MSELOSS

CLASS torch.nn.MSELoss(*size_average=None*, *reduce=None*, *reduction='mean'*) [SOURCE]

Creates a **criterion** that measures the mean squared error (squared L2 norm) between each element in the input x and target y.

The unreduced (i.e. with `reduction` set to `'none'`) loss can be described as:

$$\ell(x, y) = L = \{l_1, \dots, l_N\}^\top, \quad l_n = (x_n - y_n)^2,$$

where N is the batch size. If `reduction` is not `'none'` (default `'mean'`), then:

$$\ell(x, y) = \begin{cases} \text{mean}(L), & \text{if reduction} = \text{'mean'}; \\ \text{sum}(L), & \text{if reduction} = \text{'sum'}. \end{cases}$$

x and y are **tensors** of arbitrary shapes with a total of n elements each.

The mean operation still operates over all the elements, and divides by n.

The division by n can be avoided if one sets `reduction = 'sum'`.

Parameters:

- **size_average** (*bool, optional*) – **Deprecated** (see `reduction`). By default, the losses are averaged over each loss element in the batch. Note that for some losses, there are multiple elements per sample. If the field `size_average` is set to `False`, the losses are instead summed for each minibatch. Ignored when `reduce` is `False`. Default: `True`
- **reduce** (*bool, optional*) – **Deprecated** (see `reduction`). By default, the losses are averaged or summed over observations for each minibatch depending on `size_average`. When `reduce` is `False`, returns a loss per batch element instead and ignores `size_average`. Default: `True`
- **reduction** (*str, optional*) – Specifies the reduction to apply to the output: `'none'` | `'mean'` | `'sum'`. `'none'`: no reduction will be applied, `'mean'`: the sum of the output will be divided by the number of elements in the output, `'sum'`: the output will be summed. Note: `size_average` and `reduce` are in the process of being **deprecated**, and **in the meantime**, specifying either of those two args will override `reduction`. Default: `'mean'`

Shape:

- Input: $(*)$, where $*$ means any number of dimensions.
- Target: $(*)$, same shape as the input.

Examples:

```
>>> loss = nn.MSELoss()
>>> input = torch.randn(3, 5, requires_grad=True)
>>> target = torch.randn(3, 5)
>>> output = loss(input, target)
>>> output.backward()
```

‹ Previous

Next ›