torch.optim包含了SGD，Adagrad，RMSprop，Adam，LBFGS等

**torch.optim.Adam(params,lr)**

返回一个优化器对象

**参数解释：**

params为模型参数

lr为学习率

**反向传播优化**

设optimizer是optim.Adam的实例，则可用optimizer.zero\_grad()清空梯度，用optimizer.step()更新参数

optimizer = torch.optim.Adam(net.parameters(),lr=learning\_rate)

for input, target in dataset:

optimizer.zero\_grad()*#清空优化器所管理的参数的梯度*

output = model(input)  
loss = loss\_fn(output, target)

loss.backward()

optimizer.step()*#更新参数*

**改变学习率**

def adjust\_learning\_rate(optimizer, shrink\_factor):

for param\_group in optimizer.param\_groups:

param\_group['lr'] = param\_group['lr'] \* shrink\_factor

net = nn.Linear(3,4)

learning\_rate = 4e-4

optimizer = torch.optim.Adam(net.parameters(),lr=learning\_rate)

adjust\_learning\_rate(optimizer, 0.8)

print(optimizer.param\_groups)

>>>

[{'params': [Parameter containing:

tensor([[ 0.0387, 0.4239, -0.5624],

[ 0.4742, 0.3982, -0.1927],

[-0.0042, 0.1453, -0.4837],

[-0.0542, 0.3256, 0.1568]], requires\_grad=True), Parameter containing:

tensor([ 0.5454, 0.4124, -0.0953, -0.5062], requires\_grad=True)], 'lr': 0.00032, 'betas': (0.9, 0.999), 'eps': 1e-08, 'weight\_decay': 0, 'amsgrad': False}]

**torch.optim.LBFGS(params)**

设optimizer是optim.LBFGS的实例，则可用optimizer.zero\_grad()清空梯度，用optimizer.step(closure)更新参数