}\frac{\partial t}{\partial x} = 2t \cdot 1 = 2(x+y)$$



연쇄 법칙

✓ **연쇄 법칙과 계산 그래프**

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial t}\frac{\partial t}{\partial x} = 2t \cdot 1 = 2(x+y)$$



역전파

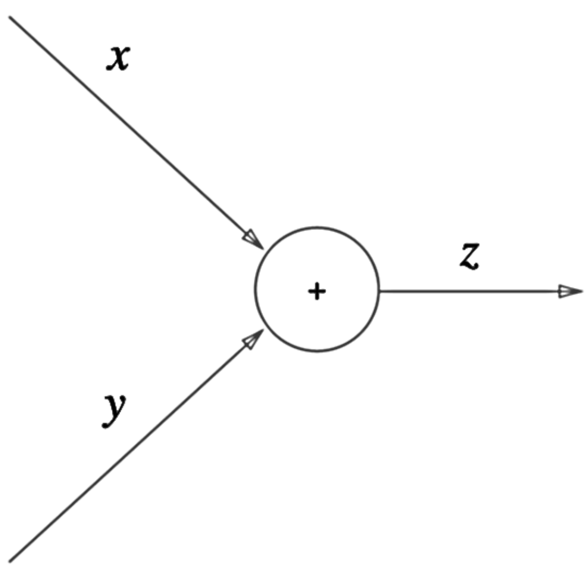
✓ 덧셈 노드의 역전파

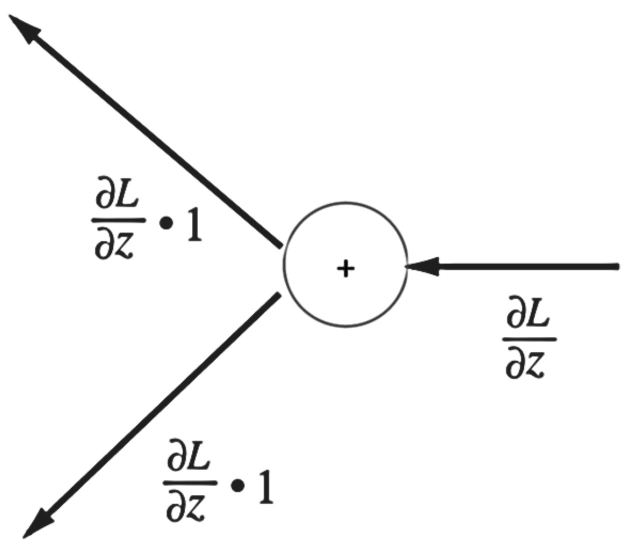
- 역전파된 미분값을 왼쪽으로 **그대로 전달**

- $z = x + y$의 미분

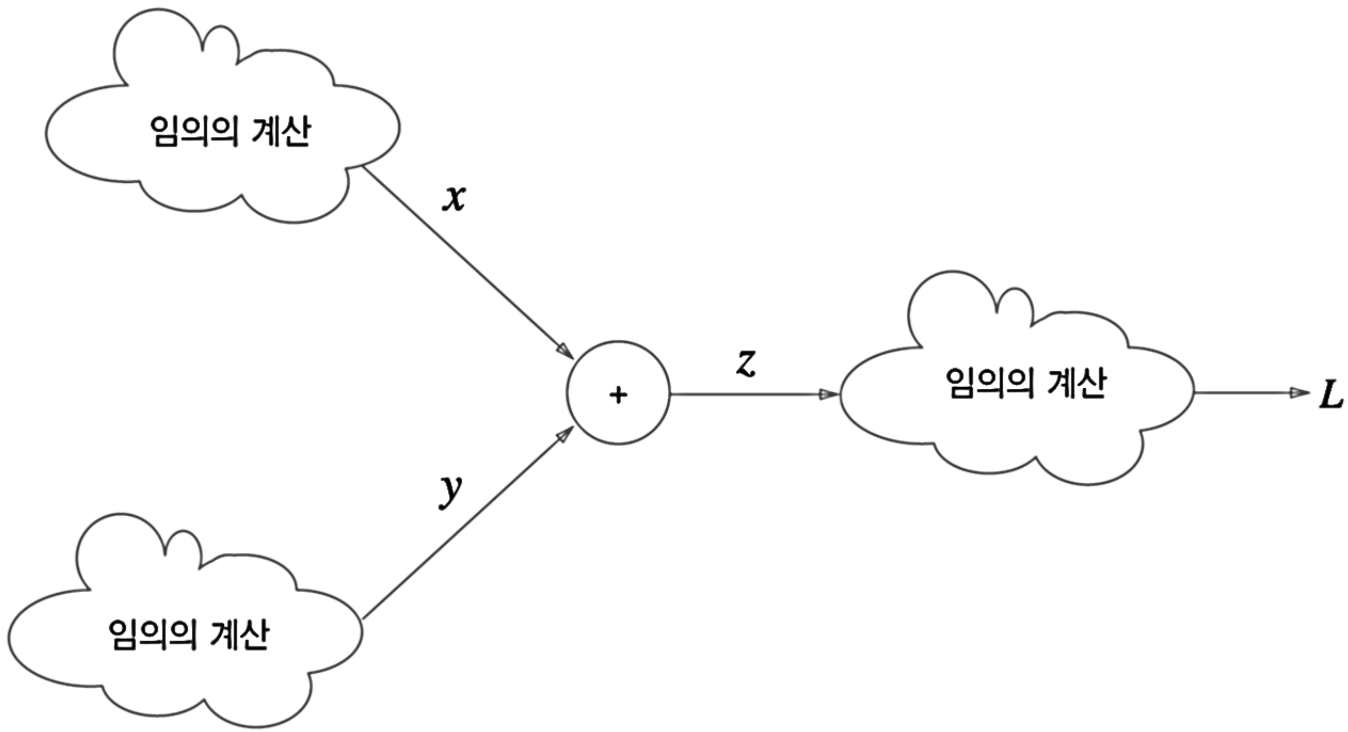
$$\frac{\partial z}{\partial x} = 1$$

$$\frac{\partial z}{\partial y} = 1$$



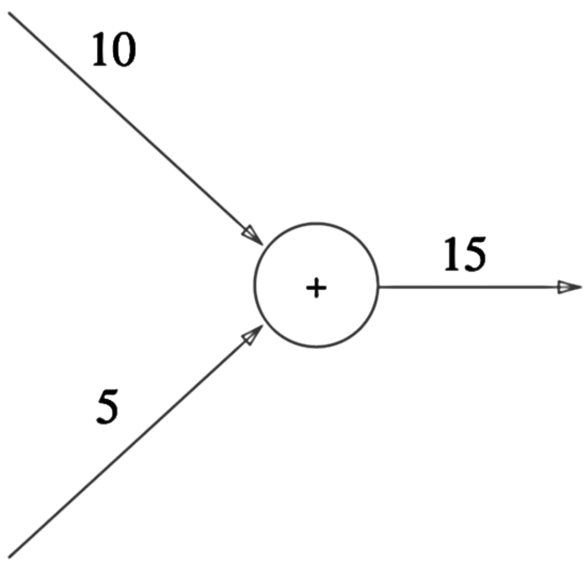


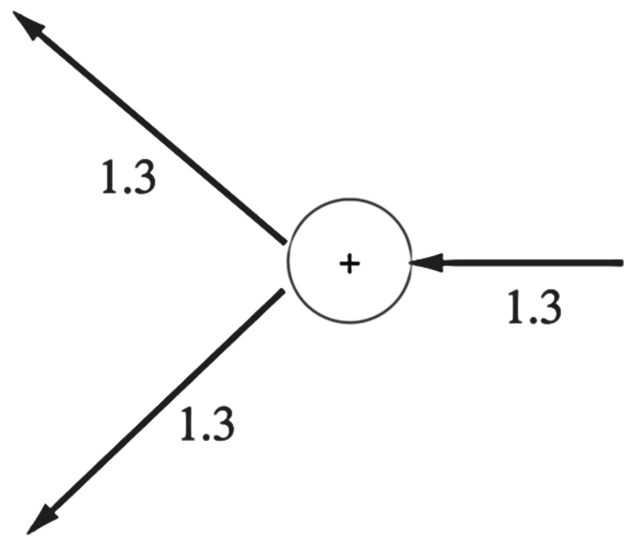
✓ 덧셈 노드의 역전파



역전파

✓ **덧셈 노드 역전파의 구체적 예**





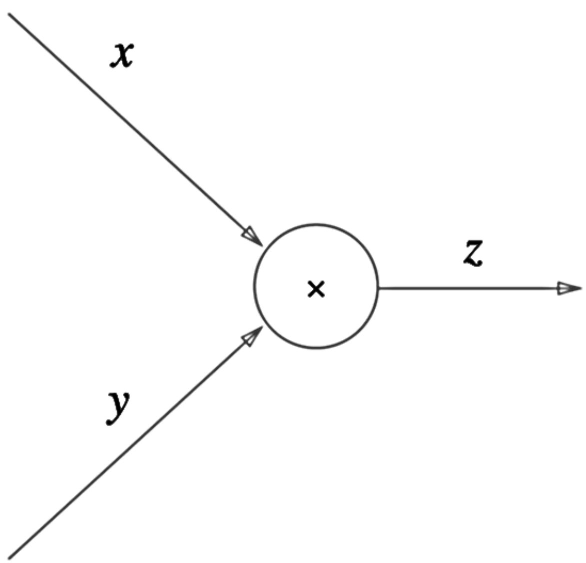
역전파

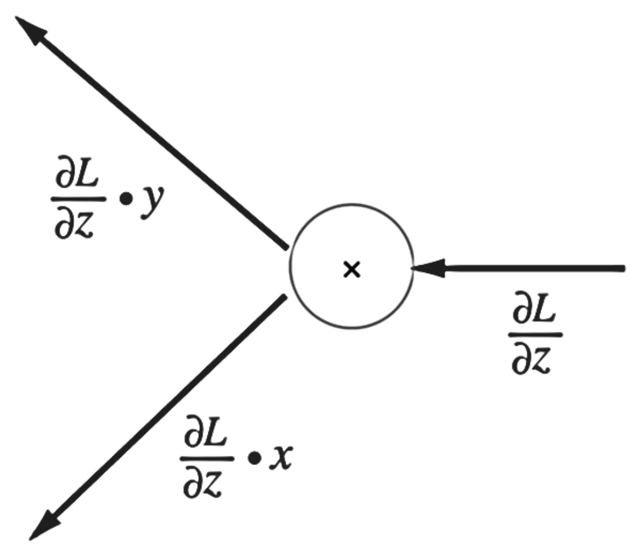
✓ 곱셈 노드의 역전파

- 역전파된 미분값에 순전파 때의 **입력 신호를 서로 바꾼 값**을 **곱**해서 왼쪽으로 전달
- $z = xy$의 미분

$$\frac{\partial z}{\partial x} = y$$

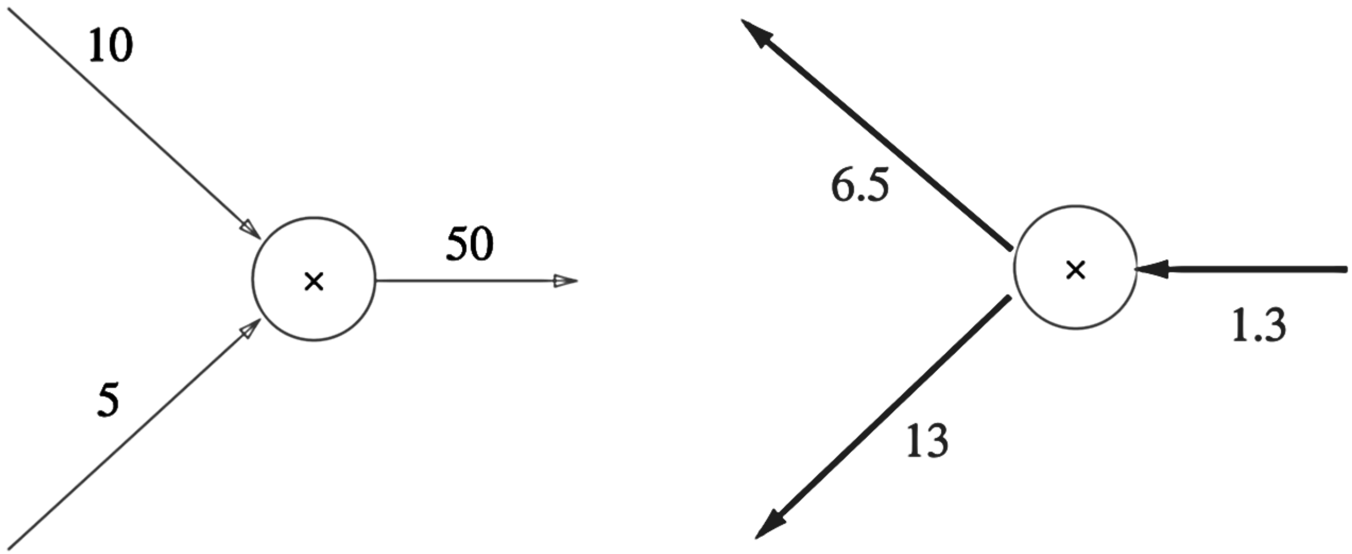
$$\frac{\partial z}{\partial y} = x$$



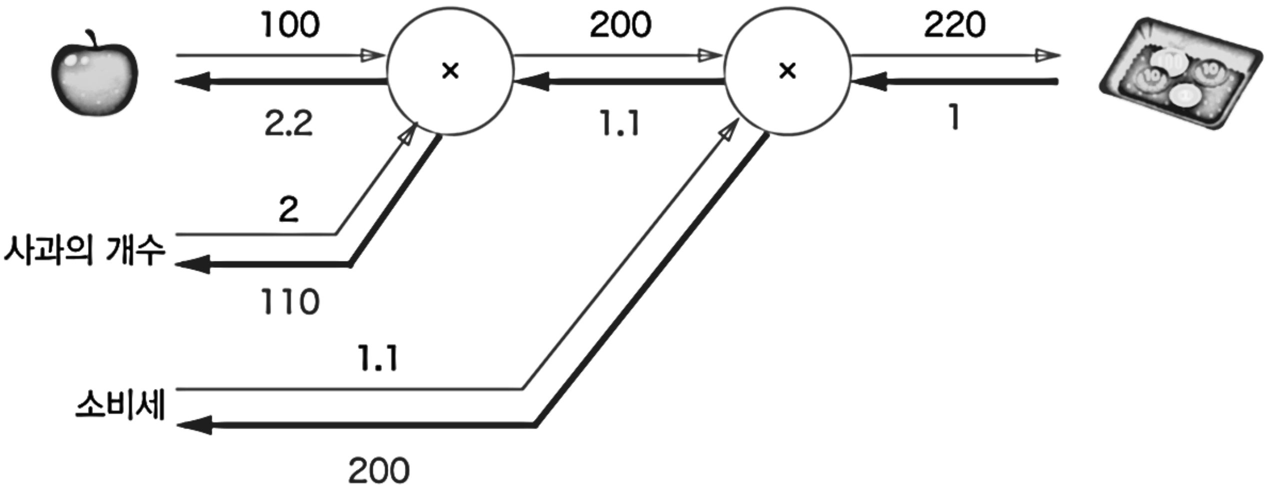


역전파

✓ **곱셈 노드 역전파의 구체적 예**



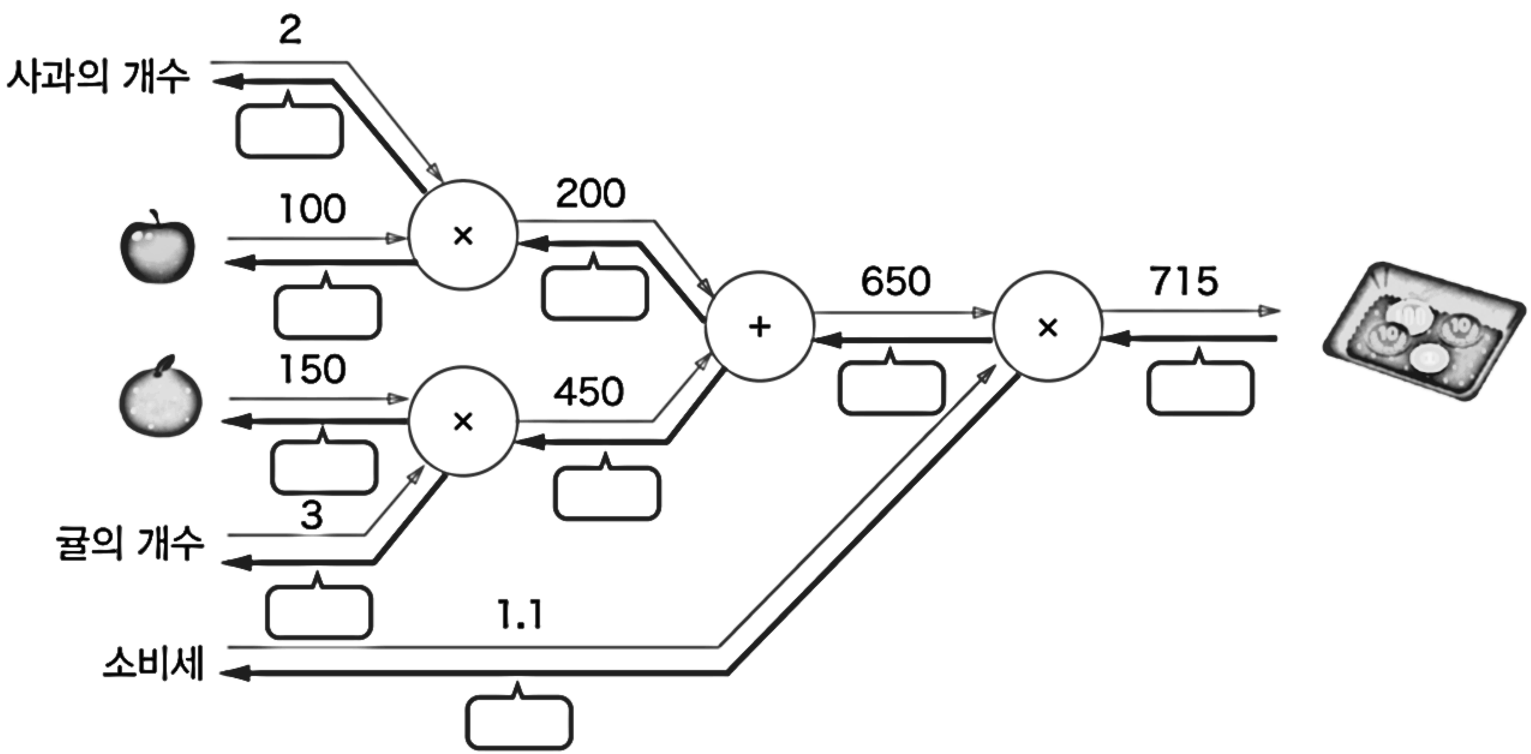
✓ 사과 쇼핑의 역전파 예



- 사과의 가격이 총 금액에 어떤 영향을 주는가?
  - 사과 가격에 대한 지불 금액의 미분
- 사과 개수에 대한 지불 금액의 미분
- 소비세에 대한 지불 금액의 미분

역전파

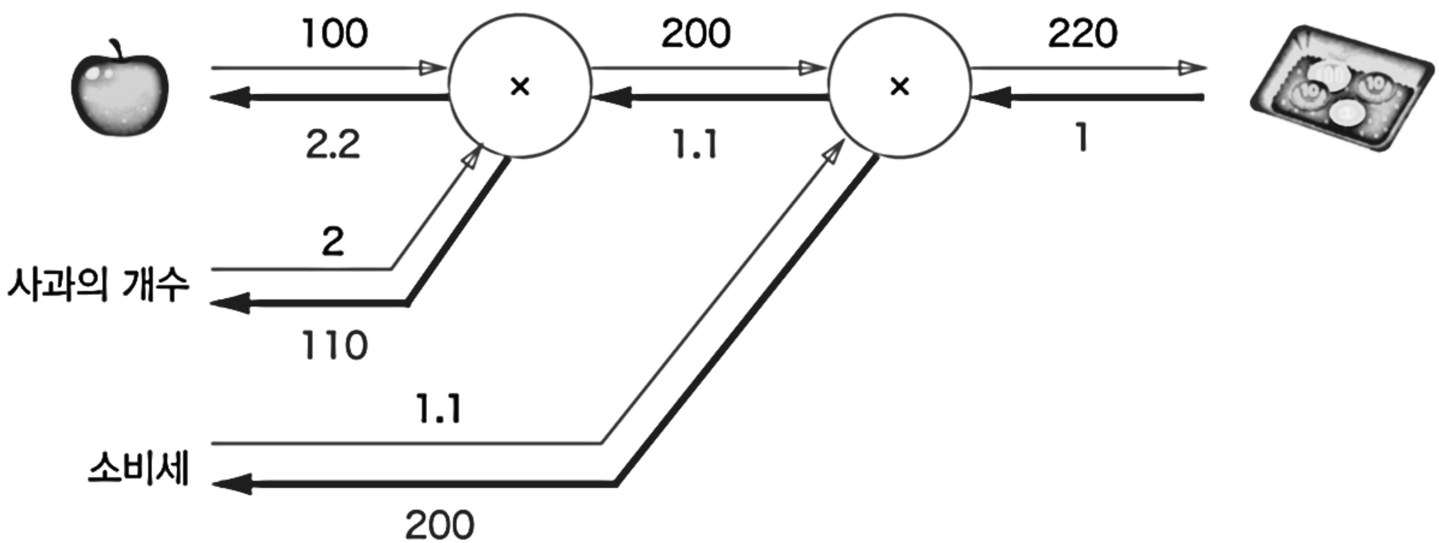
✓ **사과와 귤 쇼핑의 역전파 예**



계산 그래프 노드 구현

✓ 곱셈 노드

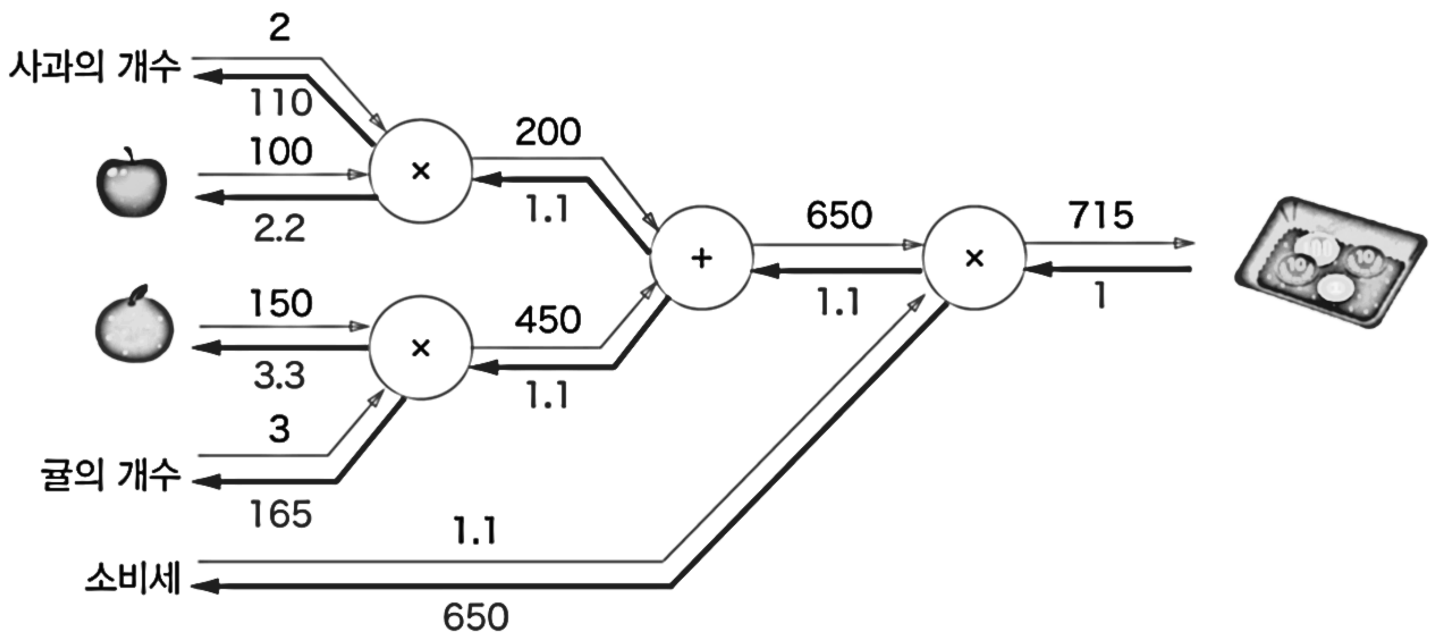
- MulLayer 클래스 (ch05/layer_naive.py)
  - 역전파 때 사용하기 위해 순전파 시 입력값을 저장
- 순전파, 역전파 (ch05/buy_apple.py)



계산 그래프 노드 구현

✓ 덧셈 노드

- AddLayer 클래스 (ch05/layer_naive.py)
  - 미분을 그대로 전파하므로 입력을 별도로 저장할 필요 없음
- 순전파, 역전파 (ch05/buy_apple_orange.py)

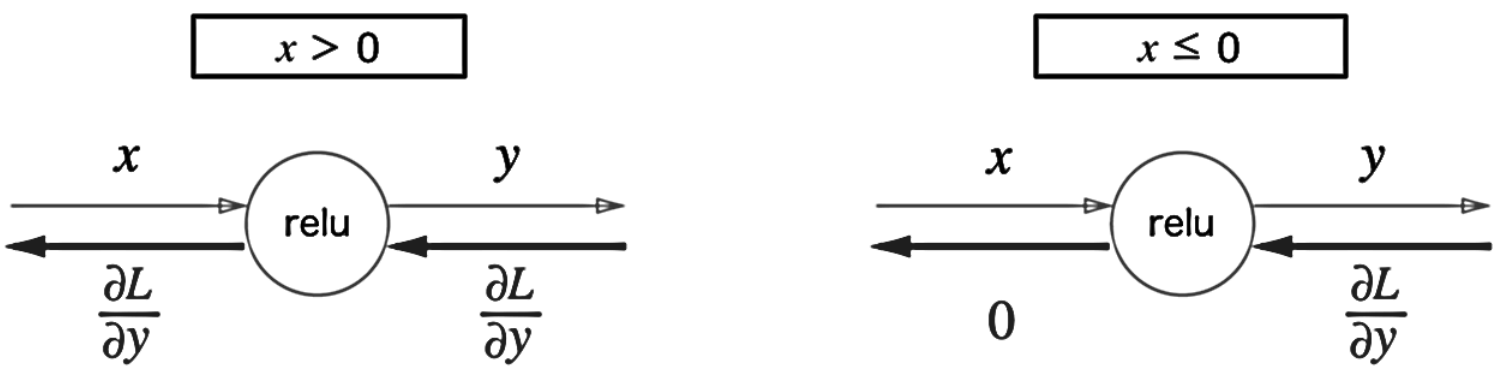


활성화 함수 계산 그래프 노드 구현

✓ ReLU 노드

- common/layers.py

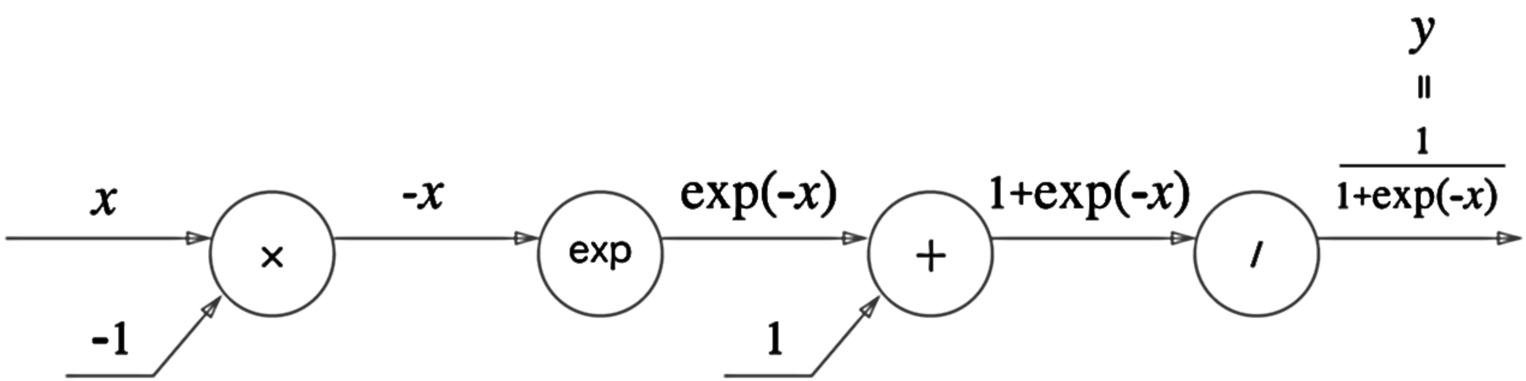
$$y = \begin{cases} x & (x > 0) \\ 0 & (x \leq 0) \end{cases} \qquad \frac{\partial y}{\partial x} = \begin{cases} 1 & (x > 0) \\ 0 & (x \leq 0) \end{cases}$$



활성화 함수 계산 그래프 노드 구현

✓ **Sigmoid 노드**

$$y = \frac{1}{1 + \exp(-x)}$$



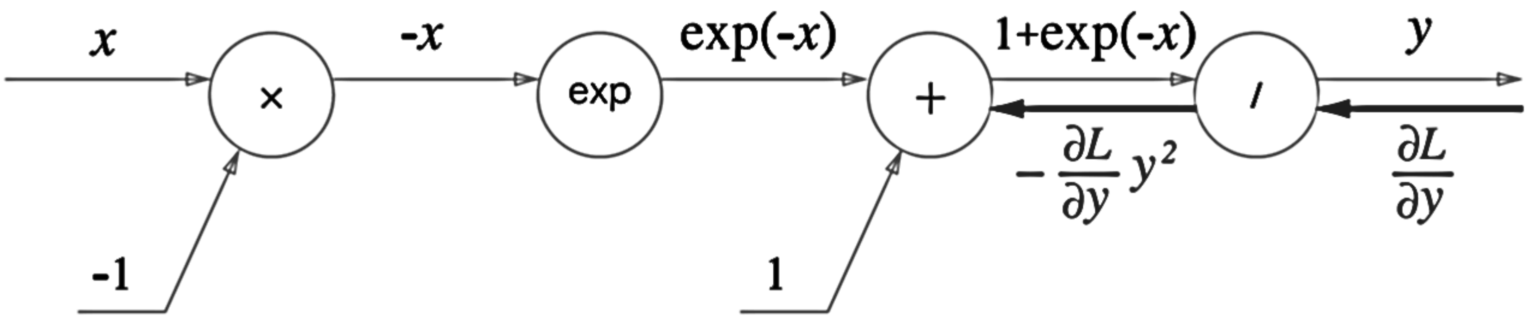
활성화 함수 계산 그래프 노드 구현

✓ **Sigmoid 노드**

- 1단계: / 노드

$$\frac{\partial y}{\partial x} = -\frac{1}{x^2}$$

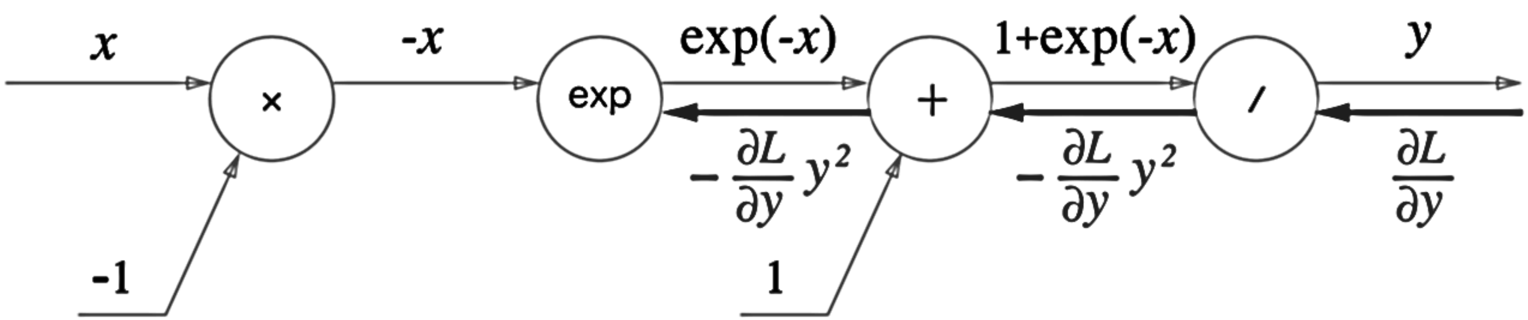
$$= -y^2$$



활성화 함수 계산 그래프 노드 구현

✓ Sigmoid 노드

- 2단계: + 노드
  - 상류의 값을 여과 없이 하류로

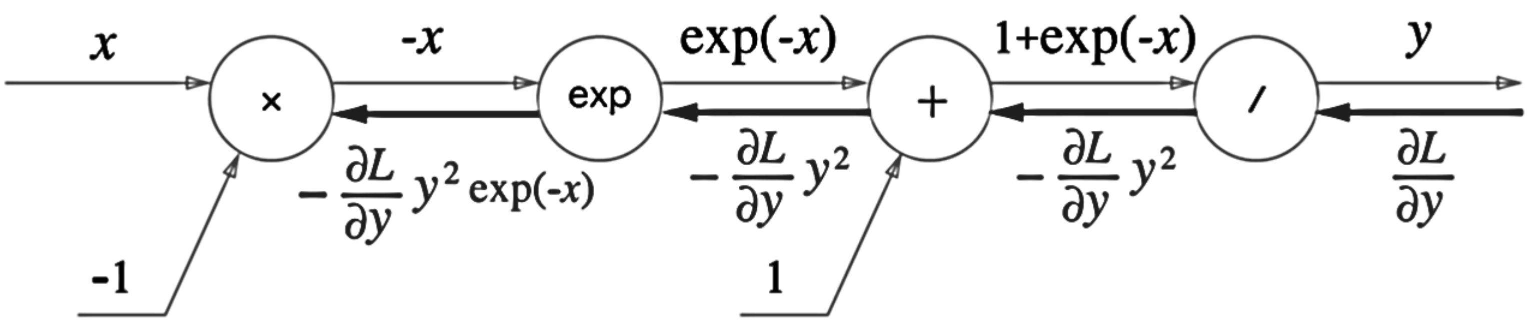


활성화 함수 계산 그래프 노드 구현

✓ **Sigmoid 노드**

- 3단계: exp 노드

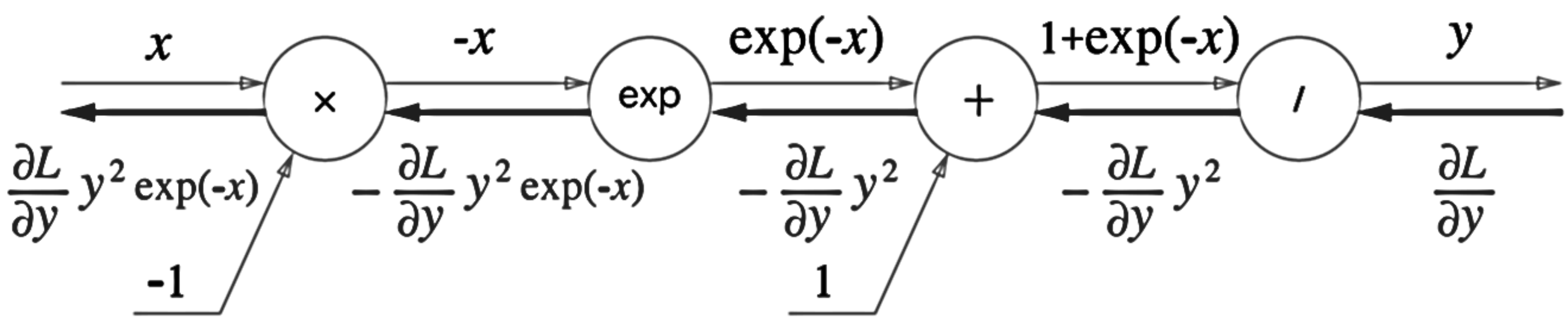
$$\frac{\partial y}{\partial x} = \exp(x)$$



활성화 함수 계산 그래프 노드 구현

✓ Sigmoid 노드

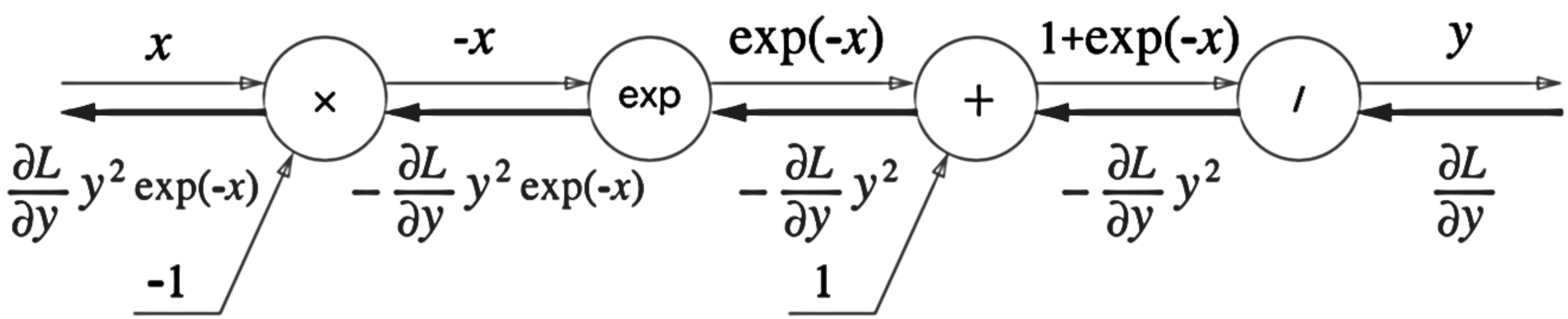
- 4단계: x 노드
  - 순전파 때의 값을 서로 바꿔 곱



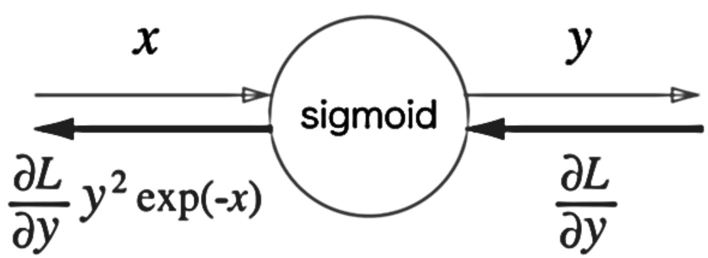
활성화 함수 계산 그래프 노드 구현

✓ Sigmoid 노드

- 4단계: x 노드
  - 순전파 때의 값을 서로 바꿔 곱



- 간소화 버전



활성화 함수 계산 그래프 노드 구현

✓ Sigmoid 노드

- $\frac{\partial L}{\partial y} y^2 e^{-x}$ 를 다음과 같이 정리
  - 순전파의 출력만으로 계산

$$\frac{\partial L}{\partial y} y^2 \exp(-x) = \frac{\partial L}{\partial y} \frac{1}{(1+\exp(-x))^2} \exp(-x)$$

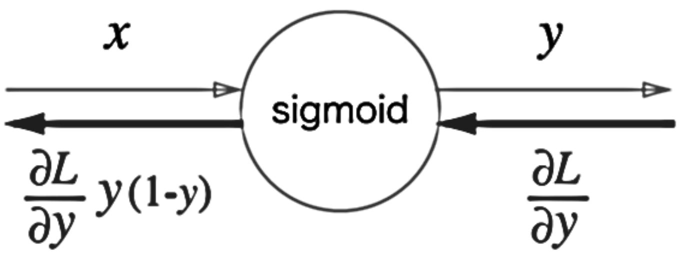
$$= \frac{\partial L}{\partial y} \frac{1}{1+\exp(-x)} \frac{\exp(-x)}{1+\exp(-x)}$$

$$= \frac{\partial L}{\partial y} y(1-y)$$

활성화 함수 계산 그래프 노드 구현

✓ **Sigmoid 노드**

- 최종 간소화 버전



- common/layers.py

Affine 계층을 계산 그래프로 구현

✓ Affine 계층

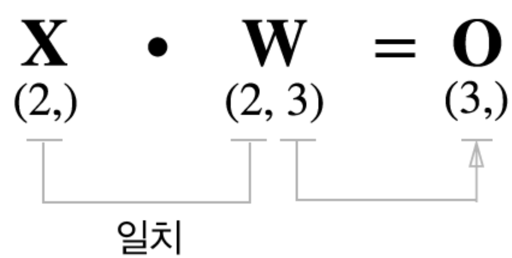
- Affine Transformation을 수행하는 신경망 계층(레이어)
  - 어파인 변환
    - 신경망의 순전파 때 수행하는 행렬의 내적을 기하학에서 일컫는 표현

- 신경망의 순전파 시 노드(뉴런)의 가중치 입력을 계산하기 위해 행렬의 내적 사용

- 이를 같은 신경망 계층의 모든 노드에 대해 표현하기 위해 입력의 행렬과 가중치 행렬을 정의
- 모든 노드의 가중치 입력을 계산하기 위해 입력의 행렬과 가중치 행렬에 대해 행렬 내적 수행

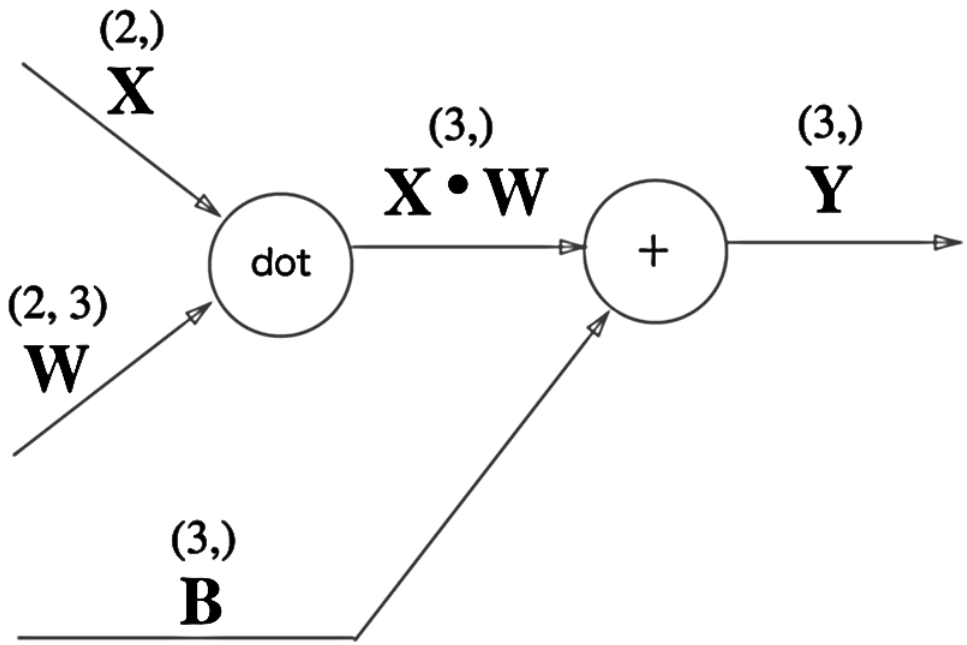
Affine 계층을 계산 그래프로 구현

```
>>> X = np.random.rand(2)    # 입력은 한 건임에 유의
>>> W = np.random.rand(2, 3) # 가중치
>>> B = np.random.rand(3)    # 바이어스

>>> X.shape # (2,)
>>> W.shape # (2, 3)
>>> B.shape # (3,)

# 어파인 계층을 구성하는 3개의 뉴런에 입력되는 가중치 합을 행렬을
# 이용해 한꺼번에 표현
>>> Y = np.dot(X, W) + B
```



Affine 계층을 계산 그래프로 구현

- ✓ **Y = np.dot(X, W) + B를 계산 그래프로 표현**
  - $\boldsymbol{X}, \boldsymbol{W}, \boldsymbol{B}$가 행렬(다차원 배열)임에 주의
    - 지금까지의 계산 그래프는 노드 사이에 '스칼라 값'이 흘렀지만
    - 어파인 계층에서는 '행렬'이 흐른다



Affine 계층을 계산 그래프로 구현

✓ **행렬 내적에 대한 역전파**

$$\frac{\partial L}{\partial \mathbf{X}} = \frac{\partial L}{\partial \mathbf{Y}} \cdot \mathbf{W}^{\mathrm{T}}$$

$$\frac{\partial L}{\partial \mathbf{W}} = \mathbf{X}^{\mathrm{T}} \cdot \frac{\partial L}{\partial \mathbf{Y}}$$

$$\mathrm{W} = \begin{pmatrix} w_{11} & w_{21} & w_{31} \\ w_{12} & w_{22} & w_{32} \end{pmatrix}$$

$$\mathrm{W}^{\mathrm{T}} = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{pmatrix}$$

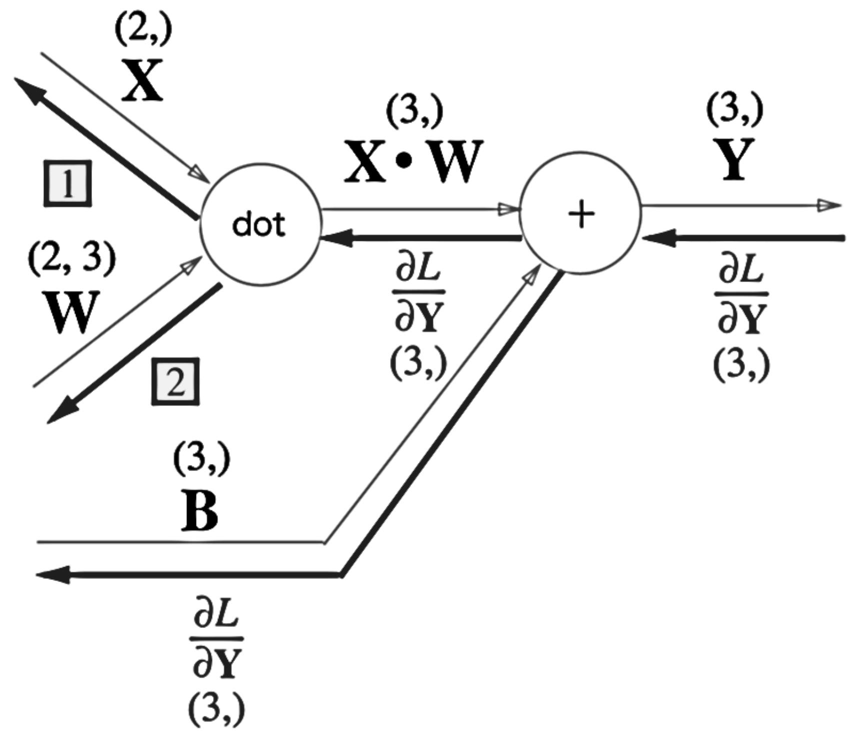
Affine 계층을 계산 그래프로 구현

✓ Affine 계층의 역전파

- 행렬의 형상을 주의 깊게 살펴보세요

$$\boxed{1}\quad \underset{(2,)}{\frac{\partial L}{\partial \mathbf{X}}} = \underset{(3,)}{\frac{\partial L}{\partial \mathbf{Y}}} \quad \underset{(3,\,2)}{\mathbf{W}^{\mathrm{T}}}$$

$$\boxed{2}\quad \underset{(2,\,3)}{\frac{\partial L}{\partial \mathbf{W}}} = \underset{(2,\,1)}{\mathbf{X}^{\mathrm{T}}} \quad \underset{(1,\,3)}{\frac{\partial L}{\partial \mathbf{Y}}}$$



$$\mathbf{X} = (x_0, x_1, \cdots, x_n)$$

$$\frac{\partial L}{\partial \mathbf{X}} = \left(\frac{\partial L}{\partial x_0}, \frac{\partial L}{\partial x_1}, \cdots, \frac{\partial L}{\partial x_n} \right)$$

$\boldsymbol{X}$와 $\frac{\partial L}{\partial \boldsymbol{X}}$의 형상이 같음에 주의

Affine 계층을 계산 그래프로 구현

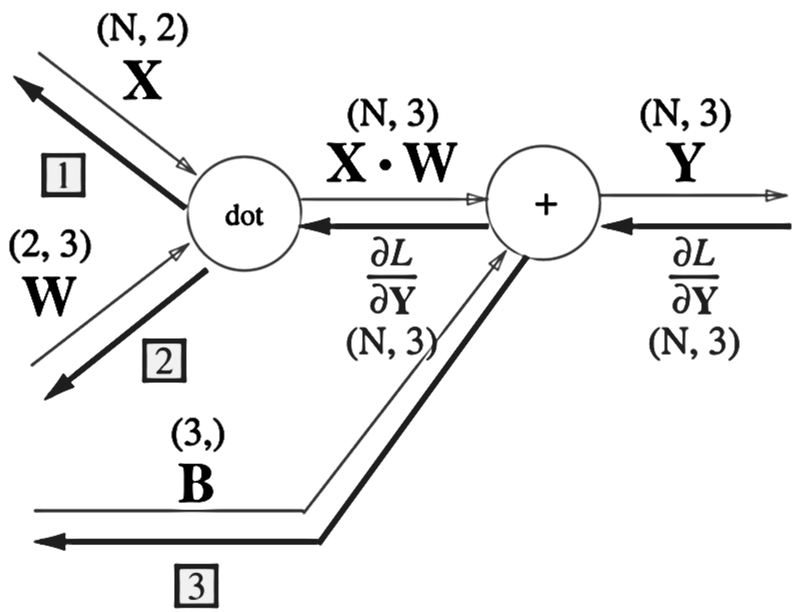
✓ 배치용 Affine 계층

- 지금까지는 입력 데이터 한 건만 고려
- 입력 데이터 N 개를 묶어서 배치 처리하는 경우 고려

1 $\underset{(N,\,2)}{\dfrac{\partial L}{\partial \mathbf{X}}} = \underset{(N,\,3)}{\dfrac{\partial L}{\partial \mathbf{Y}}} \cdot \underset{(3,\,2)}{\mathbf{W}^{\mathrm{T}}}$

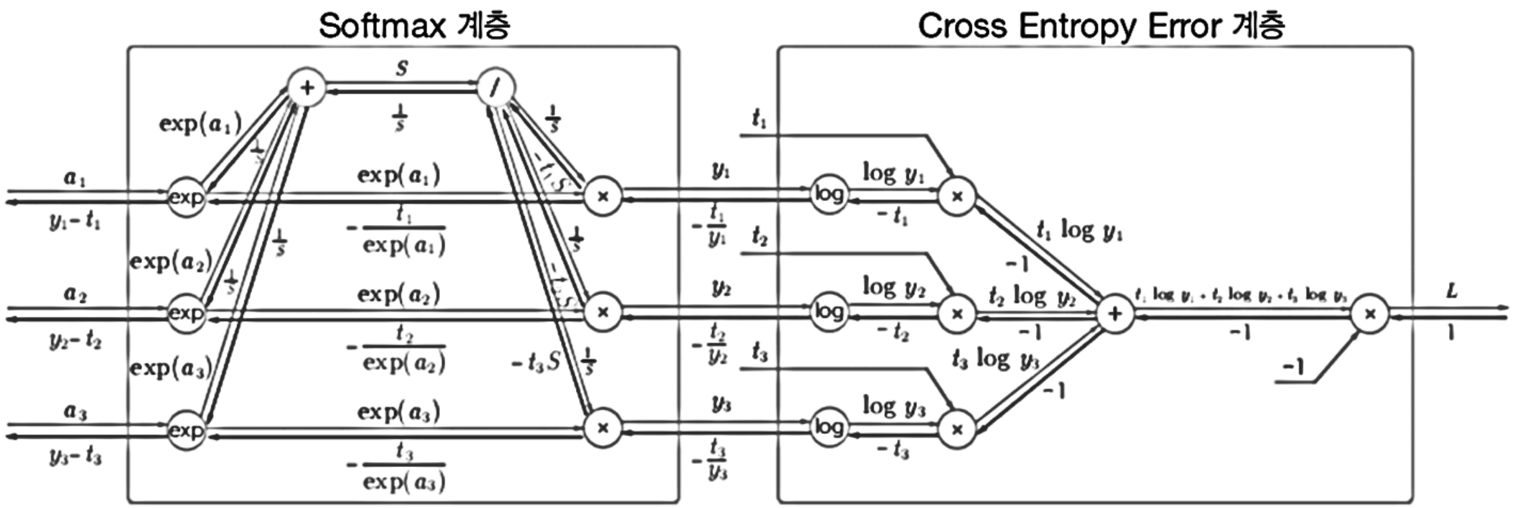
2 $\underset{(2,\,3)}{\dfrac{\partial L}{\partial \mathbf{W}}} = \underset{(2,\,N)}{\mathbf{X}^{\mathrm{T}}} \cdot \underset{(N,\,3)}{\dfrac{\partial L}{\partial \mathbf{Y}}}$

3 $\underset{(3)}{\dfrac{\partial L}{\partial \mathbf{B}}} = \underset{(N,\,3)}{\dfrac{\partial L}{\partial \mathbf{Y}}}$ 의 첫 번째 축(0축, 열방향)의 합



Softmax-with-Loss 계층

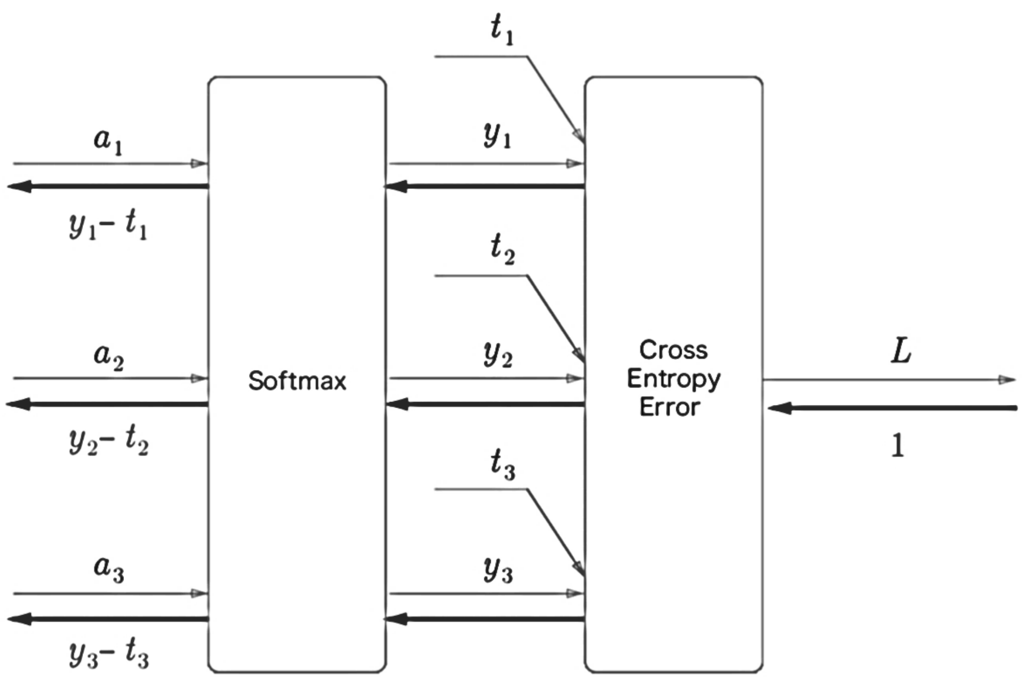
- ✓ **소프트맥스 함수에 손실 함수인 교차 엔트로피 오차도 포함**
  - 출력층에 사용



Softmax-with-Loss 계층

✓ 간략화 버전

- 소프트맥스의 역전파 결과가 깔끔한 것에 유의
  - 소프트맥스의 손실 함수로 교차 엔트로피 오차를 적용하였기 때문
  - 회귀 문제라면 출력의 활성화 함수로 '항등 함수', 손실 함수로 '평균 제곱 오차'를 사용하면 역시 깔끔한 결과를 얻을 수 있음



학습 알고리즘 구현하기

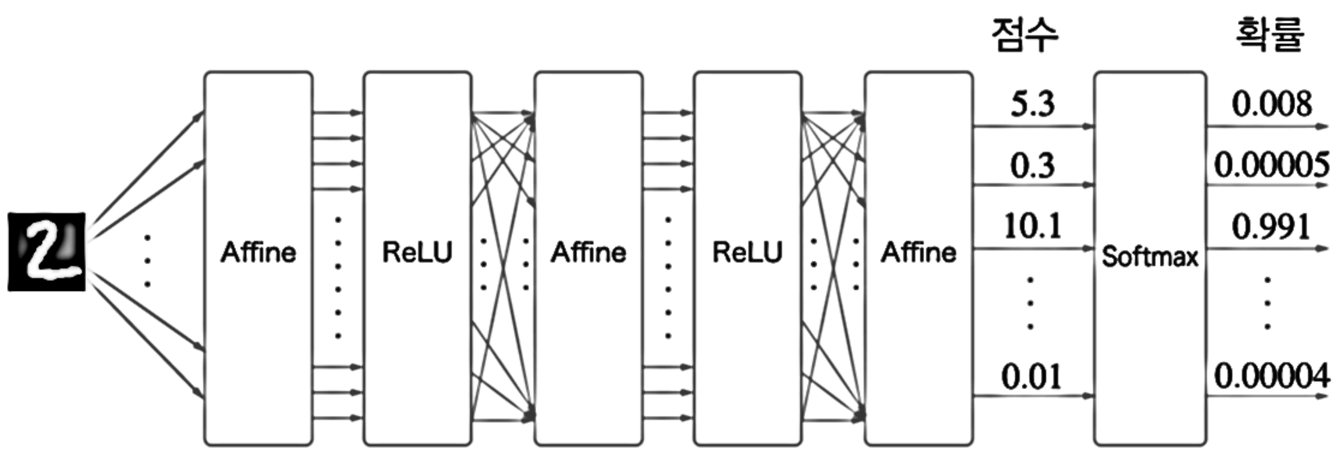
✓ **신경망 학습 절차**

- 전제
  - 신경망에는 적응 가능한 가중치와 편향(바이어스)이 있고, 이 가중치와 바이어스를 훈련 데이터에 적응하도록 조정하는 과정을 '학습'이라 한다. 신경망 학습은 다음과 같이 4단계로 수행한다.
- 1단계 - 미니 배치
  - 훈련 데이터 중 일부를 **무작위**로 가져온다.
  - 이렇게 선별된 데이터를 미니 배치라 하며, 이 미니 배치의 손실 함수 값을 줄이는 것이 목표다.
- 2단계 - 기울기 산출
  - 미니 배치의 손실 함수 값을 줄이기 위해 각 가중치 매개변수의 기울기를 구한다.
  - 기울기는 손실 함수의 값을 가장 작게 하는 방향을 제시한다.
- 3단계 - 매개변수 갱신
  - 가중치 매개변수를 기울기 방향으로 아주 조금 갱신한다.
- 4단계
  - 1~3 단계 반복

역전파 구현

✓ **ch05/two_layer_net.py**

- 신경망 계층을 OrderedDict에 보관하는 점 중요
  - 순서가 있는 딕셔너리
  - 순전파: 추가한 순서대로 각 계층의 forward() 메서드 호출로 충분
  - 역전파: 추가한 역순으로 각 계층의 backward() 메서드 호출로 충분
- 역전파 수행하려면 우선 순전파를 한 번 수행해야 함



역전파 검증

✓ Gradient Checking (기울기 확인)

- 역전파로 구한 기울기 검증하기

- 수치 미분으로 구한 기울기
  - 느리다
  - 정확하다
- 역전파로 구한 기울기
  - 빠르다
  - 버그 가능성

- 위 두 방식으로 구한 기울기가 일치하는지 확인하는 작업

  - ch05/gradient_check.py

역전파를 이용한 학습

- ✓ **기울기를 구할 때 수치 미분이 아닌 역전파법을 사용**
  - ch05/train_neuralnet.py

정리

- ✓ 계산 그래프를 이용하면 계산 과정을 시각적으로 파악할 수 있다.
- ✓ 계산 그래프의 노드는 국소적 계산으로 구성되며 국소적 계산을 조합하여 전체 계산을 구성한다.
- ✓ 계산 그래프의 순전파는 통상의 계산을 수행한다. 한편, 계산 그래프의 역전파로는 각 노드의 미분을 구할 수 있다.
- ✓ 신경망의 구성 요소를 계층으로 구현하여 기울기를 효율적으로 계산할 수 있다(역전파).
- ✓ 수치 미분과 역전파의 결과를 비교하면 역전파의 구현에 잘못이 없는지 확인할 수 있다(기울기 확인).