train_LaengeNet_all

February 5, 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)

        # Input scaling
        #temp /= temp.max()

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

        # Output scaling
        #scale = 100000
        scale = 1
        gibbs_energy, entropy, enthalpy, heat_cap = gibbs_energy/scale, entropy/
↪scale, enthalpy/scale, heat_cap/scale
        g, s, h, c = g/scale, s/scale, h/scale, c/scale

        # 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()
        #torch.nn.utils.clip_grad_norm_(net.parameters(), 100)
        optimizer.step()
        epoch_losses[i] = loss

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

[17]:
```
def train(net, dataset):
    # Hyperparameters
    n_epochs = 100
    lr = 0.01
    batch_size = 128
    std_thresh = 0.05
    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=None)

    losses = []

    # Keep track of epoch where learning rate was reduced last
    lr_reduced_last = 0

    for i in range(n_epochs):
        print('-----\nEpoch %i:\n' % i)
        loss = epoch(net, dataloader, loss_func, optimizer)
        losses.append(loss)

        # Adapt learning rate if standard deviation over the last 10 epochs is␣
↪below a threshold
        if np.array(losses[-10:]).std() < std_thresh and (i - lr_reduced_last)␣
↪>= 10:
            print('Learning rate halfed! \n')
            lr_reduced_last = i
            lr /= 2
```

[18]:
```
net = LaengeNet(init_args=(-0.2, -0.1))

element = 'Fe'
phase = ['BCC_A2']
dataset = ThermoDataset(element, phase, scaling=False)

train(net, dataset)
```

```
Fe successfully selected!

-----
Epoch 0:

Mean epoch loss:  446430902.85714287
-----
Epoch 1:

Mean epoch loss:  151830509.7142857
-----
Epoch 2:

Mean epoch loss:  224236730.85714287
-----
Epoch 3:

Mean epoch loss:  248201387.42857143
-----
Epoch 4:

Mean epoch loss:  157879505.7142857
-----
Epoch 5:

Mean epoch loss:  193528710.2857143
-----
Epoch 6:

Mean epoch loss:  253607486.2857143
-----
Epoch 7:

Mean epoch loss:  143991017.14285713
-----
Epoch 8:

Mean epoch loss:  151755648.57142857
-----
Epoch 9:
```

```
Mean epoch loss:  202552853.14285713
-----
Epoch 10:

Mean epoch loss:  222032878.2857143
-----
Epoch 11:

Mean epoch loss:  145088085.7142857
-----
Epoch 12:

Mean epoch loss:  131331097.71428572
-----
Epoch 13:

Mean epoch loss:  97225667.14285715
-----
Epoch 14:

Mean epoch loss:  72244658.28571428
-----
Epoch 15:

Mean epoch loss:  34005273.64285714
-----
Epoch 16:

Mean epoch loss:  25026904.39285714
-----
Epoch 17:

Mean epoch loss:  nan
-----
Epoch 18:

Mean epoch loss:  nan
-----
Epoch 19:

Mean epoch loss:  nan
-----
Epoch 20:

Mean epoch loss:  nan
-----
Epoch 21:
```

```
Mean epoch loss:  nan
-----
Epoch 22:

Mean epoch loss:  nan
-----
Epoch 23:

Mean epoch loss:  nan
-----
Epoch 24:

Mean epoch loss:  nan
-----
Epoch 25:

Mean epoch loss:  nan
-----
Epoch 26:

Mean epoch loss:  nan
-----
Epoch 27:

Mean epoch loss:  nan
-----
Epoch 28:

Mean epoch loss:  nan
-----
Epoch 29:

Mean epoch loss:  nan
-----
Epoch 30:

Mean epoch loss:  nan
-----
Epoch 31:

Mean epoch loss:  nan
-----
Epoch 32:

Mean epoch loss:  nan
-----
Epoch 33:
```

```
Mean epoch loss:  nan
-----
Epoch 34:

Mean epoch loss:  nan
-----
Epoch 35:

Mean epoch loss:  nan
-----
Epoch 36:

Mean epoch loss:  nan
-----
Epoch 37:

Mean epoch loss:  nan
-----
Epoch 38:

Mean epoch loss:  nan
-----
Epoch 39:

Mean epoch loss:  nan
-----
Epoch 40:

Mean epoch loss:  nan
-----
Epoch 41:

Mean epoch loss:  nan
-----
Epoch 42:

Mean epoch loss:  nan
-----
Epoch 43:

Mean epoch loss:  nan
-----
Epoch 44:

Mean epoch loss:  nan
-----
Epoch 45:
```

```
Mean epoch loss:  nan
-----
Epoch 46:

Mean epoch loss:  nan
-----
Epoch 47:

Mean epoch loss:  nan
-----
Epoch 48:

Mean epoch loss:  nan
-----
Epoch 49:

Mean epoch loss:  nan
-----
Epoch 50:

Mean epoch loss:  nan
-----
Epoch 51:

Mean epoch loss:  nan
-----
Epoch 52:

Mean epoch loss:  nan
-----
Epoch 53:

Mean epoch loss:  nan
-----
Epoch 54:

Mean epoch loss:  nan
-----
Epoch 55:

Mean epoch loss:  nan
-----
Epoch 56:

Mean epoch loss:  nan
-----
Epoch 57:
```

```
Mean epoch loss:  nan
-----
Epoch 58:

Mean epoch loss:  nan
-----
Epoch 59:

Mean epoch loss:  nan
-----
Epoch 60:

Mean epoch loss:  nan
-----
Epoch 61:

Mean epoch loss:  nan
-----
Epoch 62:

Mean epoch loss:  nan
-----
Epoch 63:

Mean epoch loss:  nan
-----
Epoch 64:

Mean epoch loss:  nan
-----
Epoch 65:

Mean epoch loss:  nan
-----
Epoch 66:

Mean epoch loss:  nan
-----
Epoch 67:

Mean epoch loss:  nan
-----
Epoch 68:

Mean epoch loss:  nan
-----
Epoch 69:
```

```
Mean epoch loss:  nan
-----
Epoch 70:

Mean epoch loss:  nan
-----
Epoch 71:

Mean epoch loss:  nan
-----
Epoch 72:

Mean epoch loss:  nan
-----
Epoch 73:

Mean epoch loss:  nan
-----
Epoch 74:

Mean epoch loss:  nan
-----
Epoch 75:

Mean epoch loss:  nan
-----
Epoch 76:

Mean epoch loss:  nan
-----
Epoch 77:

Mean epoch loss:  nan
-----
Epoch 78:

Mean epoch loss:  nan
-----
Epoch 79:

Mean epoch loss:  nan
-----
Epoch 80:

Mean epoch loss:  nan
-----
Epoch 81:
```

```
Mean epoch loss:  nan
-----
Epoch 82:

Mean epoch loss:  nan
-----
Epoch 83:

Mean epoch loss:  nan
-----
Epoch 84:

Mean epoch loss:  nan
-----
Epoch 85:

Mean epoch loss:  nan
-----
Epoch 86:

Mean epoch loss:  nan
-----
Epoch 87:

Mean epoch loss:  nan
-----
Epoch 88:

Mean epoch loss:  nan
-----
Epoch 89:

Mean epoch loss:  nan
-----
Epoch 90:

Mean epoch loss:  nan
-----
Epoch 91:

Mean epoch loss:  nan
-----
Epoch 92:

Mean epoch loss:  nan
-----
Epoch 93:
```

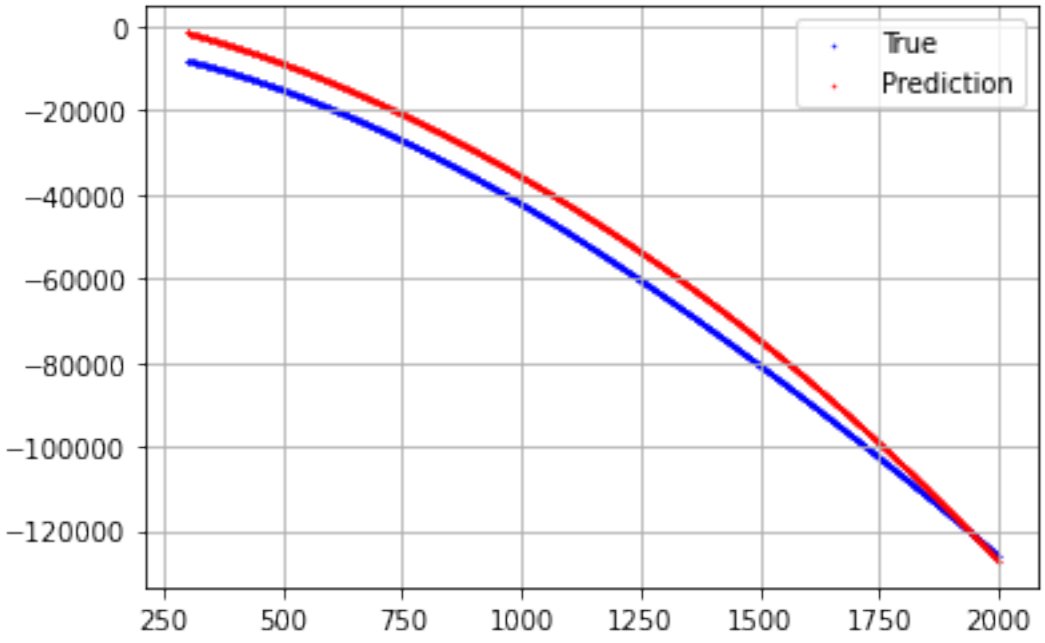
```
Mean epoch loss:  nan
-----
Epoch 94:

Mean epoch loss:  nan
-----
Epoch 95:

Mean epoch loss:  nan
-----
Epoch 96:

Mean epoch loss:  nan
-----
Epoch 97:

Mean epoch loss:  nan
-----
Epoch 98:

Mean epoch loss:  nan
-----
Epoch 99:

Mean epoch loss:  nan
```

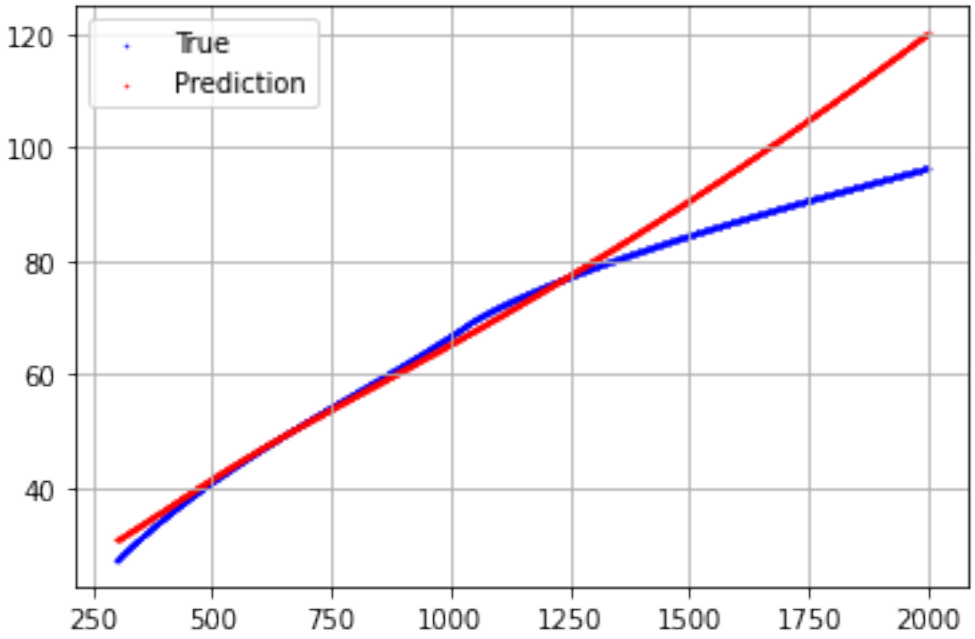
[19]:
```
print('theta_E: ', net.sub_net_1.act_1.theta_E)
print('E0: ', net.sub_net_1.act_1.E0)
print('a: ', net.sub_net_1.act_1.a)
print('b: ', net.sub_net_1.act_1.b)
```

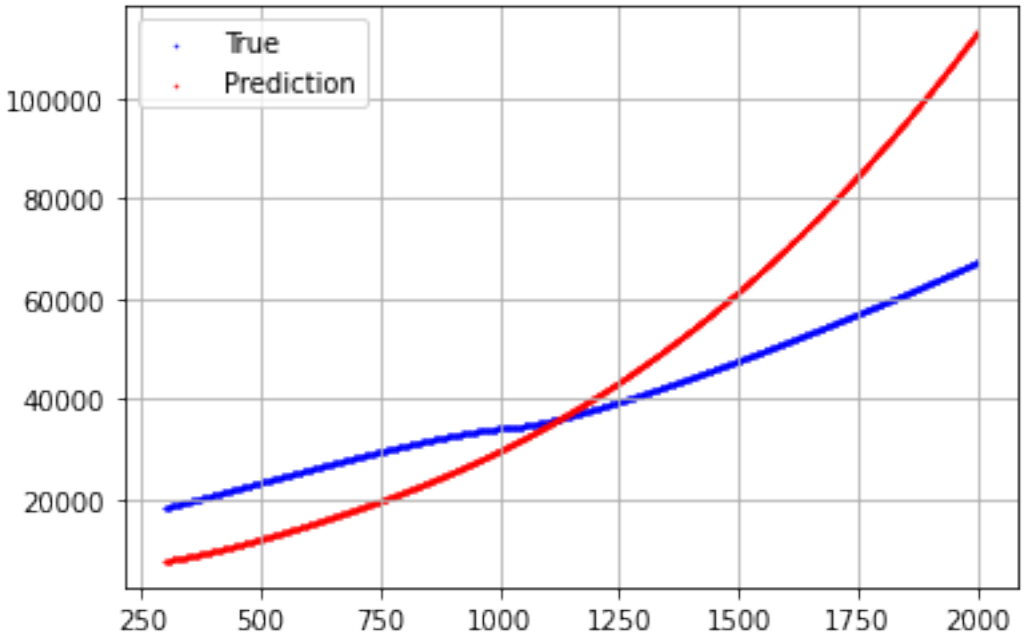
```
theta_E:  Parameter containing:
tensor(-1., requires_grad=True)
E0:  Parameter containing:
tensor(77104.9062, requires_grad=True)
a:  Parameter containing:
tensor(-37.4346, requires_grad=True)
b:  Parameter containing:
tensor(1.8301, requires_grad=True)
```

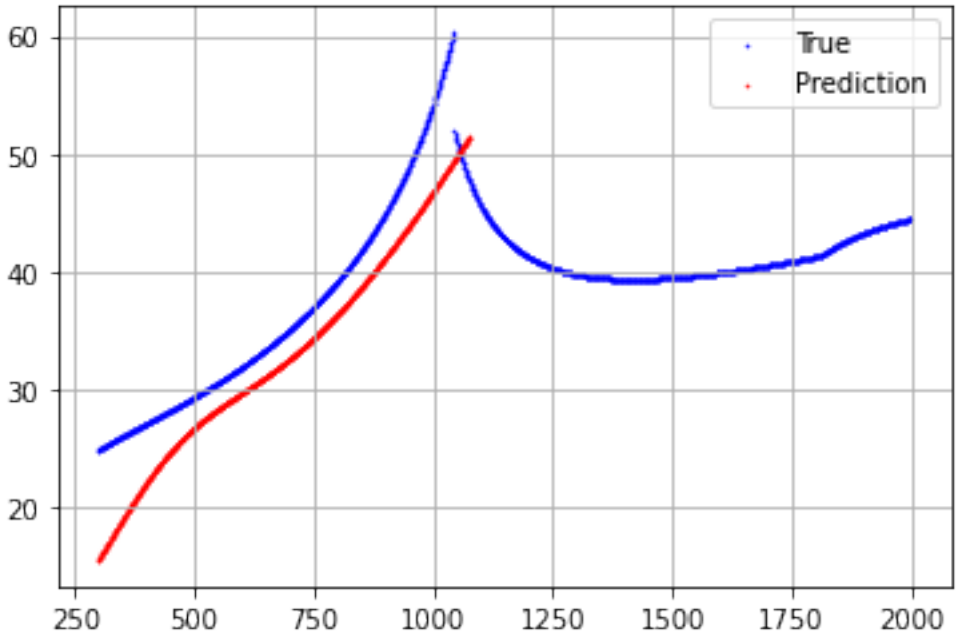
[20]:
```
ph = PlotHandler('Laenge')

ph.properties_temp(net, element, phase, scaling=False)
```

```
Fe successfully selected!
```


True
Prediction
0
−20000
−40000
−60000
−80000
−100000
−120000
250
500
750
1000
1250
1500
1750
2000


True
Prediction
120
100
80
60
40
250
500
750
1000
1250
1500
1750
2000





[7]:
```
#torch.save(net, 'LaengeNet/Models/model_12_01_22_1535')
```

[ ]:
```

```