* 원소별 계산

DeZero에서 구현한 함수들의 입출력 모두 스칼라라고 가정.

x가 텐서일 경우 sin함수가 원소별로 적용.

```
x = Variable(np.array([[1, 2, 3], [4, 5, 6]]))
y = F.sin(x)
print(y)
```

```
variable([[ 0.84147098  0.90929743  0.14112001]
          [-0.7568025  -0.95892427 -0.2794155 ]])
```

입력과 출력 텐서의 형상은 바뀌지 않음.

*텐서 사용 시의 역전파

DeZero 함수에 텐서를 건네면 텐서의 원소마다 스칼라로 계산하고 역전파는 텐서의 원소별 계산에서도 성립함.

sum함수를 사용하면 주어진 텐서의 모든 원소의 총합을 구해 하나의 스칼라로 출력.

```
x = Variable(np.array([[1, 2, 3], [4, 5, 6]]))
c = Variable(np.array([[10, 20, 30], [40, 50 ,60]]))
t = x + c
y = F.sum(t)
print(y)
```

```
variable(231)
```

y.backward(retain_grad = True)를 실행하면 각 변수의 미분값이 구해짐.

기울기의 형상과 순전파 때의 데이터의 형상이 일치함.

```
y.backward(retain_grad=True)
print(y.grad)
print(t.grad)
print(x.grad)
print(c.grad)
```

```
variable(1)
variable([[1 1 1]
          [1 1 1]])
variable([[1 1 1]
          [1 1 1]])
variable([[1 1 1]
          [1 1 1]])
```

x.shape == x.grad.shape

c.shape == c.grad.shape

* 형상 변환 함수

원소별로 계산하지 않는 함수

- reshape
- transpose

두 함수 모두 텐서의 형상을 변환하는 함수.

* reshape함수 구현

```
x = np.array([[1, 2, 3], [4, 5, 6]])
y = np.reshape(x, (6, ))
print(y)
```

```
[1 2 3 4 5 6]
```

x의 형상을 (2, 3)에서 (6, )로 변환.

reshape함수는 형상만 변환하므로 구체적인 계산은 하지 않음.

역전파는 출력 쪽에서부터 기울기를 전달하기 때문에 기울기를 x.data.shape와 x.grad.shape가 일치하도록 변환 -> (6, )인 형상을 (2, 3)으로 변환.

* 행렬의 전치

넘파이의 transpose 함수를 사용하여 전치를 할 수 있음.

```
x = np.array([[1, 2, 3], [4, 5, 6]])
y = np.transpose(x)
print(x)
print(y)
```

```
[[1 2 3]
 [4 5 6]]
[[1 4]
 [2 5]
 [3 6]]
```

x의 형상이 (2, 3)에서 (3, 2)로 변환됨.

순전파는 np.transpose 함수로 전치 수 역전파는 출력 쪽에서 전해지는 기울기를 transpose 함수를 사용하여 반환.

```
x = Variable(np.array([[1, 2, 3], [4, 5, 6]]))
y = F.transpose(x)
y.backward()
print(x.grad)
```

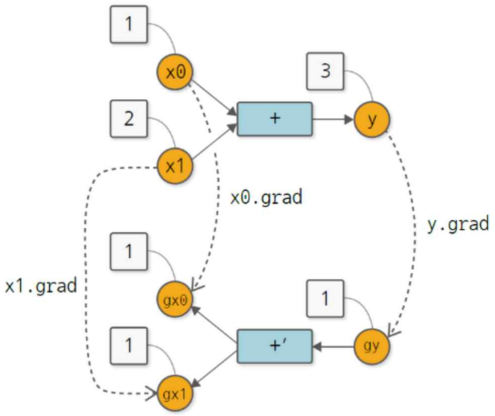
```
variable([[1 1 1]
          [1 1 1]])
```

*덧셈의 미분

$y = x_0 + x_1$일 때 $\frac{\partial y}{\partial x_0} = 1$, $\frac{\partial y}{\partial x_1} = 1$임.

역전파는 출력 쪽에서 전해지는 기울기를 그대로 입력 쪽으로 흘려보냄.

x0과 x1에는 출력 쪽에서 전해준 1이라는 기울기를 두 개로 복사하여 전달.




벡터에서 sum 함수를 적용한 역전파는 출력 쪽에서 전해준 값인 1을 [1, 1]벡터로 확장해 전파함.

원소가 2개 이상인 벡터의 경우 기울기를 입력 변수의 형상과 같아지도록 복사함.

DeZero에서 sum 함수 구현.

```
x = Variable(np.array([1, 2, 3, 4, 5, 6]))
y = F.sum(x)
y.backward()
print(y)
print(x.grad)
```

```
variable(21)
variable([1 1 1 1 1 1])
```

```
x = Variable(np.array([[1, 2, 3], [4, 5, 6]]))
y = F.sum(x)
y.backward()
print(y)
print(x.grad)
```

```
variable(21)
variable([[1 1 1]
          [1 1 1]])
```

* axis와 keepdims

axis는 축을 지정해주는 인수.

```
x = np.array([[1, 2, 3], [4, 5, 6]])
y = np.sum(x, axis = 0)
print(y)
print(x.shape, '->', y.shape)
```

```
[5 7 9]
(2, 3) -> (3,)
```

keepdims는 입력과 출력의 차원 수(축 수)를 똑같게 유지할지 정하는 플래그.

```
x = np.array([[1, 2, 3], [4, 5, 6]])
y = np.sum(x, keepdims = True)
print(y)
print(y.shape)
```

```
[[21]]
(1, 1)
```

DeZero의 sum 함수에서 axis와 keepdims 인수 지원

역전파시 broadcast_to 함수를 사용하여 입력변수의 형상이 같아지도록 기울기 원소 복사

* 브로드캐스트 함수

서로다른 형상을 가진 배열들간에 산술 연산을 수행하기 위해 벼열의 형상을 조정.

브로드캐스트를 사용하여 작은 배열을 큰 배역에 맞추어 연산을 수행할 수 있음.

* DeZero에서도 넘파이와 같은 브로드캐스트 지원

넘파이의 브로드캐스트 함수

```
x = np.array([1, 2, 3])
y = np.broadcast_to(x, (2, 3))
print(y)
```

```
[[1 2 3]
 [1 2 3]]
```

np.broadcast_to함수의 역전파

입력 x의 형상과 같아지도록 기울기의 합을 구하기 위해 sum_to(x, shape) 함수가 필요.

sum_to(x, shape) 함수는 x의 원소의 합을 구해 shape 형상으로 만들어주는 함수

sum_to 함수의 역전파는 broadcast_to 함수를 그대로 사용.

*DeZero에서의 브로드캐스트

```
x0 = Variable(np.array([1, 2, 3]))
x1 = Variable(np.array([10]))
y = x0 + x1
print(y)
```

```
variable([11 12 13])
```

브로드캐스트 역전파 계산

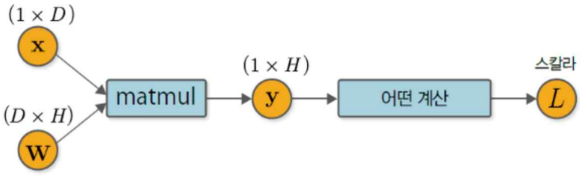
```
x0 = Variable(np.array([1, 2, 3]))
x1 = Variable(np.array([10]))
y = x0 + x1
print(y)
y.backward()
print(x1.grad)
```

```
variable([11 12 13])
variable([3])
```

* 행렬의 곱

벡터의 내적과 행렬의 곱 계산은 모두 np.dot 함수로 처리할 수 있음.

*행렬 곱의 역전파



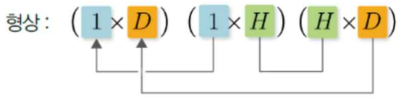

최종적으로 스칼라를 출력하는 계산을 다룸

위 순전파에서 L(손실함수)의 각 변수에 대한 미분을 역전파로 구함.

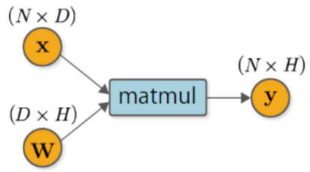
y의 각 원소의 변화를 통해 궁극적으로 L이 변화함.

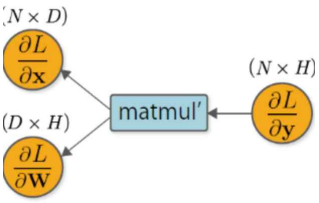
행렬곱의 형상

$$\frac{\partial L}{\partial \mathbf{x}} = \frac{\partial L}{\partial \mathbf{y}} \mathbf{W}^{\mathrm{T}}$$




역전파 수식 도출




y = xW라는 행렬 곱 계산

(x의 형상이 N * D라고 가정)

(x = N * D, W = D * H, y = N * H임)

*행렬곱 코드 구현

순전파는 np.dot(x, W) 대신 x.dot(W)로 구현

전치(W.T와 x.T) 시에는 DeZero의 transpose 함수가 호출됨

x.grad.shape와 x.shape가 동일하고, w.grad.shape와 W.shape가 동일함을 확인할 수 있음.