train_LaengeNet_all

February 7, 2022

[1]:
```
from Neural_Nets.LaengeNet.Development.LaengeNetTorch import LaengeNet,␣
↪LaengeNetLossFunc
from Neural_Nets.ThermoDataset.Development.ThermoDataset import ThermoDataset
from Neural_Nets.ThermoNetActFuncs.Development.ThermoNetActFuncs import␣
↪ChenSundman, Softplus
from Utils.PlotHandler.Development.PlotHandler import PlotHandler
import torch
from torch.utils.data import DataLoader, Dataset
import torch.nn as nn
from torch.optim import Rprop
from Data_Handling.SGTEHandler.Development.SGTEHandler import SGTEHandler
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
def epoch(net: LaengeNet, dataloader, loss_func, optimizer):
    epoch_losses = np.zeros([len(dataloader), ])

    for i, (temp, g, s, h, c) in enumerate(dataloader):
        temp = temp.unsqueeze(-1)

        # Forward pass
        gibbs_energy, entropy, enthalpy, heat_cap = net(temp, temp, temp, temp,␣
↪debug=True)

        # Get the loss
        loss = loss_func(gibbs_energy.float(), g.float(), entropy.float(), s.
↪float(), enthalpy.float(), h.float(),
                         heat_cap.float(), c.float(), debug=False)

        # Backward pass
        net.zero_grad()
        loss.backward()
        optimizer.step()
        epoch_losses[i] = loss

    mean_epoch_loss = epoch_losses.mean()
    #print('Mean epoch loss: ', mean_epoch_loss)
```

```
    return mean_epoch_loss
```

[3]:
```
def train(net, dataset):
    # Hyperparameters
    n_epochs = 200
    lr = 0.01
    batch_size = 64
    loss_weights = [1, 1300000, 0.01, 300000]
    #loss_weights = [1, 0, 0, 0]

    # Data
    dataloader = DataLoader(dataset, batch_size=batch_size, shuffle=False)

    # Optimizer
    optimizer = Rprop(net.parameters(), lr=lr)
    loss_func = LaengeNetLossFunc(weights=loss_weights)

    losses = []

    last_increase = 0

    for i in range(n_epochs):
        #print('-----\nEpoch %i:\n' % i)
        loss = epoch(net, dataloader, loss_func, optimizer)
        #if i > 0:
            #if (losses[-1] - loss)/loss < 2e-5 and i - last_increase > 50:
                #print('Learning rate increased')
                #last_increase = i
                #lr *= 2
                #optimizer = Rprop(net.parameters(), lr=lr)
        losses.append(loss)

    return losses
```

[4]:
```
net = LaengeNet(init_args=(0.1, 0.2), init_func=nn.init.uniform_,␣
↪hidden_dim_sub_net_2=16)
theta_E_real = 1.054571817 * 10 ** -34 * 1.7 * 10 ** 13 / (1.380649 * 10 ** -23)

net.sub_net_1.act_1._initialize_parameters(theta_E_real, 0., 10., 10.)

element = 'Fe'
phase = ['BCC_A2']
start_temp, end_temp = 200, 2000

dataset = ThermoDataset(element, phase, scaling=False, step=1,␣
↪start_temp=start_temp, end_temp=end_temp)
```

```
losses = train(net, dataset)
```

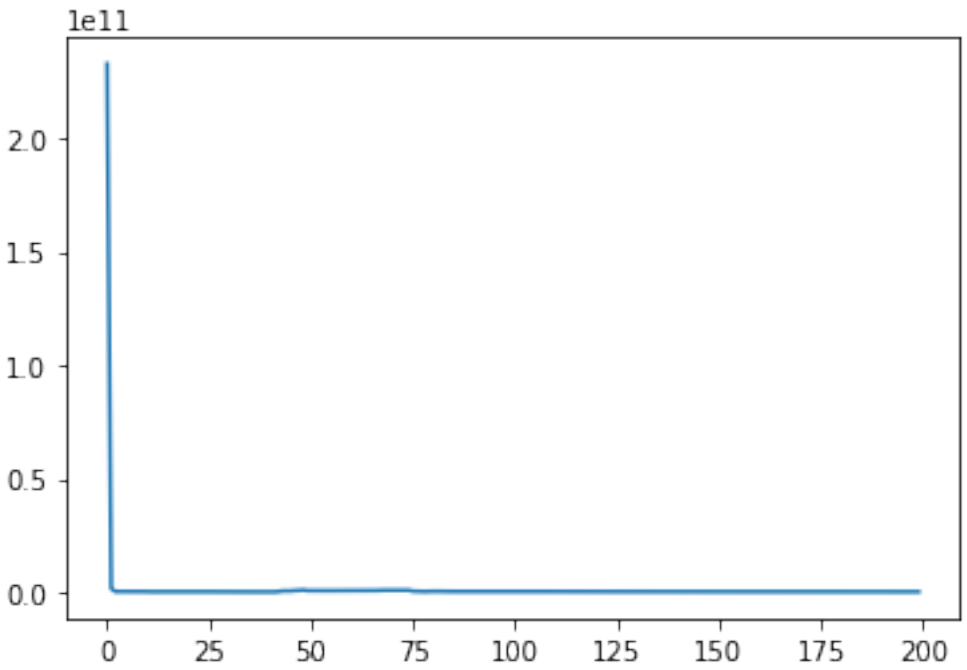
```
Fe successfully selected!


C:\Users\danie\anaconda3\envs\5_Programmcodes\lib\site-
packages\torch\nn\modules\loss.py:520: UserWarning: Using a target size
(torch.Size([64])) that is different to the input size (torch.Size([64, 1])).
This will likely lead to incorrect results due to broadcasting. Please ensure
they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
C:\Users\danie\anaconda3\envs\5_Programmcodes\lib\site-
packages\torch\nn\modules\loss.py:520: UserWarning: Using a target size
(torch.Size([37])) that is different to the input size (torch.Size([37, 1])).
This will likely lead to incorrect results due to broadcasting. Please ensure
they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
```

[5]:
```
print('theta_E: ', net.sub_net_1.act_1.theta_E)
print('E0: ', net.sub_net_1.act_1.E0)
print('a: ', net.sub_net_1.act_1.a)
print('b: ', net.sub_net_1.act_1.b)
```

```
theta_E:  Parameter containing:
tensor(129.8500)
E0:  Parameter containing:
tensor(4335.0952, requires_grad=True)
a:  Parameter containing:
tensor(24.4665, requires_grad=True)
b:  Parameter containing:
tensor(9.8453, requires_grad=True)
```

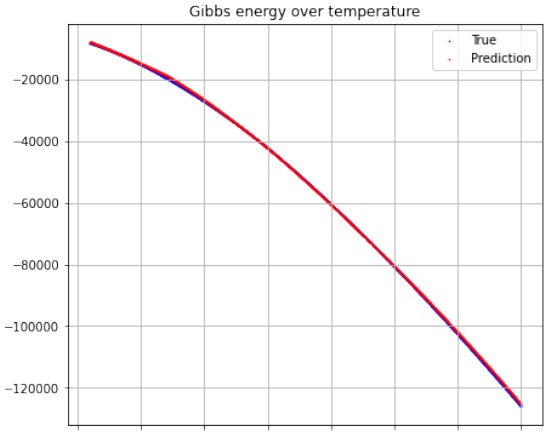
[6]:
```
plt.plot(range(len(losses)), losses)
```

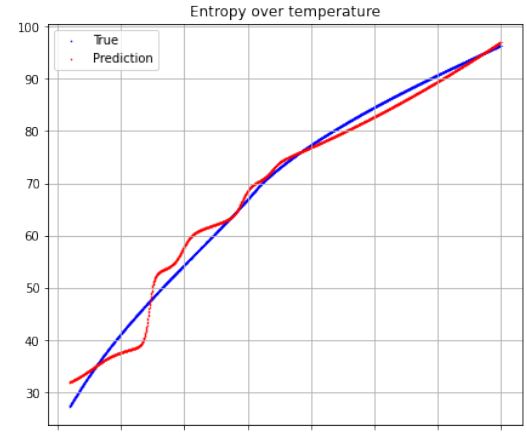
[6]:
```
[<matplotlib.lines.Line2D at 0x252d86aa130>]
```

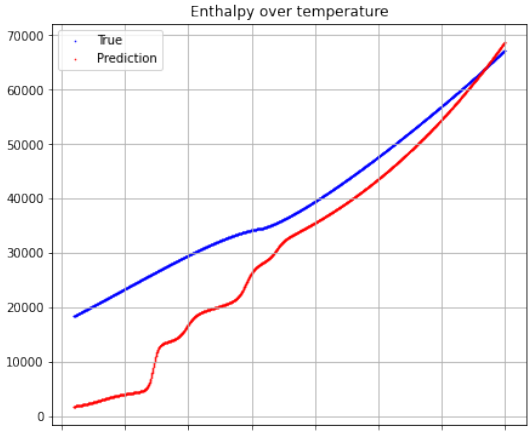


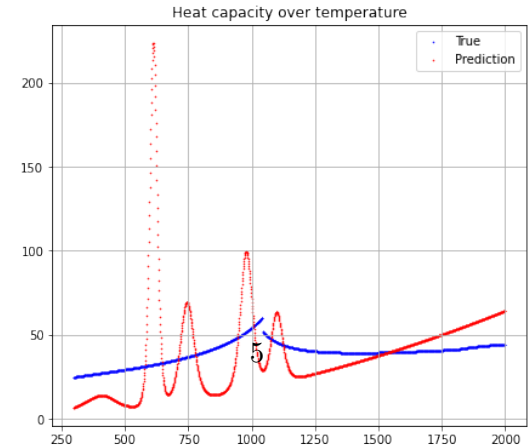
[7]:
```
ph = PlotHandler('Laenge')

ph.properties_temp(net, element, phase, scaling=False, start_temp=start_temp,␣
↪end_temp=end_temp)
```

```
Fe successfully selected!
```


Gibbs energy over temperature
True
Prediction
−20000
−40000
−60000
−80000
−100000
−120000


Entropy over temperature
True
Prediction
100
90
80
70
60
50
40
30


Enthalpy over temperature
True
Prediction
70000
60000
50000
40000
30000
20000
10000
0


Heat capacity over temperature
True
Prediction
200
150
100
50
0
250
500
750
1000
1250
1500
1750
2000

[8]:
```
#torch.save(net, 'Neural_Nets/LaengeNet/Models/model_07_02_22_1356')
```

[ ]:
```

```