## 인공지능 과제 4 **Chapter 5. Exercise**

2016920020 서예찬

|  |
| --- |
| 코드 |
| %matplotlib inline  import numpy as np  import torch  import torch.optim as optim  import matplotlib.pyplot as plt  torch.set\_printoptions(edgeitems=2, linewidth=75)    t\_c = [0.5, 14.0, 15.0, 28.0, 11.0, 8.0, 3.0, -4.0, 6.0, 13.0, 21.0]  t\_u = [35.7, 55.9, 58.2, 81.9, 56.3, 48.9, 33.9, 21.8, 48.4, 60.4, 68.4]  t\_c = torch.tensor(t\_c)  t\_u = torch.tensor(t\_u)    def model(t\_u, w1, w2, b):  return w2 \* t\_u \*\* 2 + w1 \* t\_u + b  #def model(t\_u,w,b):  # return w \* t\_u + b    def loss\_fn(t\_p, t\_c):  squared\_diffs = (t\_p - t\_c)\*\*2  return squared\_diffs.mean()    def training\_loop(n\_epochs, optimizer, params, t\_u, t\_c):  for epoch in range(1, n\_epochs + 1):  t\_p = model(t\_u, \*params)  loss = loss\_fn(t\_p, t\_c)    optimizer.zero\_grad()  loss.backward()  optimizer.step()    if epoch < 3:  print('Epoch %d, Loss %f' % (epoch, float(loss)))  if epoch % 2000 == 0:  print('Epoch %d, Loss %f' % (epoch, float(loss)))  return params    params = torch.tensor([1.0, 1.0, 0.0], requires\_grad = True)  #params = torch.tensor([1.0, 0.0], requires\_grad = True)  learning\_rate = 1e-1  optimizer = optim.Adam([params], lr = learning\_rate)    params = training\_loop(n\_epochs = 50000, optimizer = optimizer, params = params, t\_u = t\_u,t\_c = t\_c) |
| %matplotlib inline  from matplotlib import pyplot as plt    t\_p = model(t\_u,\*params)  fig = plt.figure(dpi = 600)  plt.xlabel("Temperature(℉)")  plt.ylabel("Temperature(℃")  plt.plot(t\_u.numpy(), t\_p.detach().numpy())  plt.plot(t\_u.numpy(),t\_c.numpy(), 'o') |

|  |
| --- |
| 그래프 |
|  |
| 결과 분석 |
| a) What parts of the training loop, and so on, need to change to accommodate this  redefinition?    def model (t\_u,w,b)  return w \* t\_u + b  ->  def model(t\_u, w1, w2, b):  return w2 \* t\_u \*\* 2 + w1 \* t\_u + b    params = torch.tensor([1.0, 0.0], requires\_grad = True)  ->  params = torch.tensor([1.0, 1.0, 0.0], requires\_grad = True)      b) What parts are agnostic to swapping out the model?  매개변수를 넘겨주는 부분을 제외하고는 변하지 않음.    c) Is the resulting loss higher or lower after training?  w \* t\_u + b :  Epoch 44000, Loss 2.927645  Epoch 46000, Loss 2.927647  Epoch 48000, Loss 2.927646  Epoch 50000, Loss 2.974464    w2 \* t\_u \*\* 2 + w1 \* t\_u + b :  Epoch 44000, Loss 2.094048  Epoch 46000, Loss 132.052597  Epoch 48000, Loss 2.615655  Epoch 50000, Loss 33.471905    d) Is the actual result better or worse?  전반적으로 이전보다 더 낮은 loss를 만들어 낼 수는 있었지만 시행 중에 Loss가 증가하는 경우가 있었음.  이전의 코드가 더 안정적인 결과를 보임. |