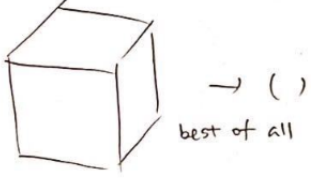
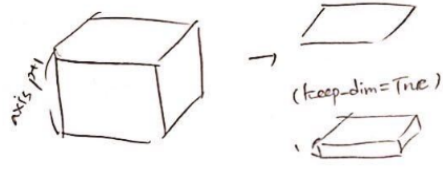


# Statistics

torch.max, torch.min

| torch.max(input) or .min  |  |
|---|--|
|  | Require  |
|   |  |
|   | Guarantees   |
|   | <ul style="list-style-type: none"> <li>• <math> y  = ()</math></li> </ul>                    |
|   | Comment  |
|   | <ul style="list-style-type: none"> <li>• 주어진 텐서에서 최대/최소 값을 ()-shape 텐서로 반환(스칼라 X)</li> </ul> |

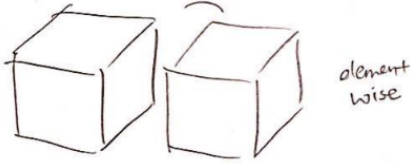
$$\forall \text{ft} \in \{\min, \max\}, \quad \frac{\sigma \vdash E \Rightarrow \cdot, c}{\sigma \vdash \text{ft}(E) \Rightarrow (), c}$$

| torch.max(input, dim, keep_dim=False, out=None) or .min                            |   |
|--|---|
|  | Require   |
|  | <ul style="list-style-type: none"> <li>• <math> input  = (d_1, d_2, \dots, d_k)</math></li> <li>• <math>k \geq 1, 0 \leq \text{dim} &lt; k</math></li> </ul>  |
|  | Guarantees  |
|  | <ul style="list-style-type: none"> <li>• <math> y  =  z  = (d_1, d_2, \dots, d_{\text{dim}}, d_{\text{dim}+2}, \dots, d_k)</math>인 <math>(y, z)</math> 튜플 반환</li> <li>• 세 번째 인자 <code>keep_dim</code>이 <code>True</code>이면 <math> y  =  z  = (d_1, d_2, \dots, d_{\text{dim}}, 1, d_{\text{dim}+2}, \dots, d_k)</math>로 처리</li> </ul> |
|  | Comment   |
|  | <ul style="list-style-type: none"> <li>• 주어진 텐서에서 축 상의 최대/최소값과, 그에 해당하는 인덱스 번호를 쌍으로 묶어 튜플로 반환</li> <li>• <code>out</code>-텐서 인자가 있는 함수</li> </ul>   |

$$\forall \text{ft} \in \{\min, \max\}, \quad \frac{\begin{array}{l} \sigma \vdash E \Rightarrow e, c \\ k = \text{rank}(e) \\ e' = e[1:n] @ e[n+2:k] \\ c' = \{(k \geq 1) \wedge (0 \leq n < k)\} \end{array}}{\sigma \vdash \text{ft}(E, n) \Rightarrow (e', e'), c \cup c'} \quad \text{tuple 형태로 반환}$$

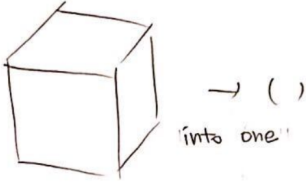
$$\forall \text{ft} \in \{\min, \max\}, \quad \frac{\begin{array}{l} \sigma \vdash E \Rightarrow e, c \\ k = \text{rank}(e) \\ e' = e[1:n] @ (1) @ e[n+2:k] \\ c' = \{(k \geq 1) \wedge (0 \leq n < k)\} \end{array}}{\sigma \vdash \text{ft}(E, n, \text{True}) \Rightarrow (e', e'), c \cup c'} \quad \text{tuple 형태로 반환}$$

$$\forall \text{ft} \in \{\min, \max\}, \quad \frac{\sigma \vdash \text{ft}(E, n) \Rightarrow (e, e), c}{\sigma \vdash \text{ft}(E, n, \text{False}) \Rightarrow (e, e), c} \quad \text{tuple 형태로 반환}$$

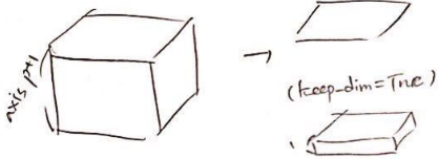
| torch.max(input, other, out=None) or .min   |  |
|---|--|
|  | Require  |
|   | <ul style="list-style-type: none"> <li>• <math>broadcastable( input ,  other )</math></li> </ul>               |
|   | Guarantees   |
|   | <ul style="list-style-type: none"> <li>• <math>broadcast( input ,  other )</math></li> </ul>                   |
|   | Comment  |
|   | <ul style="list-style-type: none"> <li>• 두 텐서의 elementwise 최대/최소</li> <li>• <i>out</i>-텐서 인자가 있는 함수</li> </ul> |

$$\forall ft \in \{\min, \max\}, \quad \frac{\sigma \vdash E_1 \Rightarrow e_1, c_1 \quad \sigma \vdash E_2 \Rightarrow e_2, c_2}{\sigma \vdash ft(E_1, E_2) \Rightarrow broadcast(e_1, e_2), c_1 \cup c_2 \cup broadcastable(e_1, e_2)}$$

torch.sum, torch.mean

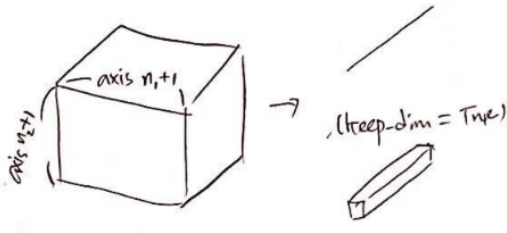
| torch.sum(input, dtype=None) or .mean   |  |
|---|--|
|  | Require  |
|   | <ul style="list-style-type: none"> <li>• .mean에 대해서는 텐서 타입이 floating이어야 함</li> </ul>       |
|   | Guarantees   |
|   | <ul style="list-style-type: none"> <li>• <math> y  = ()</math></li> </ul>                  |
|   | Comment  |
|   | <ul style="list-style-type: none"> <li>• 주어진 텐서의 합계/평균값을 ()-shape 텐서로 반환(스칼라 X)</li> </ul> |

$$\forall ft \in \{\text{sum}, \text{mean}\}, \quad \frac{\sigma \vdash E \Rightarrow \_, c}{\sigma \vdash ft(E) \Rightarrow (), c}$$

| torch.sum(input, dim, keep_dim=False, dtype=None) or .mean                          |  |
|---|--|
|  | Require  |
|   | <ul style="list-style-type: none"> <li>• <math> input  = (d_1, d_2, \dots, d_k)</math></li> <li>• <math>k \geq 1, 0 \leq dim &lt; k</math></li> <li>• .mean에 대해서는 텐서 타입이 floating이어야 함</li> </ul>  |
|   | Guarantees   |
|   | <ul style="list-style-type: none"> <li>• <math> y  = (d_1, d_2, \dots, d_{dim}, d_{dim+2}, \dots, d_k)</math></li> <li>• 세 번째 인자 <i>keep_dim</i>이 <i>True</i>이면 <math> y  = (d_1, d_2, \dots, d_{dim}, 1, d_{dim+2}, \dots, d_k)</math></li> </ul> |
|   | Comment  |
|   | <ul style="list-style-type: none"> <li>• 주어진 텐서에서 축 상의 합/평균을 반환</li> </ul>   |

$$\forall ft \in \{\text{sum}, \text{mean}\}, \quad \frac{\sigma \vdash E \Rightarrow e, c \quad k = \text{rank}(e) \quad e' = e[1:n] @ e[n+2:k] \quad c' = \{(k \geq 1) \wedge (0 \leq n < k)\}}{\sigma \vdash ft(E, n) \Rightarrow e', c \cup c'}$$

$$\begin{array}{l}
\sigma \vdash E \Rightarrow e, c \\
k = \text{rank}(e) \\
e' = e[1:n]@ (1) @ e[n+2:k] \\
c' = \{(k \geq 1) \wedge (0 \leq n < k)\} \\
\forall \text{ft} \in \{\text{sum}, \text{mean}\}, \quad \frac{}{\sigma \vdash \text{ft}(E, n, \text{True}) \Rightarrow e', c \cup c'}
\end{array}$$

| torch.sum(input, [n1, n2, ..., nl], keep_dim=False, dtype=None) or .mean          |   |
|---|---|
|  | Require   |
|   | <ul style="list-style-type: none"> <li>• <math> x  = (d_1, d_2, \dots, d_k)</math></li> <li>• <math>k \geq 1, 0 \leq n_1, n_2, \dots, n_l &lt; k</math></li> <li>• .mean에 대해서는 텐서 타입이 floating이어야 함</li> </ul>  |
|   | Guarantees  |
|   | <ul style="list-style-type: none"> <li>• <math> y  = (d_{i_1}, d_{i_2}, \dots, d_{i_{k-l}})</math><br/> - 1, 2, ..., k에서 <math>n_1+1, n_2+1, \dots, n_l+1</math>번째 항이 지워진 shape</li> <li>• 세 번째 인자 keep_dim이 True이면 <math>n_1+1, n_2+1, \dots, n_l+1</math>번째 항은 삭제되지 않고 1로 남음</li> </ul> |
|   | Comment   |
|   | <ul style="list-style-type: none"> <li>• 주어진 텐서에서 여러 축을 통합한 합/평균을 반환</li> <li>• <math>[n_1, n_2, \dots, n_l]</math>에 중복된 원소가 들어가도 상관없이 잘 작동함</li> </ul>   |

$$\begin{array}{l}
\sigma \vdash E \Rightarrow e, c \\
k = \text{rank}(e) \\
e_1 = \text{if } 0 \in \{n_1, n_2, \dots, n_r\} \text{ then } () \text{ else } (e[1]) \\
e_2 = \text{if } 1 \in \{n_1, n_2, \dots, n_r\} \text{ then } () \text{ else } (e[2]) \\
\vdots \\
e_k = \text{if } k-1 \in \{n_1, n_2, \dots, n_r\} \text{ then } () \text{ else } (e[k]) \\
e' = e_1 @ e_2 @ \dots @ e_k \\
c' = \{(k \geq 1) \wedge (\forall i = 1, 2, \dots, r, 0 \leq n_i < k)\} \\
\forall \text{ft} \in \{\text{sum}, \text{mean}\}, \quad \frac{}{\sigma \vdash \text{ft}(E, (n_1, n_2, \dots, n_r)) \Rightarrow e', c \cup c'}
\end{array}$$

$$\begin{array}{l}
\sigma \vdash E \Rightarrow e, c \\
k = \text{rank}(e) \\
e_1 = \text{if } 0 \in \{n_1, n_2, \dots, n_r\} \text{ then } (1) \text{ else } (e[1]) \\
e_2 = \text{if } 1 \in \{n_1, n_2, \dots, n_r\} \text{ then } (1) \text{ else } (e[2]) \\
\vdots \\
e_k = \text{if } k-1 \in \{n_1, n_2, \dots, n_r\} \text{ then } (1) \text{ else } (e[k]) \\
e' = e_1 @ e_2 @ \dots @ e_k \\
c' = \{(k \geq 1) \wedge (\forall i = 1, 2, \dots, r, 0 \leq n_i < k)\} \\
\forall \text{ft} \in \{\text{sum}, \text{mean}\}, \quad \frac{}{\sigma \vdash \text{ft}(E, (n_1, n_2, \dots, n_r), \text{True}) \Rightarrow e', c \cup c'}
\end{array}$$

## torch.prod

| torch.prod(input, dtype=None),<br>torch.prod(input, dim, keepdim=False, dtype=None) |  |
|---|--|
|   | Comment  |
|   | <ul style="list-style-type: none"> <li>대부분 <b>sum</b>과 똑같음</li> <li>하지만, <i>dim</i> 부분에 튜플값이 들어갈 수 없음. 즉 여러 축에 대한 곱을 계산할 수 없음</li> </ul> |

$$\frac{\sigma \vdash E \Rightarrow \_, c}{\sigma \vdash \text{prod}(E) \Rightarrow (), c}$$

$$\frac{\begin{array}{l} \sigma \vdash E \Rightarrow e, c \\ k = \text{rank}(e) \\ e' = e[1:n] @ e[n+2:k] \\ c' = \{(k \geq 1) \wedge (0 \leq n < k)\} \end{array}}{\sigma \vdash \text{prod}(E, n) \Rightarrow e', c \cup c'}$$

$$\frac{\begin{array}{l} \sigma \vdash E \Rightarrow e, c \\ k = \text{rank}(e) \\ e' = e[1:n] @ (1) @ e[n+2:k] \\ c' = \{(k \geq 1) \wedge (0 \leq n < k)\} \end{array}}{\sigma \vdash \text{prod}(E, n, \text{True}) \Rightarrow e', c \cup c'}$$

## torch.norm

| torch.norm(input, p='fro', dim=None, keepdim=False, out=None, dtype=None) |  |
|---|--|
|   | Require  |
|   | <ul style="list-style-type: none"> <li><math> input  = (d_1, d_2, \dots, d_k)</math></li> <li><math>dim = None</math> or <math>0 \leq dim &lt; k</math> or <math>dim = (p_1, p_2, \dots, p_m)</math> such that <math>0 \leq p_1, p_2, \dots, p_m &lt; k</math> (tuple)</li> </ul>  |
|   | Guarantees   |
|   | <ul style="list-style-type: none"> <li>If <math>dim = None</math> then <ul style="list-style-type: none"> <li><math> y  = ()</math></li> </ul> </li> <li>If <math>dim \neq None</math> then <ul style="list-style-type: none"> <li>If <math>keepdim = False</math> then <ul style="list-style-type: none"> <li><math> y  = (d_1, d_2, \dots, d_{dim}, d_{dim+2}, \dots, d_k)</math></li> </ul> </li> <li>If <math>keepdim = True</math> then <ul style="list-style-type: none"> <li><math> y  = (d_1, d_2, \dots, d_{dim}, 1, d_{dim+2}, \dots, d_k)</math></li> </ul> </li> </ul> </li> <li>만약 <i>dim</i>이 여러 axis로 구성된 튜플이면 해당 축들이 지워지거나, 아니면 1차원으로 바뀐 텐서 shape을 반환</li> </ul> |
|   | Comment  |
|   | <ul style="list-style-type: none"> <li>일단 기본적인 shape 연산은 <b>sum</b>과 비슷함</li> <li><i>out</i>-텐서 인자가 있는 함수</li> </ul>   |

$$\frac{\sigma \vdash E \Rightarrow \_, c}{\sigma \vdash \text{norm}(E) \Rightarrow (), c}$$

$$\begin{array}{l}
\sigma \vdash E \Rightarrow e, c \\
k = \mathbf{rank}(e) \\
e' = e[1:n]@e[n+2:k] \\
c' = \{(k \geq 1) \wedge (0 \leq n < k)\} \\
\hline
\sigma \vdash \mathbf{norm}(E, p = 'fro', n, False) \Rightarrow e', c \cup c'
\end{array}$$

$$\begin{array}{l}
\sigma \vdash E \Rightarrow e, c \\
k = \mathbf{rank}(e) \\
e' = e[1:n]@(1)@e[n+2:k] \\
c' = \{(k \geq 1) \wedge (0 \leq n < k)\} \\
\hline
\sigma \vdash \mathbf{norm}(E, p = 'fro', n, True, ...) \Rightarrow e', c \cup c'
\end{array}$$

$$\begin{array}{l}
\sigma \vdash E \Rightarrow e, c \\
k = \mathbf{rank}(e) \\
e_1 = \mathbf{if} \ 0 \in \{n_1, n_2, \dots, n_r\} \ \mathbf{then} \ () \ \mathbf{else} \ (e[1]) \\
e_2 = \mathbf{if} \ 1 \in \{n_1, n_2, \dots, n_r\} \ \mathbf{then} \ () \ \mathbf{else} \ (e[2]) \\
\dots \\
e_k = \mathbf{if} \ k-1 \in \{n_1, n_2, \dots, n_r\} \ \mathbf{then} \ () \ \mathbf{else} \ (e[k]) \\
e' = e_1@e_2@\dots@e_k \\
c' = \{(k \geq 1) \wedge (\forall i = 1, 2, \dots, r, 0 \leq n_i < k)\} \\
\hline
\sigma \vdash \mathbf{norm}(E, p = 'fro', (n_1, n_2, \dots, n_r), False, ...) \Rightarrow e', c \cup c'
\end{array}$$

$$\begin{array}{l}
\sigma \vdash E \Rightarrow e, c \\
k = \mathbf{rank}(e) \\
e_1 = \mathbf{if} \ 0 \in \{n_1, n_2, \dots, n_r\} \ \mathbf{then} \ (1) \ \mathbf{else} \ (e[1]) \\
e_2 = \mathbf{if} \ 1 \in \{n_1, n_2, \dots, n_r\} \ \mathbf{then} \ (1) \ \mathbf{else} \ (e[2]) \\
\dots \\
e_k = \mathbf{if} \ k-1 \in \{n_1, n_2, \dots, n_r\} \ \mathbf{then} \ (1) \ \mathbf{else} \ (e[k]) \\
e' = e_1@e_2@\dots@e_k \\
c' = \{(k \geq 1) \wedge (\forall i = 1, 2, \dots, r, 0 \leq n_i < k)\} \\
\hline
\forall \mathbf{ft} \in \{\mathbf{sum}, \mathbf{mean}\}, \quad \sigma \vdash \mathbf{norm}(E, p = 'fro', (n_1, n_2, \dots, n_r), True, ...) \Rightarrow e', c \cup c'
\end{array}$$

`torch.Tensor.all`, `torch.Tensor.any`

| <code>a.all()</code> , <code>a.all(dim, keepdim=False, out=None)</code> or <code>.any</code> |   |
|--|---|
|  | Comment   |
|  | <ul style="list-style-type: none"> <li>• <code>shape</code>의 기능은 <code>prod</code>과 똑같음</li> <li>• <code>torch.Tensor</code> 하위 함수이며, <code>bool</code>-타입 텐서에서만 사용 가능</li> </ul> |

$$\forall \mathbf{ft} \in \{\mathbf{all}, \mathbf{any}\}, \quad \frac{\sigma \vdash E \Rightarrow \neg, c}{\sigma \vdash E.\mathbf{ft}() \Rightarrow (), c}$$

$$\begin{array}{l}
\sigma \vdash E \Rightarrow e, c \\
k = \mathbf{rank}(e) \\
e' = e[1:n]@e[n+2:k] \\
c' = \{(k \geq 1) \wedge (0 \leq n < k)\} \\
\hline
\forall \mathbf{ft} \in \{\mathbf{all}, \mathbf{any}\}, \quad \frac{\sigma \vdash E.\mathbf{ft}(n) \Rightarrow e', c \cup c'}{\sigma \vdash E.\mathbf{ft}(n) \Rightarrow e', c \cup c'}
\end{array}$$

$$\begin{array}{c}
\sigma \vdash E \Rightarrow e, c \\
k = \mathbf{rank}(e) \\
e' = e[1:n]@ (1)@ e[n+2:k] \\
c' = \{(k \geq 1) \wedge (0 \leq n < k)\} \\
\forall \mathbf{ft} \in \{\mathbf{all}, \mathbf{any}\}, \quad \frac{}{\sigma \vdash E.\mathbf{ft}(n, \mathbf{True}) \Rightarrow e', c \cup c'}
\end{array}$$

torch.var, torch.std

| torch.var(input, unbiased=True) or .std |   |
|---|---|
|   | Require   |
|   | <ul style="list-style-type: none"> <li>• 텐서 타입이 floating이어야 함</li> </ul>                    |
|   | Guarantees  |
|   | <ul style="list-style-type: none"> <li>• <math> y  = ()</math></li> </ul>                   |
|   | Comment   |
|   | <ul style="list-style-type: none"> <li>• 주어진 텐서의 분산/표준편차를 ()-shape 텐서로 반환(스칼라 X)</li> </ul> |

$$\forall \mathbf{ft} \in \{\mathbf{var}, \mathbf{std}\}, \quad \frac{\sigma \vdash E \Rightarrow \neg, c}{\sigma \vdash \mathbf{ft}(E) \Rightarrow (), c}$$

| torch.var(input, dim, keepdim=False, unbiased=None, out=None) or<br>torch.std(input, dim, unbiased=True, keepdim=False, out=None) |  |
|---|--|
|   | Require  |
|   | <ul style="list-style-type: none"> <li>• <math> input  = (d_1, d_2, \dots, d_k)</math></li> <li>• <math>k \geq 1, 0 \leq dim &lt; k</math></li> <li>• 텐서 타입이 floating이어야 함</li> </ul>  |
|   | Guarantees   |
|   | <ul style="list-style-type: none"> <li>• <math> y  = (d_1, d_2, \dots, d_{dim}, d_{dim+2}, \dots, d_k)</math></li> <li>• <math>keepdim</math>이 <math>\mathbf{True}</math>이면 <math> y  = (d_1, d_2, \dots, d_{dim}, 1, d_{dim+2}, \dots, d_k)</math></li> </ul> |
|   | Comment  |
|   | <ul style="list-style-type: none"> <li>• 주어진 텐서에서 축 상의 분산/표준편차를 반환</li> <li>• <b>sum</b>, <b>mean</b>과 마찬가지로 <math>dim</math> 부분은 튜플로 쓰일 수도 있음 (여러 축에 대한 분산/표준편차)</li> </ul>   |

$$\begin{array}{c}
\sigma \vdash E \Rightarrow e, c \\
k = \mathbf{rank}(e) \\
e' = e[1:n]@ e[n+2:k] \\
c' = \{(k \geq 1) \wedge (0 \leq n < k)\} \\
\forall \mathbf{ft} \in \{\mathbf{var}, \mathbf{std}\}, \quad \frac{}{\sigma \vdash \mathbf{ft}(E, n) \Rightarrow e', c \cup c'}
\end{array}$$

$$\begin{array}{c}
\sigma \vdash E \Rightarrow e, c \\
k = \mathbf{rank}(e) \\
e' = e[1:n]@ (1)@ e[n+2:k] \\
c' = \{(k \geq 1) \wedge (0 \leq n < k)\} \\
\forall \mathbf{ft} \in \{\mathbf{var}, \mathbf{std}\}, \quad \frac{}{\sigma \vdash \mathbf{ft}(E, n, \mathbf{True}) \Rightarrow e', c \cup c'}
\end{array}$$

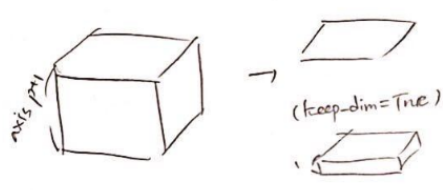
$keepdim$ 이  $\mathbf{True}$ 인 상황에 대한 proof tree

$$\begin{array}{l}
\sigma \vdash E \Rightarrow e, c \\
k = \text{rank}(e) \\
e_1 = \text{if } 0 \in \{n_1, n_2, \dots, n_r\} \text{ then } () \text{ else } (e[1]) \\
e_2 = \text{if } 1 \in \{n_1, n_2, \dots, n_r\} \text{ then } () \text{ else } (e[2]) \\
\vdots \\
e_k = \text{if } k-1 \in \{n_1, n_2, \dots, n_r\} \text{ then } () \text{ else } (e[k]) \\
e' = e_1 @ e_2 @ \dots @ e_k \\
c' = \{(k \geq 1) \wedge (\forall i = 1, 2, \dots, r, 0 \leq n_i < k)\} \\
\hline
\forall \text{ft} \in \{\text{var}, \text{std}\}, \quad \sigma \vdash \text{ft}(E, (n_1, n_2, \dots, n_r)) \Rightarrow e', c \cup c'
\end{array}$$

$$\begin{array}{l}
\sigma \vdash E \Rightarrow e, c \\
k = \text{rank}(e) \\
e_1 = \text{if } 0 \in \{n_1, n_2, \dots, n_r\} \text{ then } (1) \text{ else } (e[1]) \\
e_2 = \text{if } 1 \in \{n_1, n_2, \dots, n_r\} \text{ then } (1) \text{ else } (e[2]) \\
\vdots \\
e_k = \text{if } k-1 \in \{n_1, n_2, \dots, n_r\} \text{ then } (1) \text{ else } (e[k]) \\
e' = e_1 @ e_2 @ \dots @ e_k \\
c' = \{(k \geq 1) \wedge (\forall i = 1, 2, \dots, r, 0 \leq n_i < k)\} \\
\hline
\forall \text{ft} \in \{\text{var}, \text{std}\}, \quad \sigma \vdash \text{ft}(E, (n_1, n_2, \dots, n_r), \text{True}) \Rightarrow e', c \cup c'
\end{array}$$

*keepdim*이 *True*인 상황에 대한 proof tree

torch.mode, torch.median

| torch.mode(input, dim=-1, keep_dim=False, out=None) or torch.median  |   |
|--|---|
|   | Require   |
|  | <ul style="list-style-type: none"> <li>• <math> input  = (d_1, d_2, \dots, d_k)</math></li> <li>• <math>k \geq 1</math></li> <li>• <math>0 \leq \text{dim} &lt; k</math> (<math>\text{dim} \leftarrow k-1</math> if <math>\text{dim} = -1</math>)</li> </ul>  |
|  | Guarantees  |
|  | <ul style="list-style-type: none"> <li>• <math> y  =  z  = (d_1, d_2, \dots, d_{\text{dim}}, d_{\text{dim}+2}, \dots, d_k)</math>인 <math>(y, z)</math> 튜플 반환</li> <li>• 세 번째 인자 <i>keep_dim</i>이 <i>True</i>이면 <math> y  =  z  = (d_1, d_2, \dots, d_{\text{dim}}, 1, d_{\text{dim}+2}, \dots, d_k)</math></li> </ul> |
| Comment  |   |
| <ul style="list-style-type: none"> <li>• 입력받은 축을 기준으로한 통계적 최빈값/중간값 계산 함수</li> <li>• (결과값, 인덱스) 튜플 형태로 반환하기 때문에, <i>out</i> 인자도 들어가는 텐서도 2-원소 튜플 형태로 들어감</li> </ul> |   |

$$\begin{array}{l}
\sigma \vdash E \Rightarrow e, c \\
k = \text{rank}(e) \\
e' = e[1:n] @ e[n+2:k] \\
c' = \{(k \geq 1) \wedge (0 \leq n < k)\} \\
\hline
\forall \text{ft} \in \{\text{mode}, \text{median}\}, \quad \sigma \vdash \text{ft}(E, n = -1, \text{out} = \text{None}) \Rightarrow (e', e'), c \cup c'
\end{array}$$

tuple 형태로 반환

$$\begin{array}{c}
\sigma \vdash E \Rightarrow e, c \\
k = \mathbf{rank}(e) \\
e' = e[1:n]@ (1) @ e[n+2:k] \\
c' = \{(k \geq 1) \wedge (0 \leq n < k)\} \\
\forall \mathbf{ft} \in \{\mathbf{mode}, \mathbf{median}\}, \quad \frac{}{\sigma \vdash \mathbf{ft}(E, n = -1, \mathbf{True}, \mathbf{out} = \mathbf{None}) \Rightarrow (e', e'), c \cup c'}
\end{array}$$

tuple 형태로 반환

$$\forall \mathbf{ft} \in \{\mathbf{mode}, \mathbf{median}\}, \quad \frac{\sigma \vdash \mathbf{ft}(E, n) \Rightarrow (e, e), c}{\sigma \vdash \mathbf{ft}(E, n = -1, \mathbf{False}, \mathbf{out} = \mathbf{None}) \Rightarrow (e, e), c}$$

tuple 형태로 반환

`torch.sort`

| torch.sort(input, dim=-1, descending=False, out=None) |  |
|---|--|
|   | Require  |
|   | <ul style="list-style-type: none"> <li>• <math> input  = (d_1, d_2, \dots, d_k)</math></li> <li>• <math>dim = -1</math> or <math>-k \leq dim &lt; k</math></li> </ul>  |
|   | Guarantees   |
|   | <ul style="list-style-type: none"> <li>• <math> y  =  z  = (d_1, d_2, \dots, d_k) =  input </math>인 <math>(y, z)</math> 튜플 반환</li> </ul>   |
|   | Comment  |
|   | <ul style="list-style-type: none"> <li>• <math>dim</math> 축을 기준으로 원소를 정렬</li> <li>• <math>(y, z)</math>에서 <math>y</math>는 정렬된 텐서, <math>z</math>는 인덱스 텐서로 원본 <math>input</math> 텐서와 <math>shape</math>이 서로 같다.</li> <li>• <math>out</math>-텐서 옵션이 있는 함수</li> </ul> |

$$\begin{array}{c}
\sigma \vdash E \Rightarrow e, c \\
c' = \{(dim = -1 \vee -\mathbf{rank}(e) \leq dim < \mathbf{rank}(e))\} \\
\sigma \vdash \mathbf{sort}(E, dim = -1, \mathbf{descending} = \mathbf{False}, \mathbf{out} = \mathbf{None}) \Rightarrow (e, e), c \cup c'
\end{array}$$

tuple 형태로 반환