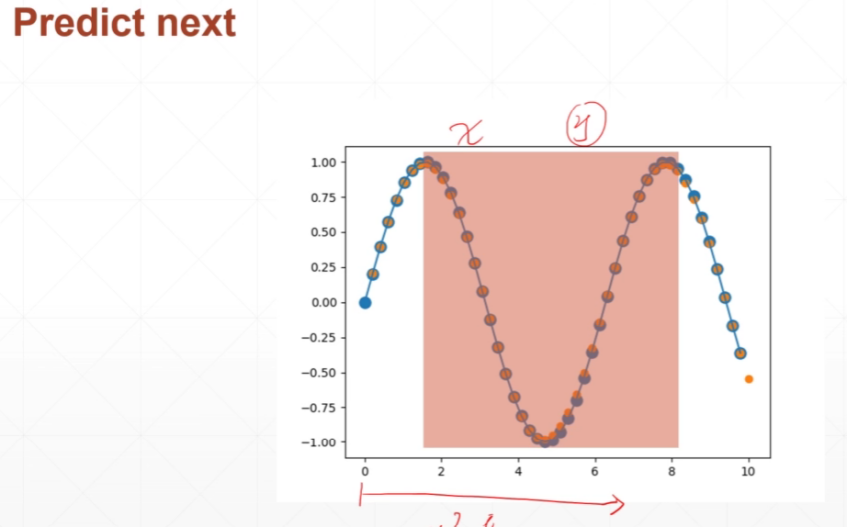
给出一段波形，希望预测出下一个时间段的波形



|  |
| --- |
| # name:shuaiqiuping  # university:sichuan normal unniversity  # time:2022/4/22 10:24  import torch  import torch.nn as nn  import torch.optim as optim  import numpy as np  from matplotlib import pyplot as plt  num\_time\_steps=50#一段波形里面50个点  input\_size=1#相当于有50个单词，1句话，每个单词用一个实数表示【50，1，1】  hidden\_size=16#memory维度定义为16，最终输出的out维度也一定是16，整合起来【50，1，16】  output\_size=1  lr=0.01  class Net(nn.Module):  def \_\_init\_\_(self,):  super(Net, self).\_\_init\_\_()  self.rnn=nn.RNN(  input\_size=input\_size,  hidden\_size=hidden\_size,  num\_layers=1,  batch\_first=True#因为batch放在第一个维度  )  for p in self.rnn.parameters():  nn.init.normal\_(p,mean=0.0,std=0.001)  #用一个线性层把16维变成1维  self.linear=nn.Linear(hidden\_size,output\_size)  def forward(self,x,hidden\_prev):  # x:[1,50,1]  #hidden\_prev:[b,16]  out,hidden\_prev=self.rnn(x,hidden\_prev)#out[b,50,16]h[b,16]  out=out.view(-1,hidden\_size)#将out打平送到线性层[b,50\*16]  out =self.linear(out)#[b,1]  out=out.unsqueeze(dim=0)#在第0维的位置上添加1维[1,seq,1],这里为什么要用三个维度，因为这是从网络中出去的预测值，要和y做一个均方差计算loss  return out,hidden\_prev  # x=torch.randn(1,50,1)  # h=torch.zeros(1,1,16)  # model=Net()  # output,hidden\_prev=model(x,h)  # print(output.shape,hidden\_prev.shape)  #torch.Size([1, 50, 1]) torch.Size([1, 1, 16])  model=Net()  criterion=nn.MSELoss()  optimizer=optim.Adam(model.parameters(),lr)  hidden\_prev=torch.zeros(1,1,hidden\_size)#将hidden\_prev初始化  for iter in range(6000):  start=np.random.randint(3,size=1)[0]#随便搞一个3以内的数作为起始点  time\_steps=np.linspace(start,start+10,num\_time\_steps)#在起始点->起始点+10这段数据内取50个点  data=np.sin(time\_steps)#[1,50]  data=data.reshape(num\_time\_steps,1)#[50,1]  x=torch.tensor(data[:-1]).float().view(1,num\_time\_steps-1,1)#0~48 [1,49,1]  y=torch.tensor(data[1:]).float().view(1,num\_time\_steps-1,1)#1~49 [1,49,1]  #(x,y)是一对数据对  output,hidden\_prev=model(x,hidden\_prev)#[b,50,1][b,1,16]  hidden\_prev=hidden\_prev.detach()#detach的作用是深拷贝解决tenso.date()的安全性  loss=criterion(output,y)  model.zero\_grad()  loss.backward()  optimizer.step()  if iter%100==0:  print("第{}次循环:loss:{}".format(iter,loss.item()))  #下面是做预测  #生成数据  start=np.random.randint(3,size=1)[0]  time\_steps=np.linspace(start,start+10,num\_time\_steps)  data=np.sin(time\_steps)  data=data.reshape(num\_time\_steps,1)  x=torch.tensor(data[:-1]).float().view(1,num\_time\_steps-1,1)  y=torch.tensor(data[1:]).float().view(1,num\_time\_steps-1,1)  #做预测，前面的代码训练好了以后，我们随便生成一个数据x，用它来预测下一个数据，再用这个数据预测下一个数据，循环，这样我们就可以得出一条预测曲线  predictions=[]  input=x[:,0,:] #x[1,49,1]=>[1,1]  for \_ in range(x.shape[1]):#从0到48  input =input.view(1,1,1)#[1,1]=>[1,1,1]  (pred,hidden\_prev)=model(input,hidden\_prev)  input=pred  predictions.append(pred.detach().numpy())  #画图  x = x.data.numpy().ravel()  y = y.data.numpy()  plt.scatter(time\_steps[:-1], x.ravel(), s=90)  plt.plot(time\_steps[:-1], x.ravel())  plt.scatter(time\_steps[1:], predictions)  plt.show() |

