train_PhaseClassifier

February 16, 2022

[1]:
```
from Neural_Nets.Classification.PhaseClassifier.Development.PhaseClassifier␣
↪import PhaseClassifier
from Neural_Nets.Classification.ClassificationDataset.Development.
↪ClassificationDataset import *
from torch.utils.data import DataLoader
import torch
import torch.nn as nn
from torch.optim import Adam
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
def epoch(net, train_loader, loss_func, optimizer, batch_size, seq_len):
    epoch_losses = np.zeros([len(train_loader), ])

    for i, d in enumerate(train_loader):
        # Scale inputs and get predictions
        inp = d[:, :, :-1].squeeze(1)/1000
        predictions = net(inp.float())

        # Each batch consists of measurement batches, where seq_len␣
↪measurements are put into one batch. In such a
        # measurement batch, every measurement has the same label as it needs␣
↪to be from the same element an phase.
        # This leads to a target array where the target class is contained for␣
↪each measurement in the measurement
        # batch. With this, CrossEntropyLoss would not work as the predictions␣
↪are made for the whole measurement
        # batch and CEL therefore expects only on class label per measurement␣
↪batch. Therefore, only the first
        # element of the last dimension of d is considered as target (all the␣
↪entries in the last dimension are the
        # same anyways so it could be any entry)
        targets = d[:, :, -1][:, 0].long()

        # Calculate the loss
        loss = loss_func(predictions, targets)
        epoch_losses[i] = loss
```

```
        # Backward step
        net.zero_grad()
        loss.backward()
        optimizer.step()

    return epoch_losses.mean()
```

[3]:
```
# Training and testing routines

def train(net, train_loader, batch_size, seq_len, save=True):
    # Hyperparameters
    nr_epochs = 1000
    lr = 0.0025

    loss_func = nn.CrossEntropyLoss()
    optimizer = Adam(net.parameters(), lr=lr)

    losses = np.zeros([nr_epochs, ])

    for i in range(nr_epochs):
        losses[i] = epoch(net, train_loader, loss_func, optimizer, batch_size,␣
 ↪seq_len)

def test(net, test_loader):
    correct = 0
    incorrect = 0

    for d in test_loader:
        inp = d[:, :, :-1].squeeze(1)/1000
        prediction = net(inp.float()).argmax()
        target = d[:, :, -1][:, 0].long()
        if prediction == target:
            correct += 1
        else:
            incorrect += 1

    accuracy = correct/(correct + incorrect)
    print('Test accuracy: ', accuracy)

    return accuracy
```

[4]:
```
# Hyperparameters
seq_len = 1
batch_size = 32

# Create the dataset
```

```
measurement = 'G'
element = 'Fe'
dc = DatasetCreator(r"C:
↪\Users\danie\Documents\Montanuni\Masterarbeit\4_Daten\Elements.xlsx",␣
↪elements=[element],
                    splits=(0.8, 0.2), validation=False, seq_len=seq_len,␣
↪measurement=measurement)
train_dataset, test_dataset, val_dataset = dc.get_datasets()

# Create the DataLoaders
train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=1, shuffle=True)
if val_dataset:
    val_loader = DataLoader(val_dataset)

# Create the network
t = True
net = PhaseClassifier(element=element, train=t, measurement=measurement)
```

[5]:
```
if t:
    train(net, train_loader, batch_size, seq_len)
else:
    test(net, test_loader)
```

[8]:
```
#torch.save(net, 'Neural_Nets/Classification/PhaseClassifier/Models/
↪model_14_02_22_1246')
```

[5]:
```
elements = pd.read_excel(r"C:
↪\Users\danie\Documents\Montanuni\Masterarbeit\4_Daten\Elements.xlsx",␣
↪sheet_name='Phases', index_col='Index')
measurements = ['G', 'S', 'H', 'C']

test_accuracies = pd.DataFrame(index=elements.columns.values,␣
↪columns=measurements)

# Hyperparameters
seq_len = 1
batch_size = 32
```

[ ]:
```
# Train all networks

for element in elements:
    print('----------')
    print(element)
    for measurement in measurements:
        print('*****')
        print(measurement)
```

```
        # Create the dataset
        dc = DatasetCreator(r"C:
↪\Users\danie\Documents\Montanuni\Masterarbeit\4_Daten\Elements.xlsx",␣
↪elements=[element],
                    splits=(0.8, 0.2), validation=False, seq_len=seq_len,␣
↪measurement=measurement)
        train_dataset, test_dataset, val_dataset = dc.get_datasets()

        # Create the DataLoaders
        train_loader = DataLoader(train_dataset, batch_size=batch_size,␣
↪shuffle=True)
        test_loader = DataLoader(test_dataset, batch_size=1, shuffle=True)
        if val_dataset:
            val_loader = DataLoader(val_dataset)

        # Create the network
        t = True
        net = PhaseClassifier(element=element, train=t, measurement=measurement)

        # Train the network
        train(net, train_loader, batch_size, seq_len)

        # Save the network
        path = 'Neural_Nets/Classification/PhaseClassifier/Models/' + element +␣
↪'_' + measurement
        torch.save(net, path)

        # Test the network
        #test_accuracy = test(net, test_loader)

        # Save the test results
        #test_accuracies[measurement][element]

test_accuracies.to_excel(r"C:
↪\Users\danie\Documents\Montanuni\Masterarbeit\5_Programmcodes\Neural_Nets\Classification\Ph
↪statistics.xlsx")
```

[ ]: