

**from** \_\_future\_\_ **import** print\_function

**import** matplotlib.pyplot **as** plt

**import** torch **import** torch.nn.functional **as** F

**from** sklearn.datasets **import** load\_iris

**from** torch.autograd **import** Variable

**from** torch.optim **import** SGD

**import** os

os.environ['KMP\_DUPLICATE\_LIB\_OK'] = 'TRUE'

use\_cuda = torch.cuda.is\_available()

print("use\_cuda: ", use\_cuda)

iris = load\_iris()

print(iris.keys())

x = iris['data'] *# 特征信息*

y = iris['target'] *# 目标分类*

print(x.shape) *# (150, 4)*

print(x.shape) *# (150,)*

print(y) x = torch.FloatTensor(x)

y = torch.LongTensor(y)

x, y = Variable(x), Variable(y)

**class** **Net**(torch.nn.Module):

""" 定义网络 """

**def** **\_\_init\_\_**(self, n\_feature, n\_hidden, n\_output):

""" 初始化函数，接受自定义输入特征维数，隐藏层特征维数，输出层特征维数 """ super(Net, self).\_\_init\_\_()

self.hidden = torch.nn.Linear(n\_feature, n\_hidden) *# 一个线性隐藏层*

self.predict = torch.nn.Linear(n\_hidden, n\_output) *# 线性输出层*

**def** **forward**(self, x): """ 前向传播过程 """

x = F.sigmoid(self.hidden(x))

x = self.predict(x)

out = F.log\_softmax(x, dim=1)

**return** out

net = Net(n\_feature=4, n\_hidden=5, n\_output=3)

print(net)

**if** use\_cuda:

x = x.cuda()

y = y.cuda()

net = net.cuda()

optimizer = SGD(net.parameters(), lr=0.5)

iter\_num = 1000  
px, py = [], []  
  
plt.rcParams['font.sans-serif'] = ['STSong'] # 用来正常显示中文标签  
plt.rcParams['axes.unicode\_minus'] = False # 用来正常显示负号  
  
for i in range(iter\_num):  
 # 数据集传入网络前向计算  
 prediction = net(x)  
 # 计算loss  
 loss = F.nll\_loss(prediction, y)  
 # 这里也可用CrossEntropyLoss  
 # loss = loss\_func(prediction, y)  
 # 清除网络状态  
 optimizer.zero\_grad()  
 # loss 反向传播  
 loss.backward()  
 # 更新参数  
 optimizer.step()  
 # 打印每次迭代的损失情况，并记录当前的index 和 loss  
 print(i, " loss: ", loss.item())  
 px.append(i)  
 py.append(loss.item())  
 #每10次迭代绘制训练动态  
 if i % 10 == 0:  
 # 动态画出loss走向 结果：loss.png  
 plt.cla()  
 plt.title(u'训练过程的loss曲线')  
 plt.xlabel(u'迭代次数')  
 plt.ylabel('损失')  
 plt.plot(px, py, 'r-', lw=1)  
 plt.text(0, 0, 'Loss=%.4f' % loss.item(), fontdict={'size': 20, 'color': 'red'})  
 plt.pause(0.1)  
 if i == iter\_num - 1:  
 # 最后一个图像定格  
 plt.show()