

# torch常用基础函数



zhaoQiang012 (/u/94e624cec680) [+ 关注](#)

2018.07.23 14:27\* 字数 6068 阅读 1317 评论 0 喜欢 0

(/u/94e624cec680)

## torch

本笔记引用自PyTorch中文文档

包 torch 包含了多维张量的数据结构及基于其上的多种数学操作。

### 1. 张量Tensors

`torch.is_tensor(obj) :`

如果 obj 是一个 pytorch 张量，则返回 True

`torch.is_storage(obj) :`

如果 obj 是一个 pytorch storage 对象，则返回 True

`torch.numel(input) :`

返回 input 张量中的元素个数。

### 2. 创建操作

`torch.eye(n, m=None, out=None) :`

返回一个2维张量，对角线为1,其它位置为0

- n (int) -行数
- m (int, optional)列数，如果为None,则默认为n
- out (Tensor, optional)

`torch.from_numpy(ndarray) :`

将 `numpy.ndarray` 转换为 `Tensor`，返回的张量tensor和numpy的ndarray共享同一内存空间，修改一个会导致另一个也被修改，返回的张量不能改变大小

`torch.linspace(start, end, steps=100, out=None) :`

返回一个1维张量，包含在 start 和 end 上均匀间隔的 steps 个点

- start (float) -序列起点
- end (float) - 序列终点
- steps (int) - 在 start 与 end 间生成的样本数
- out (Tensor, optional) - 结果张量



```
torch.logspace(start, end, steps=100, out=None) :
```

返回一个1维张量，包含在区间 $10^{\text{start}}$ 和 $10^{\text{end}}$ 上以对数刻度均匀间隔的 `steps` 个点。

```
torch.ones(*sizes, out=None) :
```

返回一个全为1的张量，形状由可变参数 `sizes` 定义

- `sizes (int...)` - 整数序列，定义了输出形状

```
torch.rand(*sizes, out=None) :
```

返回一个张量，包含了从区间(0, 1)的均匀分布中抽取的一组随机数，形状由可变参数 `sizes` 定义。

```
torch.randn(*sizes, out=None) :
```

返回一个张量，包含了从标准正态分布(mean=0, std=1)中抽取一组随机数，形状由可变参数 `sizes` 定义。

```
torch.randperm(n, out=None) :
```

给定参数 `n`，返回一个从0到`n-1`的随机整数排列

- `n (int)` - 上边界(不包含)

```
torch.arange(start, end, step=1, out=None) :
```

返回一个1维张量，长度为  $\text{floor}((\text{end}-\text{start})/\text{step})$ ，以 `step` 为步长的一组序列值。

- `start (float)` - 起点
- `end (float)` - 终点(不包含)
- `step (float)` - 相邻点的间隔大小
- `out (Tensor, optional)`

```
torch.range(start, end, step=1, out=None) :
```

**还是推荐使用** `torch.arange()`

```
torch.zeros(*sizes, out=None) :
```

返回一个全为标量0的张量，形状由可变参数 `sizes` 定义

### 3. 索引，切片，连接，换位(Index, Slicing, Joining, Mutating)

```
torch.cat(inputs, dimension=0) :
```

在给定维度上对输入的张量序列 `seq` 进行连接操作。

- `inputs (sequence of Tensors)`
- `dimension (int optional)` - 沿着此维连接张量序列

```
torch.chunk(tensor, chunks, dim=0) :
```



在给定维度上将输入张量进行分块

- tensors(Tensors) - 待分块的输入张量
- chunks (int) - 分块的个数
- dim (int) - 沿着此维度

```
torch.gather(input, dim, index, out=None) :
```

沿给定轴 `dim`,将输入索引张量 `index` 指定位置的值进行聚合。

- input(Tensor) - 源张量
- dim(int) - 索引的轴
- index(LongTensor) - 聚合元素的下标
- out - 目标张量

```
torch.index_select(input, dim, index, out=None) :
```

沿指定维度对输入进行切片,取 `index` 中指定的相应项,然后返回一个新的张量,返回的张量与原始张量有相同的维度(在指定轴上),**返回的张量与原始张量不共享内存空间**

- input(Tensor) - 输入张量
- dim(int) - 索引的轴
- index(LongTensor) - 包含索引下标的一维张量
- out - 目标张量

```
torch.masked_select(input, mask, out=None) :
```

根据掩码张量 `mask` 中的二元值,取输入张量中的指定项,将取值返回到一个新的1D张量。

张量 `mask` 须跟 `input` 张量有相同的元素数目,但形状或维度不需要相同。**返回的张量不与原始张量共享内存空间**

- input(Tensor) - 输入张量
- mask(ByteTensor) - 掩码张量,包含了二元索引值
- out - 目标张量

```
torch.nonzero(input, out=None) :
```

返回一个包含输入 `input` 中非零元素索引的张量,输出张量中的每行包含输入中非零元素的索引

若输入 `input` 有 `n` 维,则输出的索引张量 `output` 形状为 `z * n`,这里`z`是输入张量 `input` 中所有非零元素的个数

- input(Tensor) - 输入张量
- out - 包含索引值的结果张量

```
torch.split(tensor, split_size, dim=0) :
```

将输入张量分割成相等形状的chunks(如果可分)。如果沿指定维的张量形状大小不能被 `split_size` 整分,则最后一个分块会小于其它分块。

- tensor(Tensor) - 待分割张量
- split\_size(int) - 单个分块的形状大小
- dim(int) - 沿着此维进行分割



```
torch.squeeze(input, dim=None, out=None) :
```

将输入张量形状中的 1 去除并返回，如果输入是形如(A \* 1 \* B \* 1 \* C \* 1 \* D)，那么输出形状就为：(A \* B \* C \* D)。

当给定 dim 时，则只在给定维度上进行挤压，如输入形状为(A \* 1 \* B)，squeeze(input, 0)，将会保持张量不变，只有用 squeeze(input, 1)，形状会变成(A \* B)。

#### 输入张量与返回张量共享内存

- input(Tensor) - 输入张量
- dim(int, optional) - 如果给定，则只在给定维度挤压
- out(Tensor, optional) - 输出张量

```
torch.stack(sequence, dim=0) :
```

沿着一个新维度对输入张量进行连接，序列中所有张量都应该为相同的形状。

- sequence(Sequence) - 待连接的张量序列
- dim(int) - 插入的维度

```
torch.t(input, out=None) :
```

输入一个矩阵(2维张量)，并转置0,1维，可以被视为 transpose(input, 0, 1) 的简写函数

- input(Tensor) - 输入张量
- out(Tensor, optional) - 结果张量

```
torch.transpose(input, dim0, dim1, out=None) :
```

返回输入矩阵 input 的转置，交换维度 dim0 和 dim1。**输入张量与输出张量共享内存。**

- input(Tensor) - 输入张量
- dim0(int) - 转置的第一维
- dim1(int) - 转置的第二维

```
torch.unbind(tensor, dim=0)[source] :
```

移除指定维度后，返回一个元组，包含了沿着指定维切片后的各个切片

- tensor(Tensor) - 输入张量
- dim(int) - 删除的维度

```
torch.unsqueeze(input, dim, out=None) :
```

返回一个新的张量，对输入的指定位置插入维度 1，**返回张量与输入张量共享内存，若 dim 为负，则将被转化为 dim+input.dim()+1**

- tensor(Tensor) - 输入张量
- dim(int) - 插入维度的索引
- out(Tensor, optional) - 结果张量

## 4. 随机抽样Random sampling

```
torch.manual_seed(seed) :
```

设定生成随机数的种子，并返回一个 torch.\_C.Generator 对象



```
torch.initial_seed() :
```

返回生成随机数的原始种子值

```
torch.rng_state()[source]
```

返回随机生成器状态(ByteTensor)

```
torch.set_rng_state(new_state)[source] :
```

设定随机生成器状态参数： new\_state(torch.ByteTensor) - 期望的状态

```
torch.default_generator
```

```
torch.bernoulli(input, out=None) :
```

从伯努利分布中抽取二元随机数(0或者1)，输入中所有值必须在[0, 1]区间，输出张量的第 i 个元素值，将以输入张量的第 i 个概率值等于 1。  
返回值将会是与输入相同大小的张量，每个值为0或1

- input(Tensor) - 输入为伯努利分布的概率值
- out(Tensor, optional)

```
torch.multinomial(input, num_samples, replacement=False, out=None) :
```

返回一个张量，每行包含从 input 相应行中定义的多项式分布中抽取的 num\_samples 个样本。

input 每行的值不需要总和为1,但必须非负且总和不能为0.

- input(Tensor) - 包含概率值的张量
- num\_samples(int) - 抽取的样本数
- replacement(bool, optional) - 布尔值，决定是否重复抽取
- out(Tensor, optional)

```
torch.normal(means, std, out=None) :
```

返回一个张量，包含从给定 means, std 的离散正态分布中抽取随机数，均值和标准差的形状不须匹配，但每个张量的元素个数须相同

- means(Tensor) - 均值
- std(Tensor) - 标准差
- out(Tensor, optional)

## 5. 序列化 Serialization

```
torch.save(obj, f, pickle_module, pickle_protocol=2) :
```

保存一个对象到一个硬盘文件上

- obj - 保存对象
- f - 类文件对象
- pickle\_module - 用于pickling元数据和对象的模块
- pickle\_protocol - 指定pickle protocol可以覆盖默认参数

```
torch.load(f, map_location=None, pickle_module=) :
```



从磁盘文件中读取一个通过 `torch.save()` 保存的对象，可通过参数 `map_location` 动态地进行内存重映射

- `f` - 类文件对象
- `map_location` - 一个函数或字典规定如何remap存储位置
- `pickle_module` - 用于unpickling元数据和对象的模块

## 6. 并行化 Parallelism

```
torch.get_num_threads() :
```

获得用于并行化CPU操作的OpenMP线程数

```
torch.set_num_threads(int) :
```

设定用于并行化CPU操作的OpenMP线程数

## 7. 数学操作 Math operations

### 7.1 Pointwise Ops

```
torch.abs(input, out=None) :
```

计算输入张量的每个元素绝对值

- `input(Tensor)` - 输入张量
- `out(Tensor, optional)` - 结果张量

```
torch.acos(input, out=None) :
```

返回一个新张量，包含输入张量每个元素的反余弦

- `input(Tensor)` - 输入张量
- `out(Tensor, optional)`

```
torch.add(input, value, out=None) :
```

对输入张量 `input` 逐元素加上标量值 `value`，并返回结果到一个新的张量。

- `input(Tensor)` - 输入张量
- `value(Number)` - 添加到输入每个元素的数
- `out(Tensor, optional)`

```
torch.addcddiv(tensor, value=1, tensor1, tensor2, out=None) :
```

用 `tensor2` 对 `tensor1` 逐元素相除，然后乘以标量值 `value` 并加到 `tensor` 上。

- `tensor(Tensor)` - 张量
- `value(Number, optional)` - 标量
- `tensor1(Tensor)` - 张量，作为分子
- `tensor2(Tensor)` - 张量，作为分母
- `out(Tensor, optional)`

```
torch.addcmul(tensor, value=1, tensor1, tensor2, out=None) :
```



用 `tensor2` 对 `tensor1` 逐元素相乘，并对结果乘以标量值 `value` 然后加到 `tensor`，张量形状不需要匹配，但元素数量必须一致。

- `tensor(Tensor)` - 张量
- `value(Number, optional)` - 标量，
- `tensor1(Tensor)` - 张量，乘子1
- `tensor2(Tensor)` - 张量，乘子2
- `out(Tensor, optional)`

`torch.asin(input, out=None) :`

返回一个新张量，包含输入 `input` 张量每个元素的反正弦函数

- `input(Tensor)` - 输入张量
- `out(Tensor, optional)`

`torch.atan(input, out=None) :`

返回一个新张量，包含输入 `input` 张量每个元素的反正切函数

- `input(Tensor)`
- `out(Tensor, optional)`

`torch.atan2(input1, input2, out=None) :`

返回一个新张量，包含两个输入张量 `input1` 和 `input2` 的反正切函数

- `input1(Tensor)` - 第一个输入张量
- `input2(Tensor)` - 第二个输入张量
- `out(Tensor, optional)`

`torch.ceil(input, out=None) :`

对输入 `input` 张量每个元素向上取整，即取不小于每个元素的最小整数，并返回结果到输出

- `input(Tensor)` - 输入张量
- `out(Tensor, optional)`

`torch.clamp(input, min, max, out=None) :`

将输入 `input` 张量每个元素值约束到区间`[min, max]`，并返回结果到一个新张量  
**也可以只设定 `min` 或只设定 `max`**

- `input(Tensor)` - 输入张量
- `min(Number)` - 限制范围下限
- `max(Number)` - 限制范围上限
- `out(Tensor, optional)`

`torch.cos(input, out=None) :`

返回一个新张量，包含输入 `input` 张量每个元素的余弦

- `input(Tensor)`
- `out(Tensor, optional)`



`torch.cosh(input, out=None) :`

- input(Tensor)
- out(Tensor, optional)

`torch.div(input, value, out=None) :`

将 input 逐元素除以标量值 value , 并返回结果到输出张量 out

- input(Tensor) - 输入张量
- value(Number) - 除数
- out(Tensor, optional)

`torch.exp(tensor, out=None) :`

返回一个新张量, 包含输入 input 张量每个元素的指数

- input(Tensor)
- out(Tensor, optional)

`torch.floor(input, out=None) :`

返回一个新张量, 包含输入 input 张量每个元素的floor, 即不大于元素的最大整数。

- input(Tensor)
- out(Tensor, optional)

`torch.fmod(input, divisor, out=None) :`

计算除法余数, 余数的正负与被除数相同

- input(Tensor)
- divisor(Tensor or float) - 除数
- out(Tensor, optional)

`torch.frac(tensor, out=None) :`

返回每个元素的分数部分

`torch.lerp(start, end, weight, out=None) :`

对两个张量以 start , end 做线性插值, 将结果返回到输出张量

$out = start + weight * (end - start)$

- start(Tensor) - 起始点张量
- end(Tensor) - 终止点张量
- weight(float) - 插值公式中的weight
- out(Tensor, optional)

`torch.log(input, out=None) :`

计算 input 的自然对数

`torch.log1p(input, out=None) :`





计算  $\text{input} + 1$  的自然对数  $y = \log(x + 1)$

**对值比较小的输入，此函数比 `torch.log()` 更准确**

- `input(Tensor)`
- `out(Tensor, optional)`

`torch.mul(input, value, out=None) :`

用标量值 `value` 乘以输入 `input` 的每个元素，并返回一个新的结果张量

`torch.mul(input, other, out=None) :`

两个张量 `input` , `other` 按**元素相乘**，并返回到输出张量，两个张量形状不须匹配，但总元素数须一致。当形状不匹配时，`input` 的形状作为输出张量的形状

- `input(Tensor)` - 第一个张量
- `other(Tensor)` - 第二个张量
- `out(Tensor, optional)`

`torch.neg(input, out=None) :`

返回一个新张量，包含输入 `input` 张量按元素取负。

`torch.pow(input, exponent, out=None) :`

对输入 `input` 按元素求 `exponent` 次幂，并返回结果张量。幂可以为 `float` 数或与 `input` 相同元素数的张量

- `input(Tensor)` - 输入张量
- `exponent(float or Tensor)` - 幂值
- `out(Tensor, optional)`

`torch.pow(base, input, out=None) :`

`base` 为标量浮点值，`input` 为张量。

- `base(float)` - 标量值，指数的底
- `input(Tensor)` - 幂值
- `out(Tensor, optional)`

`torch.reciprocal(input, out=None) :`

返回一个新张量，包含输入 `input` 张量每个元素的倒数，即  $1.0/x$

`torch.remainder(input, divisor, out=None) :`

返回一个新张量，包含输入 `input` 张量每个元素的除法余数，余数与除数有相同的符号。

- `input(Tensor)` - 被除数
- `divisor(Tensor or float)` - 除数
- `out(Tensor, optional)`

`torch.round(input, out=None) :`

返回一个新张量，将输入 `input` 张量每个元素四舍五入到最近的整数。



```
torch.rsqrt(input, out=None) :
```

返回一个新张量，包含输入 `input` 张量每个元素的平方根倒数。

```
torch.sigmoid(input, out=None) :
```

返回一个新张量，包含输入 `input` 张量每个元素的sigmoid值

```
torch.sign(input, out=None) :
```

符号函数：返回一个新张量，包含输入 `input` 张量每个元素的正负。

```
torch.sin(input, out=None) :
```

返回一个新张量，包含输入 `input` 张量每个元素的正弦。

```
torch.sinh(input, out=None) :
```

返回一个新张量，包含输入 `input` 张量每个元素的双曲正弦。

```
torch.sqrt(input, out=None) :
```

返回一个新张量，包含输入 `input` 张量每个元素的平方根。

```
torch.tan(input, out=None) :
```

返回一个新张量，包含输入 `input` 张量每个元素的正切。

```
torch.tanh(input, out=None) :
```

返回一个新张量，包含输入 `input` 张量每个元素的双曲正切。

```
torch.trunc(input, out=None) :
```

返回一个新张量，包含输入 `input` 张量每个元素的截断值，使更接近零。即有符号数的小数部分被舍弃。

## 7.2 Reduction Ops

```
torch.cumprod(input, dim, out=None) -> Tensor :
```

返回输入沿指定维度的累积积，如输入是一个N元向量，则结果也是一个N元向量，第  $i$  个输出元素值为  $y_i = x_1 * x_2 * x_3 * \dots * x_i$

- `input(Tensor)` - 输入张量
- `dim(int)` - 累积乘积操作的维度
- `out(Tensor, optional)`

```
torch.cumsum(input, dim, out=None) -> Tensor :
```

返回输入沿指定维度的累积和

```
torch.dist(input, other, p=2, out=None) -> Tensor :
```

返回  $(input - other)$  的  $p$  范数



- `input(Tensor)` - 输入张量
- `other(Tensor)` - 右侧输入张量
- `p(float, optional)` - 要计算的范数
- `out(Tensor, optional)`

`torch.mean(input) -> float :`

返回输入张量所有元素的均值

`torch.mean(input, dim, out=None) :`

返回输入张量给定维度 `dim` 上每行的均值，输出形状与输入相同，除了给定维度上为1。

`torch.median(input, dim=-1, values=None, indices=None) -> (Tensor, LongTensor) :`

返回输入张量给定维度每行的中位数，同时返回一个包含中位数的索引。`dim` 默认为输入张量的最后一维

- `input(Tensor)` - 输入张量
- `dim(int)` - 缩减的维度
- `values(Tensor, optional)` - 结果张量
- `indices(Tensor, optional)` - 返回的索引结果张量

`torch.mode(input, dim=-1, values=None, indices=None) -> (Tensor, LongTensor) :`

返回给定维度 `dim` 上，每行的众数值，同时返回一个索引张量。`dim` 值默认为输入张量的最后一维。输出形状与输入相同，除了给定维度上为1。

`torch.norm(input, p=2) -> float :`

返回输入张量 `input` 的 `p` 范数。

- `input(Tensor)` - 输入张量
- `p(float, optional)` - 范数计算中的幂指数值

`torch.norm(input, p, dim, out=None) -> Tensor :`

返回输入张量给定维度 `dim` 上每行的 `p` 范数。

`torch.prod(input) -> float :`

返回输入张量 `input` 所有元素的积

`torch.prod(input, dim, out=None) -> Tensor :`

返回输入张量给定维度上每行的积。

`torch.std(input) -> float :`

返回输入张量 `input` 所有元素的标准差

`torch.std(input, dim, out=None) :`

返回输入张量给定维度上每行的标准差。



```
torch.sum(input) -> float :
```

返回输入张量 input 所有元素的各

```
torch.sum(input, dim, out=None) -> Tensor :
```

返回输入张量是给定维度上每行的和

```
torch.var(input) -> float :
```

返回输入张量所有元素的方差

```
torch.var(input, dim, out=None) -> Tensor :
```

返回输入张量给定维度上每行的方差。

## 7.3 比较操作Comparison Ops

```
torch.eq(input, other, out=None) -> Tensor :
```

比较元素相等性，第二个参数可为一个数，或与第一个参数同类型形状的张量

- input(Tensor) - 待比较张量
- other(Tensor or float) - 比较张量或数
- out(Tensor, optional) - 输出张量，须为ByteTensor类型或与 input 同类型

```
torch.equal(tensor1, tensor2) -> bool :
```

若两个张量有相同的形状和元素值，则返回 True ， 否则 False 。

```
torch.ge(input, other, out=None) -> Tensor :
```

逐元素比较 input 和 other ， 即是否 input >= other

第二个参数可以为一个数或与第一个参数相同形状和类型的张量。

- input(Tensor) - 待对比的张量
- other(Tensor or float) - 对比的张量或 float 值
- out(Tensor, optional) - 输出张量，必须为 ByteTensor 或与第一个参数相同类型。

```
torch.gt(input, other, out=None) -> Tensor :
```

逐元素比较 input 和 other ， 是否 input > other 。若两个张量有相同的形状和元素值，则返回 True ， 否则 False 。第二个参数

可以为一个数或与第一个参数相同形状和类型的张量。

```
torch.kthvalue(input, k, dim=None, out=None) -> (Tensor, LongTensor) :
```

取输入张量 input 指定维度上第 k 个最小值，若不指定 dim ， 则默认为 input 的最后一维。返回一个元组，其中 indices 是原始输入张量 input 中沿 dim 维的第 k 个最小值下标。

- input(Tensor) - 输入张量
- k(int) - 第 k 个最小值
- dim(int, optional) - 沿着此维度进行排序
- out(tuple, optional) - 输出元组



```
torch.le(input, other, out=None) -> Tensor :
```

逐元素比较 input 和 other , 即是否  $\text{input} \leq \text{other}$  , 第二个参数可以为一个数或与第一个参数相同形状和类型的张量。

```
torch.lt(input, other, out=None) -> Tensor :
```

逐元素比较 input 和 other , 即是否  $\text{input} < \text{other}$

```
torch.max(input, dim, max=None, max_indices=None) -> (Tensor, LongTensor) :
```

返回输入张量给定维度上每行的最大值, 并同时返回每个最大值的位置索引。

- input(Tensor) - 输入张量
- dim(int) - 指定的维度
- max(Tensor, optional) - 结果张量, 包含给定维度上的最大值
- max\_indices(LongTensor, optional) - 包含给定维度上每个最大值的位置索引。

```
torch.min(input, dim, min=None, min_indices=None) -> (Tensor, LongTensor) :
```

返回输入张量给定维度上每行的最小值, 并同时返回每个最小值的位置索引。

```
torch.min(input, other, out=None) -> Tensor :
```

input 中逐元素与 other 相应位置的元素对比, 返回最小值到输出张量。两张量形状不需匹配, 但元素数须相同。

```
torch.ne(input, other, out=Tensor) -> Tensor :
```

逐元素比较 input 和 other , 即是否  $\text{input} \neq \text{other}$  。第二个参数可以为一个数或与第一个参数相同形状和类型的张量。

返回值: 一个 torch.ByteTensor 张量, 包含了每个位置的比较结果(如果  $\text{tensor} \neq \text{other}$  为 True , 返回 1 )。

```
torch.sort(input, dim=None, descending=False, out=None) -> (Tensor, LongTensor) :
```

对输入张量 input 沿着指定维度按升序排序, 如果不给定 dim , 默认为输入的最后一维。如果指定参数 descending 为 True , 则按降序排序。

返回两项: 重排后的张量, 和重排后元素在原张量的索引

- input(Tensor) - 输入张量
- dim(int, optional) - 沿此维排序, 默认为最后一维
- descending(bool, optional) - 布尔值, 默认升序

```
torch.topk(input, k, dim=None, largest=True, sorted=True, out=None) -> (Tensor, LongTensor) :
```

沿给定 dim 维度返回输入张量 input 中 k 个最大值, 不指定 dim , 则默认为最后一维, 如果 largest 为 False , 则返回最小的 k 个值。

## 7.4 其它操作 Other Operations

```
torch.cross(input, other, dim=-1, out=None) -> Tensor :
```



返回沿着维度 `dim` 上，两个张量 `input` 和 `other` 的叉积。`input` 和 `other` 必须有相同的形状，且指定的 `dim` 维上 `size` 必须为 3。如果不指定 `dim`，则默认为第一个尺度为 3 的维。

```
torch.diag(input, diagonal=0, out=None) -> Tensor :
```

如果输入是一个向量，则返回一个以 `input` 为对角线元素的2D方阵

如果输入是一个矩阵，则返回一个包含 `input` 为对角元素的1D张量

参数 `diagonal` 指定对角线：

- `diagonal = 0`, 主对角线
- `diagonal > 0`, 主对角线之上
- `diagonal < 0`, 主对角线之下

```
torch.histc(input, bins=100, min=0, max=0, out=None) -> Tensor :
```

计算输入张量的直方图。如果 `min` 和 `max` 都为0,则利用数据中的最大最小值作为边界。

```
torch.renorm(input, p, dim, maxnorm, out=None) -> Tensor :
```

返回一个张量，包含规范化后的各个子张量，使得沿着 `dim` 维划分的各子张量的 `p` 范数小于 `maxnorm`。如果 `p` 范数的值小于 `maxnorm`，则当前子张量不需要修改。

```
torch.trace(input) -> float :
```

返回输入2维矩阵对角元素的和(迹)

```
torch.tril(input, diagonal=0, out=None) -> Tensor :
```

返回一个张量，包含输入张量(2D张量)的下三角部分，其余部分设为0，参数 `diagonal` 控制对角线。

```
torch.triu (input, diagonal=0, out=None) -> Tensor :
```

返回一个张量，包含输入矩阵的上三角部分，其余被置为 0。

## 7.5 BLAS and LAPACK Operations

```
torch.dot(tensor1, tensor2) -> float :
```

计算两个张量的点乘，两个张量都为1-D向量

```
torch.eig(a, eigenvectors=False, out=None) -> (Tensor, Tensor) :
```

计算方阵 `a` 的特征值和特征向量。

- `a(Tensor)` - 方阵
- `eigenvectors(bool)` - 如果为 `True`，同时计算特征值和特征微量，否则只计算特征值  
返回值：
- `e(Tensor)` - `a` 的右特征向量
- `v(Tensor)` - 如果 `eigenvectors` 为 `True`，则为包含特征向量的张量，否则为空。

```
torch.inverse(input, out=None) -> Tensor :
```

对方阵 `input` 求逆



```
torch.mm(mat1, mat2, out=None) -> Tensor :
```

对矩阵 mat1 和 mat2 进行相乘。

```
torch.mv(mat, vec, out=None) -> Tensor :
```

对矩阵 mat 和向量 vec 进行相乘。

小礼物走一走，来简书关注我

赞赏支持

DL (/nb/27819157)

举报文章 © 著作权归作者所有



zhaoQiang012 (/u/94e624cec680)  
写了 8375 字，被 0 人关注，获得了 1 个喜欢  
(/u/94e624cec680)

+ 关注

喜欢



更多分享



写下你的评论...

评论

智慧如你，不想发表一点想法咩~

被以下专题收入，发现更多相似内容

+ 收入我的专题

推荐阅读

更多精彩内容 > (/)

五绝·悼金庸 (/p/8a2b57c018e9?utm\_campaign=mal...  
名师逝落鸿，传世射雕雄。驾鹤飘然去，幽怀泰斗公。

(/p/8a2b57c018e9?  
utm\_campaign=maleskine&utm\_content=note&utm\_

