NOTE 1. CROSS ENTROPY LOSS

class ***torch.nn.CrossEntropyLoss(weight=None, size_ average=True, ignore_ index=-100, reduce=True)***

This criterion combines *LogSoftMax* and *NLLLoss* is one single class. It is useful when training a classification problem with C classes. If provided, the optional argument *weight* should be a *1D Tensor* assigning weight to each of the classes. This is particularly useful when you have an unbalanced training set.

The *input* is expected to contain scores for each class. *input* has to be a *2D Tensor* of size *(minibatch, C)*. This criterion expects a class index *(0 to C-1)* as the *target* for each value of a *1D Tensor* of size of *minibatch*.

The loss can be described as:

$$loss(x, class) = -\log \frac{e^{x[class]}}{\sum_j e^{x[j]}} = -x[class] + \log \sum_j e^{x[j]}$$

or in the case of the *weight* argument being specified:

$$loss(x, class) = weight[class] \cdot (-x[class] + \log \sum_j e^{x[j]})$$

The losses are averaged across observations for each minibatch.

Parameter:

- weight(Tensor, optional) - a manual rescaling weight given to each class. If given, has to be a *Tensor* of size C
- size_average(bool, optional) - By default, the losses are averaged over observations for each minibatch. However, if the field size_average is set to *False*, the losses are instead summed for each minibatch. Ignored if reduce is *False*
- ignore_index(int, optional) - Specifies a target value that is ignored and doesn't contribute to the input gradient. When size_average is *True*, the loss is averaged over non-ignored targets
- reduce(bool, optional) - 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*

Shape:

- Input: *(N, C)* where $C = number\ of\ classes$
- Target: *(N)* where each value is $0 \leq targets[i] \leq C - 1$
- Output: scalar. If reduce is *False*, then *(N)* instead

Example:

```python
import torch
import torch.nn as nn

# y_pred.shape = torch.Size([2, 4])
# y.shape = torch.Size([2])
y_pred = torch.tensor([[0.1, 0.1, 0.2, 0.6], [0.1, 0.1, 0.2, 0.6]])
y = torch.tensor([1, 1])
print('y_pred: ', y_pred)
print('y: ', y)
criterion = nn.CrossEntropyLoss()
loss = criterion(y_pred, y)
print('loss: ', loss)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

PS D:\Downloads\QA-20220824T162725Z-001\QA> python -u "d:\Downloads\QA-20220824T1
y_pred:  tensor([[0.1000, 0.1000, 0.2000, 0.6000],
        [0.1000, 0.1000, 0.2000, 0.6000]])
y:  tensor([1, 1])
loss:  tensor(1.5590)
```

$$loss(x_1, class_1) = -\log \frac{e^{0.1}}{e^{0.1} + e^{0.1} + e^{0.2} + e^{0.6}} = 1.5590$$

$$loss(x_2, class_2) = -\log \frac{e^{0.1}}{e^{0.1} + e^{0.1} + e^{0.2} + e^{0.6}} = 1.5590$$

$$loss(x, class) = \frac{1}{2} \cdot (loss(x_1, class_1) + loss(x_2, class_2)) = 1.5590$$