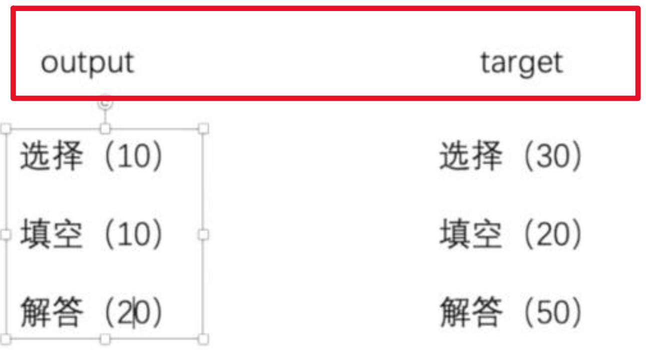
损失函数与反向传播

2025年11月25日 21:37

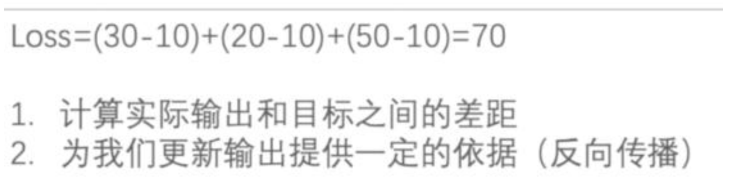
- Loss function backward()



Loss=(30-10)+(20-10)+(50-10)=70

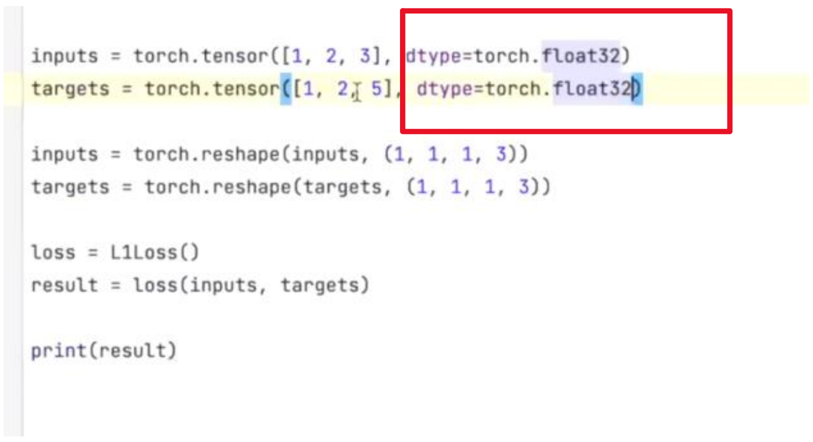
Loss function

衡量输出结果与预期的误差



Loss=(30-10)+(20-10)+(50-10)=70

1. 计算实际输出和目标之间的差距
2. 为我们更新输出提供一定的依据（反向传播）



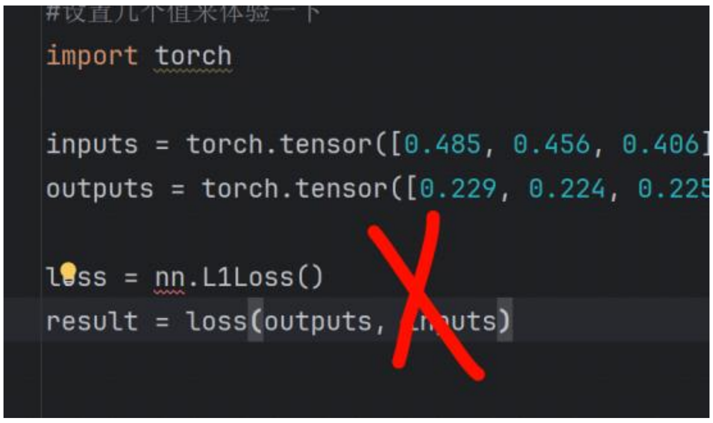
```
inputs = torch.tensor([1, 2, 3], dtype=torch.float32)
targets = torch.tensor([1, 2, 5], dtype=torch.float32)

inputs = torch.reshape(inputs, (1, 1, 1, 3))
targets = torch.reshape(targets, (1, 1, 1, 3))

loss = L1Loss()
result = loss(inputs, targets)

print(result)
```

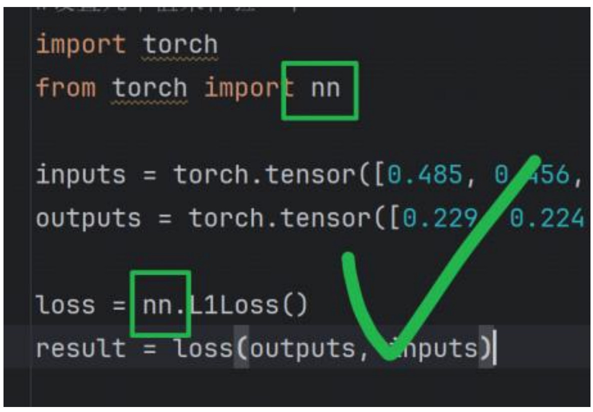
损失函数衡量模型效果与预期的差距，再通过微调，反向传回模型，让模型不断优化



```
#设置几个值来体验一下
import torch

inputs = torch.tensor([0.485, 0.456, 0.406]
outputs = torch.tensor([0.229, 0.224, 0.225

loss = nn.L1Loss()
result = loss(outputs, inputs)
```

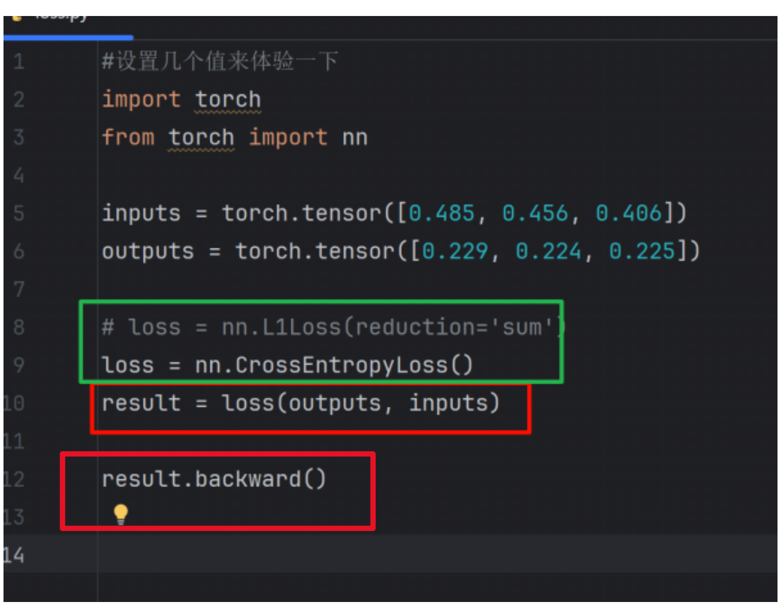


```
import torch
from torch import nn

inputs = torch.tensor([0.485, 0.456,
outputs = torch.tensor([0.229, 0.224

loss = nn.L1Loss()
result = loss(outputs, inputs)
```

我import的有谁
我才能直接写谁
没有import nn
就不能直接nn开头



```
#设置几个值来体验一下
import torch
from torch import nn

inputs = torch.tensor([0.485, 0.456, 0.406])
outputs = torch.tensor([0.229, 0.224, 0.225])

# loss = nn.L1Loss(reduction='sum')
loss = nn.CrossEntropyLoss()
result = loss(outputs, inputs)

result.backward()

```

Backward()就是把哪个值传回去，反向传播
我们把损失函数的计算结果放在result里，所以返回的就是result