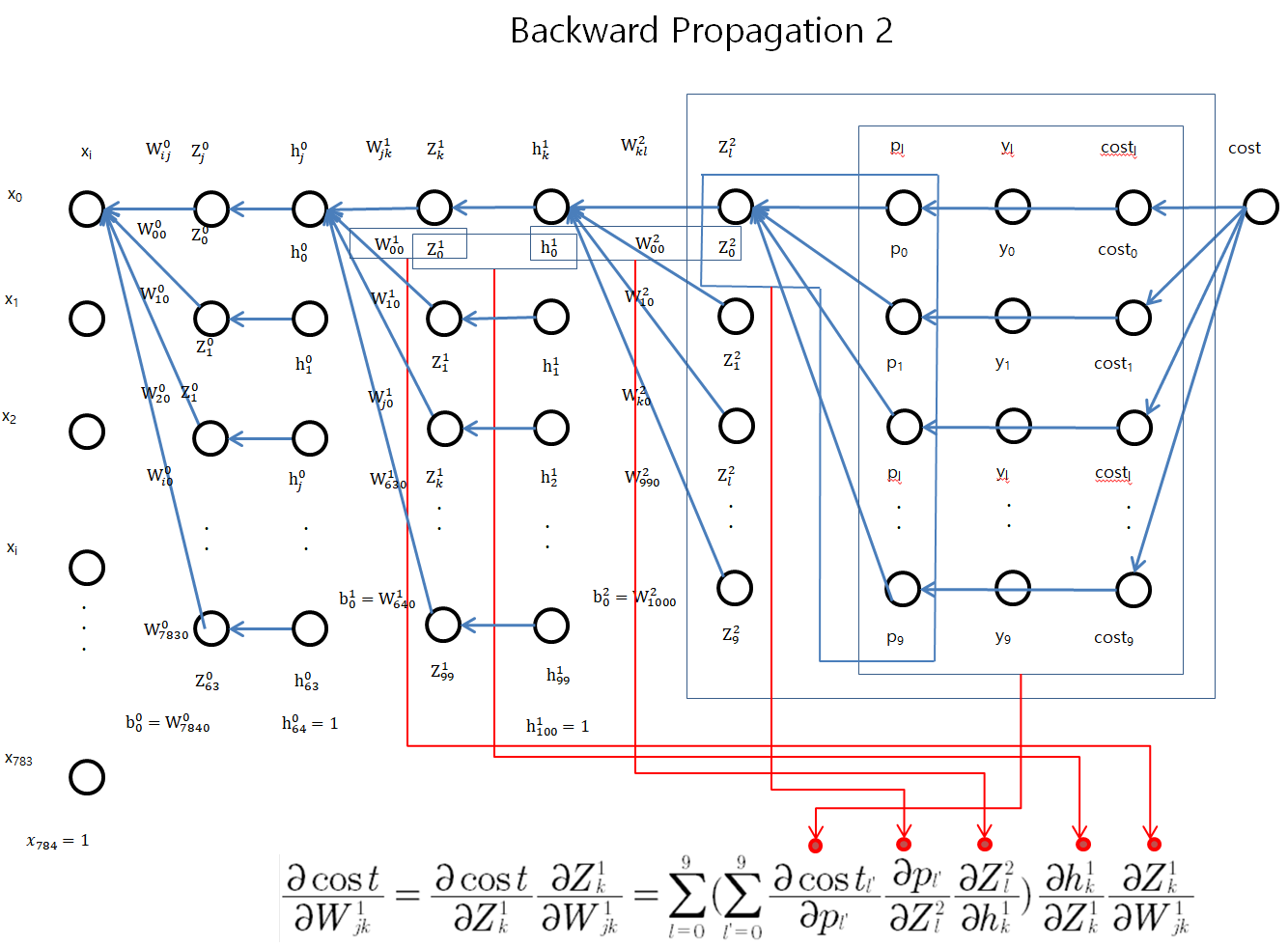
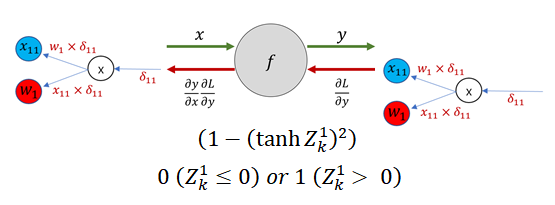
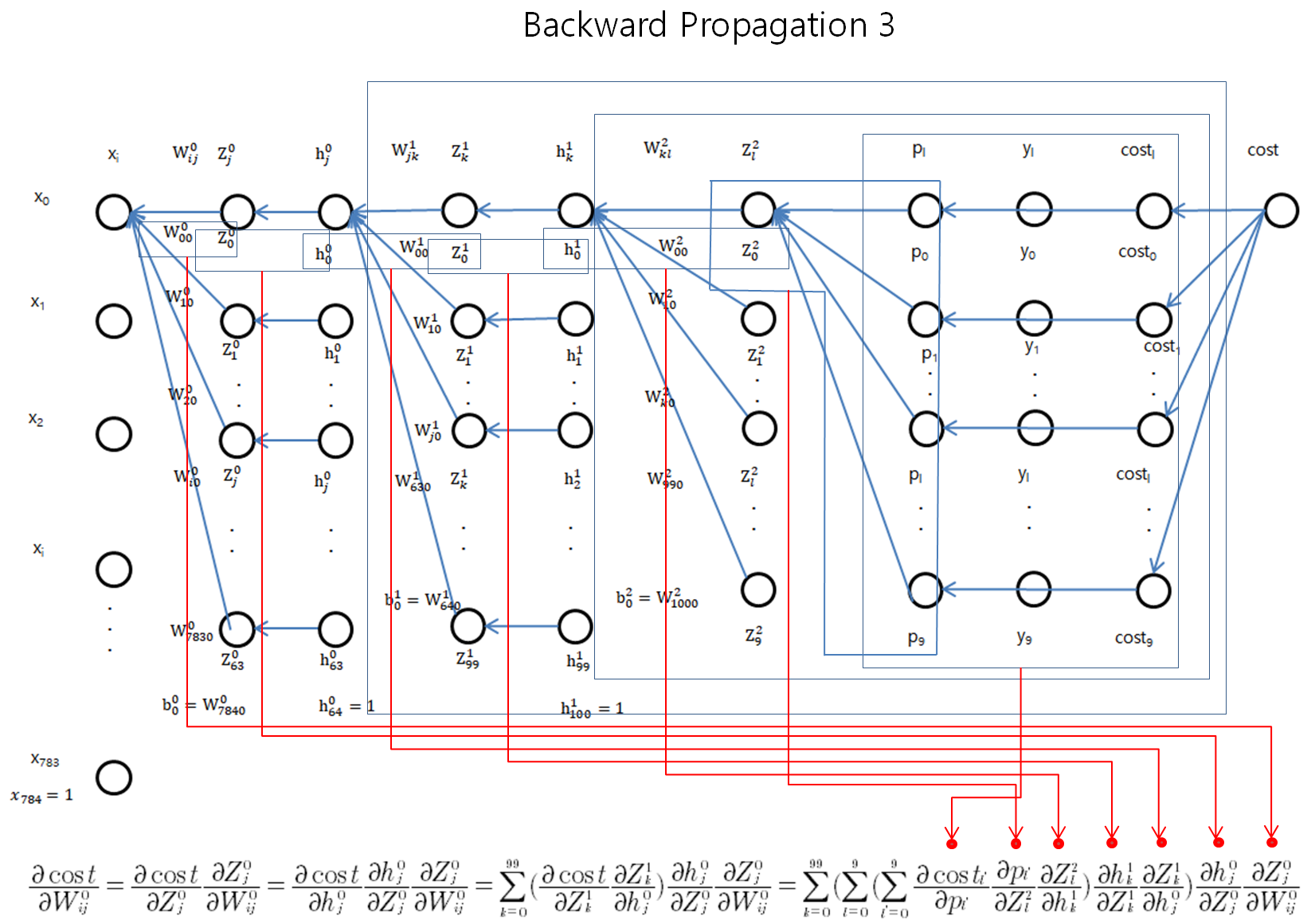


m.blog.naver.com/laonple/220527647084





---------------------------------------------------------------------------------------------------------------------------



참조: Kapathy의 계산그래프로 이해하기

<https://ratsgo.github.io/deep%20learning/2017/05/14/backprop/>

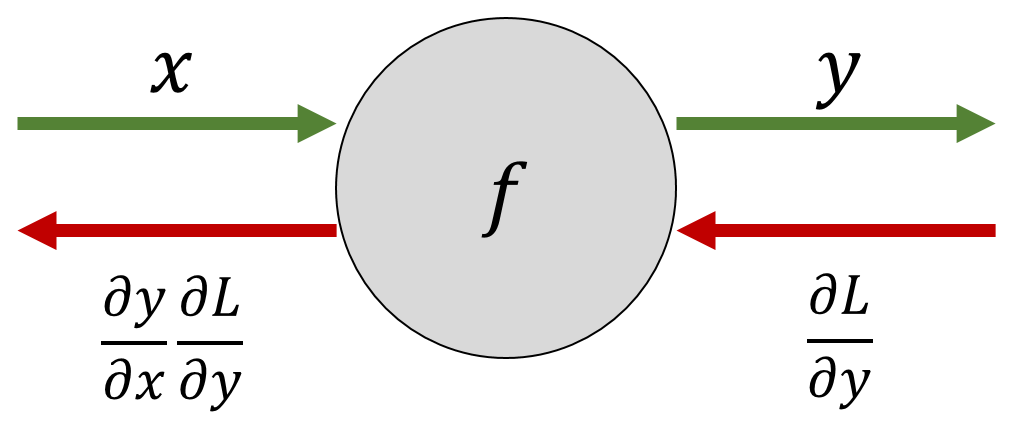
# 오차 역전파 (backpropagation)

#### 14 May 2017 | [backpropagation](https://ratsgo.github.io/blog/tags/" \l "backpropagation)

이번 글에서는 **오차 역전파법(backpropagation)**에 대해 살펴보도록 하겠습니다. 이번 글은 미국 스탠포드대학의 CS231n 강의를 기본으로 하되, 고려대학교 데이터사이언스 연구실의 김해동 석사과정이 쉽게 설명한 자료를 정리했음을 먼저 밝힙니다. 그럼 시작하겠습니다.

## 계산그래프와 chain rule

**계산그래프(computational graph)**는 계산과정을 그래프로 나타낸 것입니다. **노드(node, 꼭지점)**은 함수(연산), **엣지(edge, 간선)**는 값을 뜻합니다. y=f(x)y=f(x)를 나타내는 계산그래프는 아래 그림과 같습니다.

[](http://imgur.com/o8Q7slz)

계산그래프에서 계산을 왼쪽에서 오른쪽으로 진행하는 단계를 **순전파(forward propagation)**라고 합니다. 위 그림 기준으로는 녹색 화살표가 됩니다. 입력값 xx는 함수 ff를 거쳐 yy로 순전파되고 있는 점을 확인할 수 있습니다.

반대로 계산을 오른쪽에서 왼쪽으로 진행하는 단계를 **역전파(backward propagation)**라고 합니다. 빨간색 화살표가 역전파를 가리킵니다.

여기에서 ∂L/∂y∂L/∂y의 의미에 주목할 필요가 있습니다. 지금은 예시이기 때문에 노드를 하나만 그렸지만, 실제 뉴럴네트워크는 이러한 노드가 꽤 많은 큰 계산그래프입니다. 이 네트워크는 최종적으로는 정답과 비교한 뒤 **Loss**를 구합니다.

우리의 목적은 뉴럴네트워크의 오차를 줄이는 데 있기 때문에, 각 파라메터별로 Loss에 대한 그래디언트를 구한 뒤 그래디언트들이 향한 쪽으로 파라메터들을 업데이트합니다. ∂L/∂y∂L/∂y는 yy에 대한 Loss의 변화량, 즉 Loss로부터 흘러들어온 그래디언트라고 이해하면 좋을 것 같습니다.

이제는 현재 입력값 xx에 대한 Loss의 변화량, 즉 ∂L/∂x∂L/∂x를 구할 차례입니다. 이는 **미분의 연쇄법칙(chain rule)**에 의해 다음과 같이 계산할 수 있습니다.

#### ∂L∂x=∂y∂x∂L∂y∂L∂x=∂y∂x∂L∂y

이미 설명드렸듯 ∂L/∂y∂L/∂y는 Loss로부터 흘러들어온 그래디언트입니다. ∂y/∂x∂y/∂x는 현재 입력값에 대한 현재 연산결과의 변화량, 즉 **로컬 그래디언트(Local Gradient)**입니다.

다시 말해 현재 입력값에 대한 Loss의 변화량은 Loss로부터 흘러들어온 그래디언트에 로컬 그래디언트를 곱해서 구한다는 이야기입니다. 이 그래디언트는 다시 앞쪽에 배치돼 있는 노드로 역전파됩니다.

## 덧셈 노드

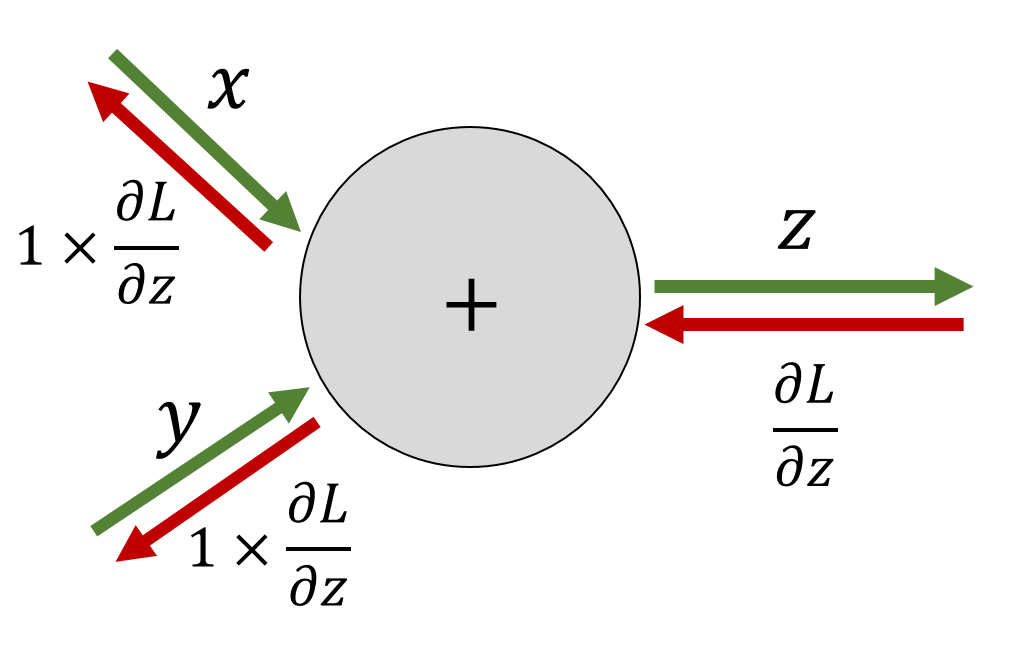
덧셈 노드의 수식은 아래와 같습니다.

#### z=f(x,y)=x+yz=f(x,y)=x+y

덧셈 노드의 로컬 그래디언트는 아래와 같습니다.

#### ∂z∂x=∂(x+y)∂x=1∂z∂y=∂(x+y)∂y=1∂z∂x=∂(x+y)∂x=1∂z∂y=∂(x+y)∂y=1

덧셈 노드의 계산그래프는 아래와 같습니다. 현재 입력값에 대한 Loss의 변화량은 로컬 그래디언트에 흘러들어온 그래디언트를 각각 곱해주면 됩니다. 덧셈 노드의 역전파는 흘러들어온 그래디언트를 그대로 흘려보내는 걸 확인할 수 있습니다.

[](http://imgur.com/phMalGK)

## 곱셈 노드

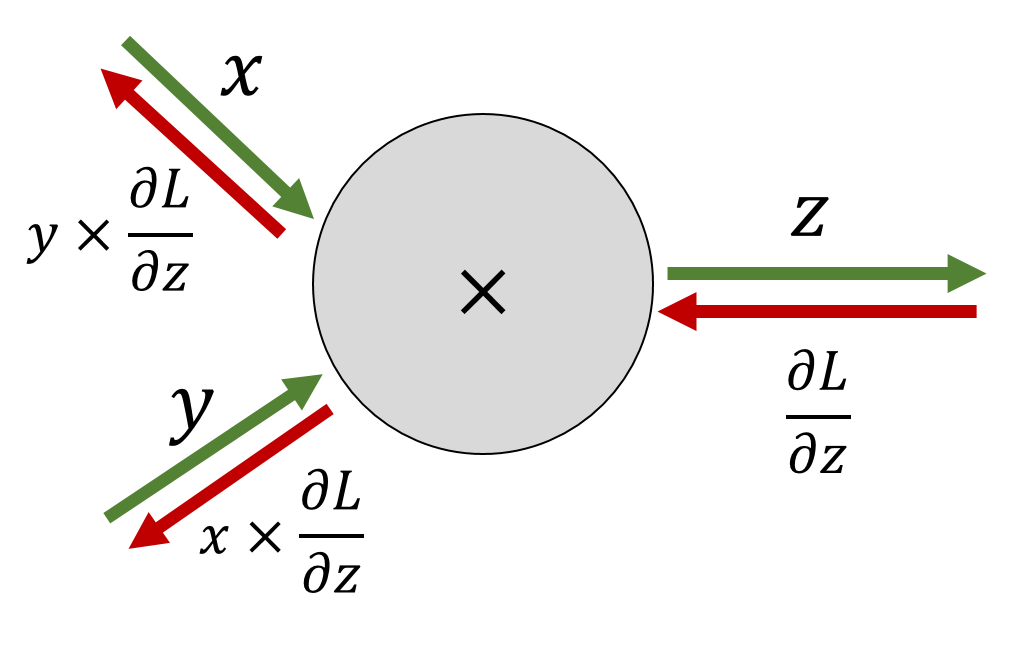
곱셈 노드의 수식은 아래와 같습니다.

#### z=f(x,y)=xyz=f(x,y)=xy

곱셈 노드의 로컬 그래디언트는 아래와 같습니다.

#### ∂z∂x=∂(xy)∂x=y∂z∂y=∂(xy)∂y=x∂z∂x=∂(xy)∂x=y∂z∂y=∂(xy)∂y=x

곱셈 노드의 계산그래프는 아래와 같습니다. 현재 입력값에 대한 Loss의 변화량은 로컬 그래디언트에 흘러들어온 그래디언트를 각각 곱해주면 됩니다. 곱셈 노드의 역전파는 순전파 때 입력 신호들을 서로 바꾼 값을 곱해서 하류로 흘려보내는 걸 확인할 수 있습니다.

[](http://imgur.com/5BycHN1)

## ReLU 노드

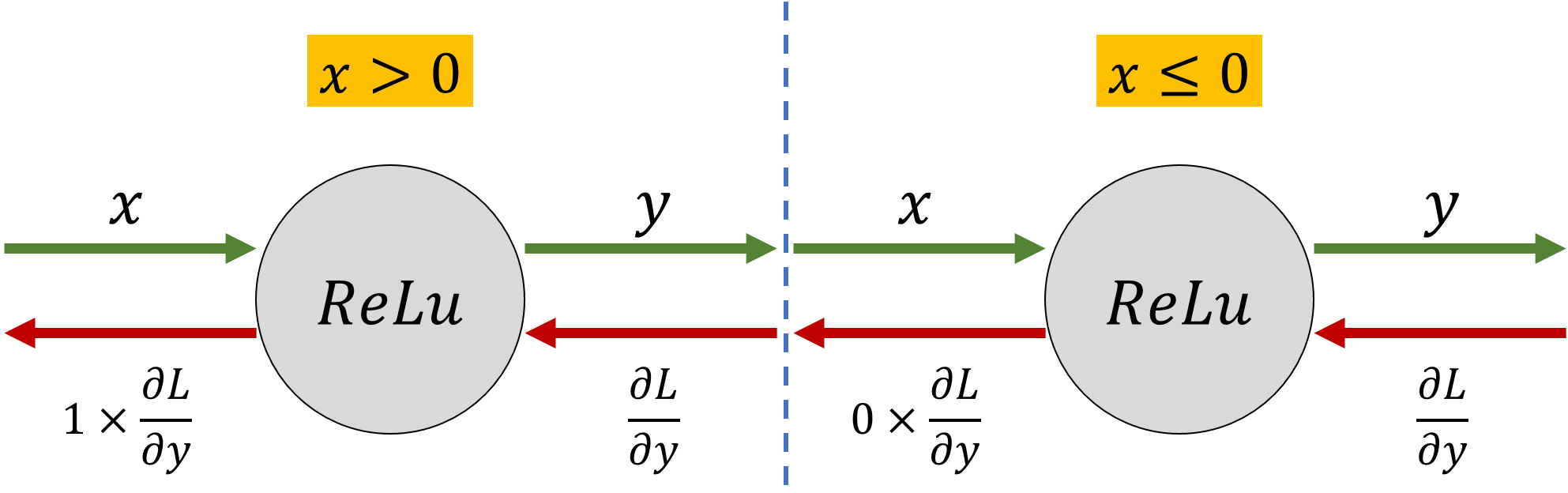
**활성화함수(activation function)**로 사용되는 **ReLU**는 다음 식처럼 정의됩니다.

#### y=x(x>0)y=0(x≤0)y=x(x>0)y=0(x≤0)

ReLU 노드의 로컬 그래디언트는 아래와 같습니다.

#### ∂y∂x=1(x>0)∂y∂x=0(x≤0)∂y∂x=1(x>0)∂y∂x=0(x≤0)

계산그래프는 아래와 같습니다.

[](http://imgur.com/FrxDrr5)

## Sigmoid 노드

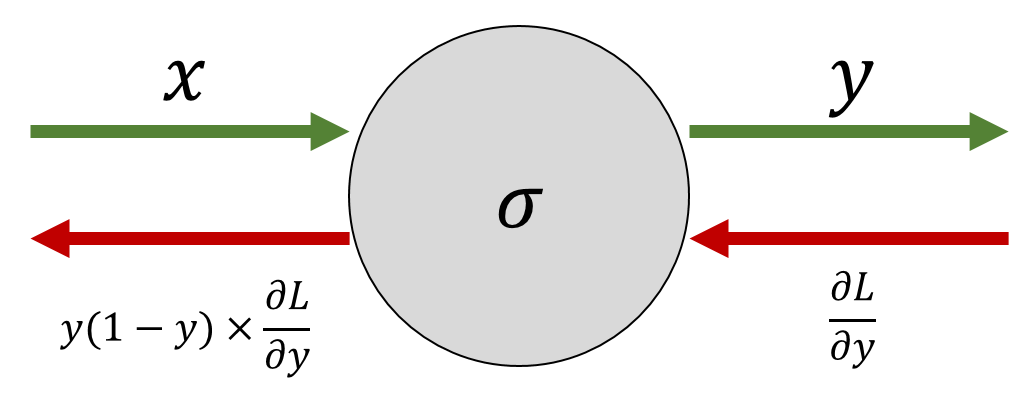
**시그모이드(sigmoid)** 함수는 아래와 같이 정의됩니다.

#### y=11+exp(−x)y=11+exp(−x)

시그모이드 노드의 로컬 그래디언트는 다음과 같습니다.

#### ∂y∂x=y(1−y)∂y∂x=y(1−y)

계산그래프는 아래와 같습니다.

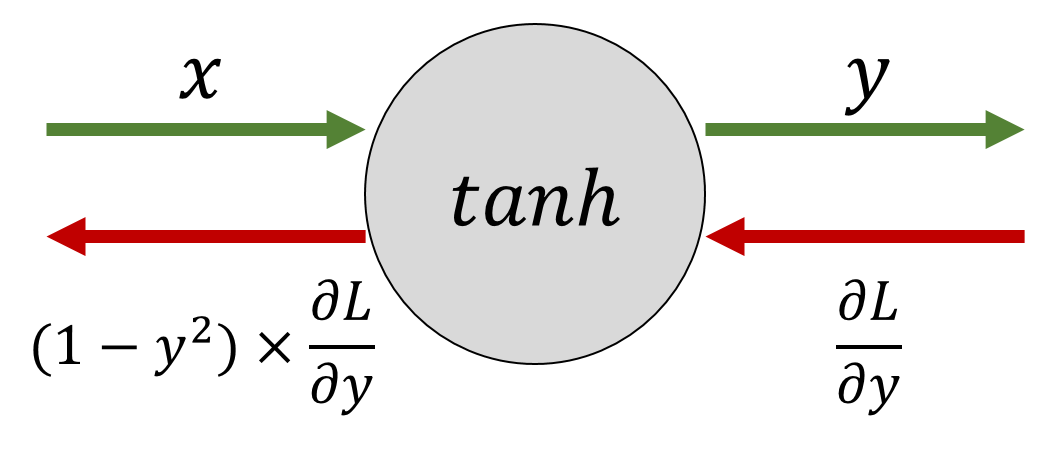
[](http://imgur.com/riURjqG)

## 하이퍼볼릭탄젠트 노드

하이퍼볼릭탄젠트 노드 y=tanh(x)y=tanh(x)의 로컬 그래디언트는 다음과 같습니다.

#### ∂y∂x=1−y2∂y∂x=1−y2

계산그래프는 아래와 같습니다.

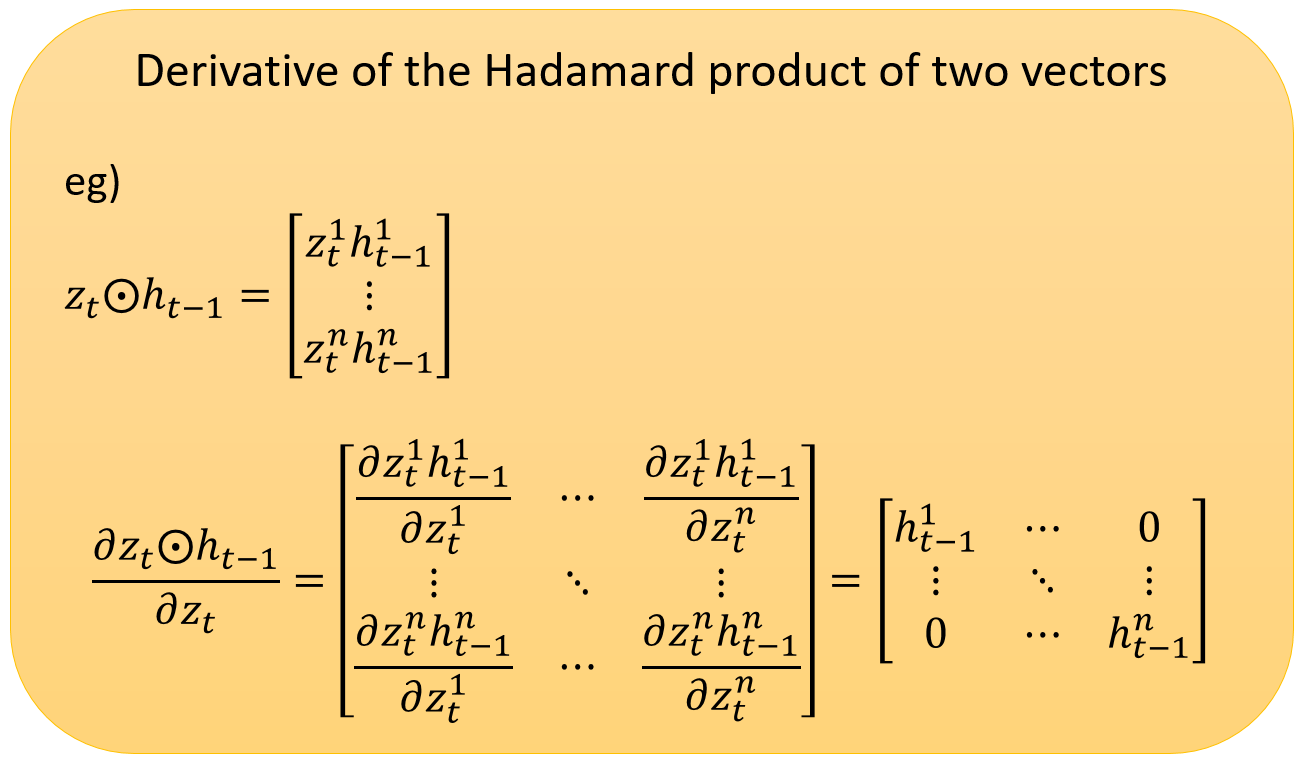
[](http://imgur.com/571sOas)

## Hadamard product 노드

**Hadamard product**란 요소별 곱셈을 뜻합니다. 기호로는 ⊙⊙ 등을 씁니다. 예컨대 아래와 같습니다.

#### ⎡⎣⎢132⎤⎦⎥⊙⎡⎣⎢453⎤⎦⎥=⎡⎣⎢4156⎤⎦⎥[132]⊙[453]=[4156]

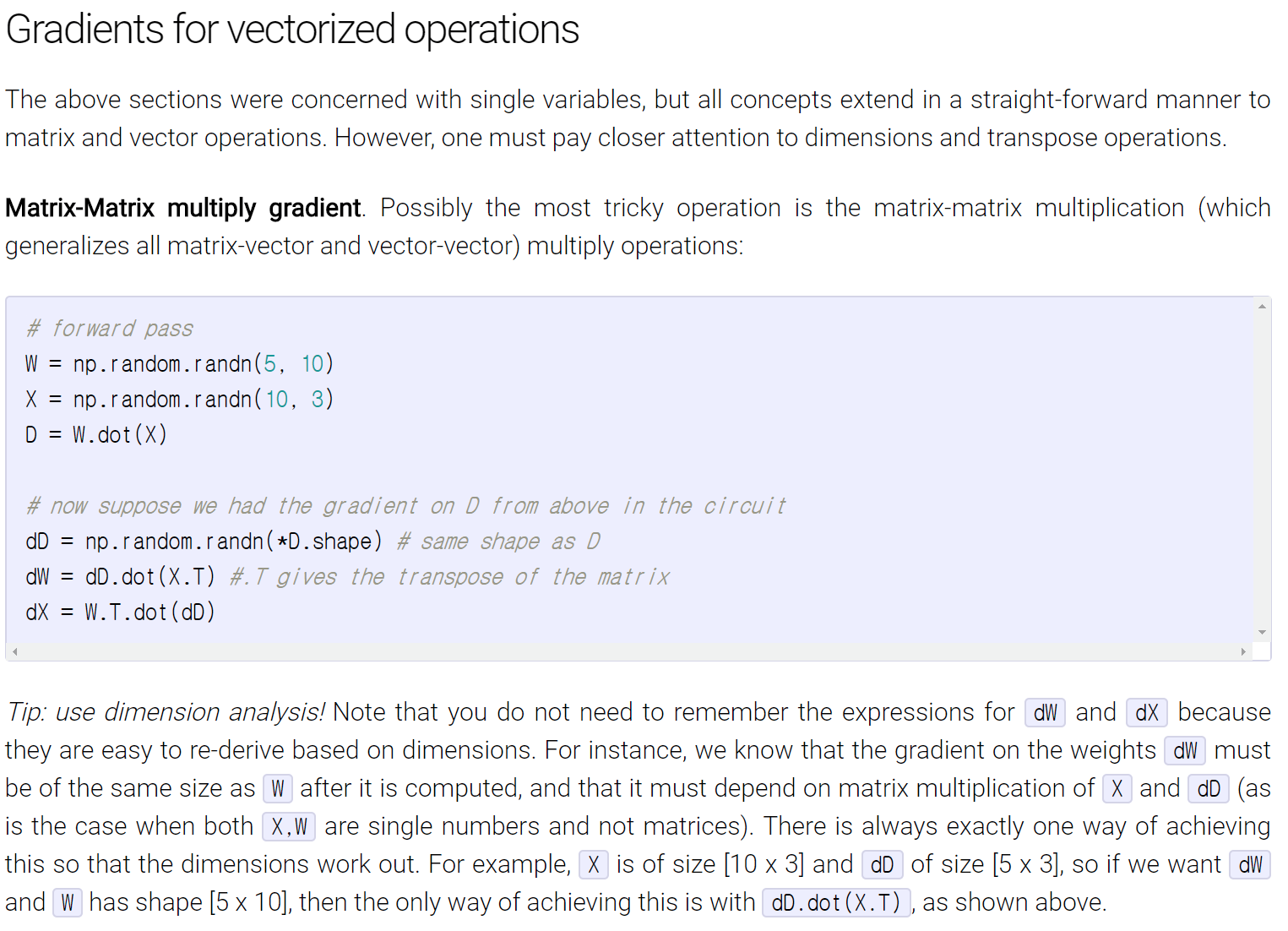
두 벡터에 Hadamard product 연산을 적용했을 때, 그 로컬 그래디언트는 아래와 같습니다.

[](http://imgur.com/u5VqWFj)

Hadamard product 노드 또한 다른 노드와 마찬가지로 위 로컬 그래디언트에 흘러들어온 그래디언트를 내적(inner product)해서 현시점의 그래디언트를 계산합니다. 그런데 흘러들어온 그래디언트 또한 벡터일 경우 Hadamard product 노드 로컬 그래디언트의 대각성분(위 그림에서 h1t−1ht−11,…,hnt−1ht−1n)과 요소별 곱셈을 하여도 같은 결과가 나옵니다.

## 벡터, 행렬로의 확장

지금까지 말씀드린 역전파는 기본적으로 스칼라를 대상으로 한 편미분과 역전파였습니다. 하지만 여기에 적용된 원칙들은 벡터, 행렬에도 적용할 수 있습니다. 해당 변수에 대한 그래디언트는 해당 변수의 차원 수와 일치해야 한다는 원칙을 기억하고 있으면 됩니다. 이와 관련 cs231n의 한 단락을 정리 용도로 캡처해 놨습니다.

[](https://imgur.com/OzLAK1L)

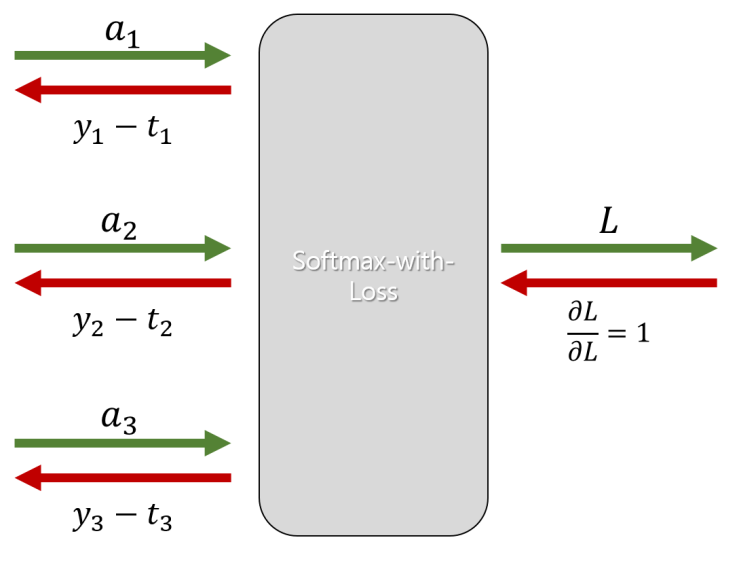
## Softmax-with-Loss 노드

뉴럴네트워크 말단에 보통 **Softmax-with-Loss** 노드를 둡니다. Softmax-with-Loss란 소프트맥스 함수와 **교차 엔트로피(Cross-Entropy)** 오차를 조합한 노드를 뜻합니다. 소프트맥스 함수와 교차 엔트로피의 수식은 아래와 같습니다.

akak=노드의 입력값, LL=노드의 출력값(Loss) tktk=정답 레이블(0 혹은 1), nn=정답 범주 개수

#### yk=exp(ak)∑ni=1exp(ai)L=−∑ktklogykyk=exp(ak)∑i=1nexp(ai)L=−∑ktklog⁡yk

Softmax-with-Loss 노드의 계산그래프를 매우 단순하게 그리면 아래와 같습니다.

[](http://imgur.com/gyeTKAn)

위 그림을 설명하자면 이렇습니다. Softmax-with-Loss 노드는 aa를 입력으로 받아서 Loss LL을 출력합니다. 역전파하는 그래디언트는 yk−tkyk−tk가 됩니다. 예컨대 정답이 t3t3이라면 역전파되는 그래디언트는 각각 y1,y2,y3−1y1,y2,y3−1이 됩니다.

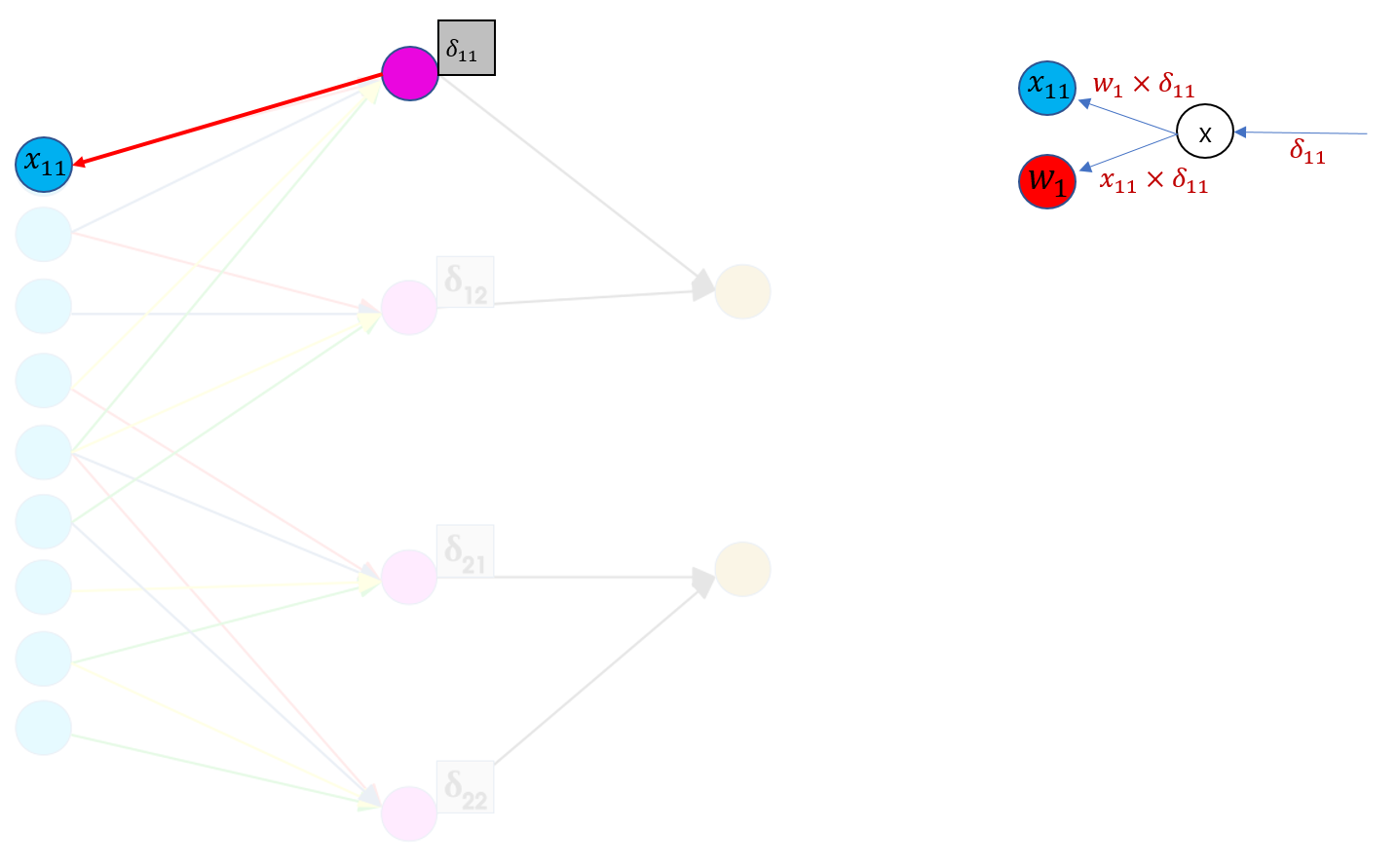
요컨대 Softmax-with-Loss 노드의 역전파 그래디언트를 구하려면 입력값에 소프트맥스 확률값을 취한 뒤, 정답 레이블에 해당하는 요소만 1을 빼주면 된다는 얘기입니다. 이를 파이썬 코드로 구현하면 아래와 같습니다.

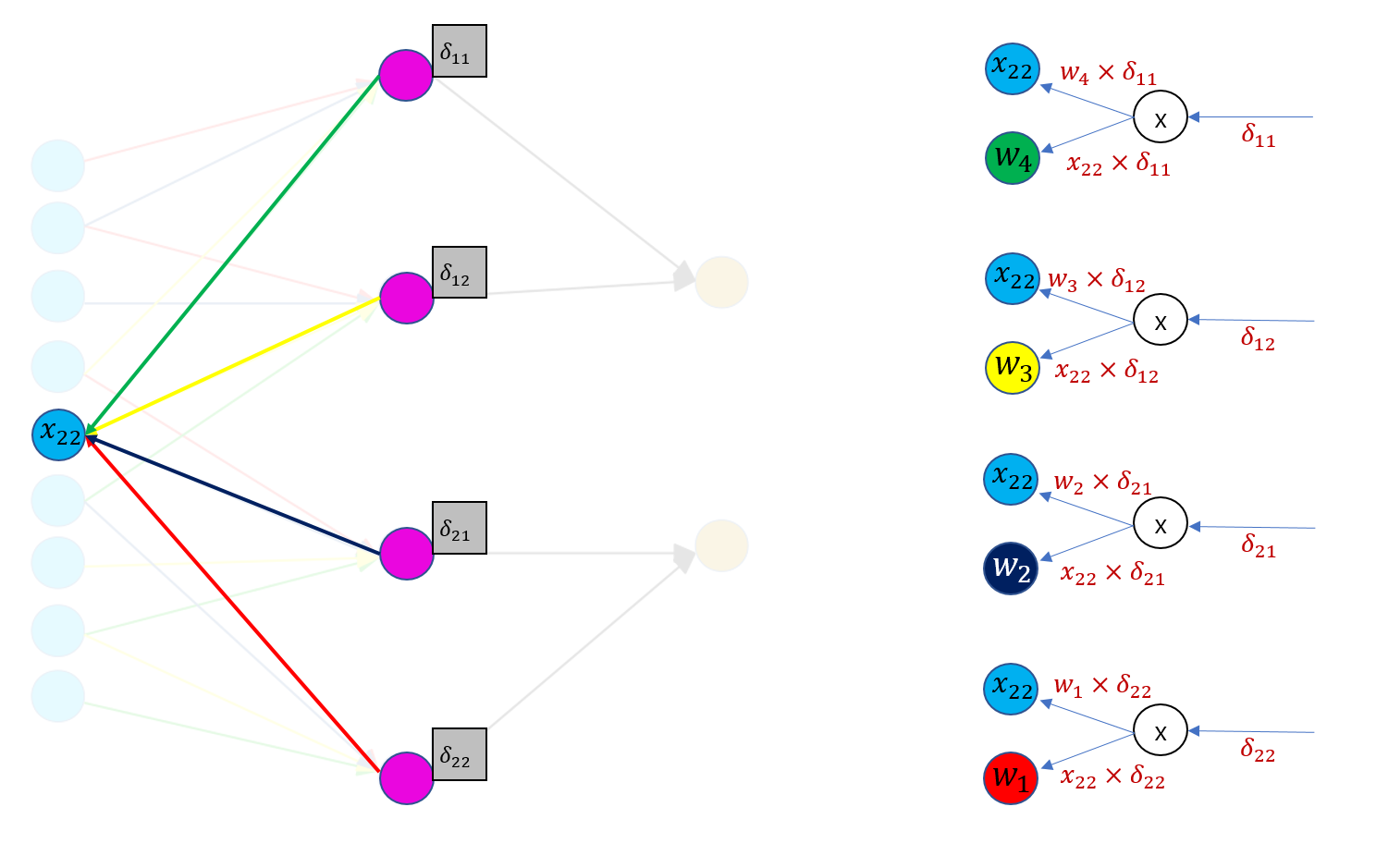
import numpy as np

p = np.exp(a) / np.sum(np.exp(a)) # softmax 확률 계산

da = np.copy(p)

da[target] -= 1 # target=정답 인덱스를 갖고 있는 변수



[](http://imgur.com/lxTuzam)