1. Tensor

创建tensor

import torch

# torch.empty, torch.rand  
x = torch.tensor([5.5, 3])  
print(x)  
x = torch.zeros(5, 3, dtype=torch.long)  
print(x)  
x = x.new\_ones(5, 3, dtype=torch.double)   
print(x)

#对象的size是相同的，覆盖了dtype  
x = torch.randn\_like(x, dtype=torch.float)  
print(x)

几种加法运算

y = torch.rand(5, 3)

# 方法1

z=x+y

# 方法2

z=torch.add(x, y)

# 方法3

result = torch.empty(5, 3)  
torch.add(x, y, out=result)  
print(result)

# 方法4，这里会改变y的值

y.add\_(x)  
print(y)

任何以”\_”结尾的操作都会用结果替换原变量，如”x.copy\_(y)”, “x.t\_()”都会改变“x”。

其他api

# 索引与numpy一样

print(x[:, 1])

# torch.view类似于numpy的reshape，改变tensor的形状

x = torch.randn(4, 4)  
y = x.view(16)  
z = x.view(-1, 8) # size -1 从其他维度推断

# 与numpy的转化

a = torch.ones(5)  
b = a.numpy() # tensor转为numpy  
c = torch.from\_numpy(b) # numpy转tensor

转到gpu，使用.to方法将tensor移到gpu或cpu

x = torch.ones(5)  
print(x)  
if torch.cuda.is\_available():  
 device = torch.device("cuda") # a CUDA 设备对象  
 y = torch.ones\_like(x, device=device) # 直接从GPU创建变量  
 x = x.to(device)

z = x + y  
 print(z)  
 print(z.to("cpu", torch.double))

1. Autograd自动求导

x = torch.ones(2, 2, requires\_grad=True)  
print(x)  
y = x + 2  
print(y)  
z = y \* y \* 3  
out = z.mean()  
print(z, out)

# 反向传播  
out.backward()  
print(x.grad)

1. nn
2. aa