PyTorch 深度学习实践 第3讲

第3讲 梯度下降法 源代码

B站 刘二大人 ，传送门PyTorch 深度学习实践 梯度下降法

深度学习算法中，并没有过多的局部最优点。

|  |
| --- |
| import matplotlib.pyplot as plt    # prepare the training set  x\_data = [1.0, 2.0, 3.0]  y\_data = [2.0, 4.0, 6.0]    # initial guess of weight  w = 1.0    # define the model linear model y = w\*x  def forward(x):  return x\*w    #define the cost function MSE  def cost(xs, ys):  cost = 0  for x, y in zip(xs,ys):  y\_pred = forward(x)  cost += (y\_pred - y)\*\*2  return cost / len(xs)    # define the gradient function gd  def gradient(xs,ys):  grad = 0  for x, y in zip(xs,ys):  grad += 2\*x\*(x\*w - y)  return grad / len(xs)    epoch\_list = []  cost\_list = []  print('predict (before training)', 4, forward(4))  for epoch in range(100):  cost\_val = cost(x\_data, y\_data)  grad\_val = gradient(x\_data, y\_data)  w-= 0.01 \* grad\_val # 0.01 learning rate  print('epoch:', epoch, 'w=', w, 'loss=', cost\_val)  epoch\_list.append(epoch)  cost\_list.append(cost\_val)    print('predict (after training)', 4, forward(4))  plt.plot(epoch\_list,cost\_list)  plt.ylabel('cost')  plt.xlabel('epoch')  plt.show() |

随机梯度下降法在神经网络中被证明是有效的。效率较低(时间复杂度较高)，学习性能较好。

随机梯度下降法和梯度下降法的主要区别在于：

1、损失函数由cost()更改为loss()。cost是计算所有训练数据的损失，loss是计算一个训练函数的损失。对应于源代码则是少了两个for循环。

2、梯度函数gradient()由计算所有训练数据的梯度更改为计算一个训练数据的梯度。

3、本算法中的随机梯度主要是指，每次拿一个训练数据来训练，然后更新梯度参数。本算法中梯度总共更新100(epoch)x3 = 300次。梯度下降法中梯度总共更新100(epoch)次。

|  |
| --- |
| import matplotlib.pyplot as plt    x\_data = [1.0, 2.0, 3.0]  y\_data = [2.0, 4.0, 6.0]    w = 1.0    def forward(x):  return x\*w    # calculate loss function  def loss(x, y):  y\_pred = forward(x)  return (y\_pred - y)\*\*2    # define the gradient function sgd  def gradient(x, y):  return 2\*x\*(x\*w - y)    epoch\_list = []  loss\_list = []  print('predict (before training)', 4, forward(4))  for epoch in range(100):  for x,y in zip(x\_data, y\_data):  grad = gradient(x,y)  w = w - 0.01\*grad # update weight by every grad of sample of training set  print("\tgrad:", x, y,grad)  l = loss(x,y)  print("progress:",epoch,"w=",w,"loss=",l)  epoch\_list.append(epoch)  loss\_list.append(l)    print('predict (after training)', 4, forward(4))  plt.plot(epoch\_list,loss\_list)  plt.ylabel('loss')  plt.xlabel('epoch')  plt.show() |

————————————————

版权声明：本文为CSDN博主「错错莫」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/bit452/article/details/109637108