```
import torch             # torch.nn para Neural Networks.

P = 100
N = 8
H = N+1
M = 1

x = torch.randn( P, N).sign()
z = torch.prod( x, dim=1).view( P, M)

model = torch.nn.Sequential(        # Seq es feedforward.
        torch.nn.Linear( N, H),     # Linear son los pesos.
        torch.nn.Tanh(),            # Tanh es la activacion.
        torch.nn.Linear( H, M),     # Linear incluye los bias.
        torch.nn.Tanh() )

costf = torch.nn.MSELoss( reduction='sum')  # Puedo usar la suma.

lr = 1e-2
for t in range(999):
    y = model( x)                   # Puedo usarlo como funcion.
    model.zero_grad()               # Reseteo los grad antes de back.
    error = costf( y, z)
    error.backward()
    with torch.no_grad():
        for param in model.parameters():
            param -= lr * param.grad
    if not t%100:
        print( t, error.item()/P)
```

```
# P, N, H, y M igual que antes.

device = torch.device( "cuda:0" if torch.cuda.is_available() else "cpu")

x = torch.randn( P, N).sign()
z = torch.prod( x, dim=1).view( P, M)
x, z = x.to(device), z.to( device)          # Muevo los datos a gpu.


class mlp( torch.nn.Module):        # Module para hacer modelos propios.
    def __init__( _, isize, hsize, osize):      # Uso _ en lugar de self.
        super().__init__()
        _.l1 = torch.nn.Linear( isize, hsize)   # Params de modelo.
        _.l2 = torch.nn.Linear( hsize, osize)

    def forward( _, x):
        h = torch.tanh( _.l1( x))               # Grafo de computo.
        y = torch.tanh( _.l2( h))
        return y


model = mlp( N, H, M).to( device)           # Muevo los parametros a gpu.
optim = torch.optim.SGD( model.parameters(), lr=0.1)
costf = torch.nn.MSELoss()

t, E = 0, 1.
model.train()
while E>=0.01 and t<9999:
    y = model( x)
    optim.zero_grad()                       # Optim sabe que resetear.
    error = costf( y, z)
    error.backward()
    optim.step()                            # step aplica los gradientes.
    E = error.item()
    t += 1
    if t%100==0:
        print( t, E)

model.eval()                                # train/eval no son necesarios
with torch.no_grad():                       # para este modelo sencillo.
    y = model( x)
    E = costf( y, z).item()
print( E)
```