

Problem 2 (fitting a simple neural network)

Use the dataset card transdata.csv from the previous homework and maintain the same train-test split.

Fit a feedforward neural network with two ReLU layers using stochastic gradient descent (SGD). Follow [this tutorial](#). Experiment with the number of neurons per layer, the number of epochs, the learning rate for SGD, and the batch size for backpropagation. Report accuracy and F1 score on the test sample. Does your model perform better than a simple decision tree from the last homework?

0.2 Problem 2

Below I download the card dataset and split it into 50% for training and 50% for testing.

```
df =pd.read_csv("/Users/sandinatatu/Desktop/card_transdata-1-2.csv")
```

```
X=df.drop("fraud", axis=1).values
y=df["fraud"].values
```

```
[11]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .5,
↳ random_state = 10)
```

I then transform my training and testing datasets into torch object. After doing so, I create a function detailing the structure of my neural network (the number of neurons per hidden layer will be detailed by a variable I called “hidden_neurons”).

```
import torch
from torch import nn
from torch.utils.data import DataLoader
from torchvision import datasets
from torchvision.transforms import ToTensor
[12]: from torch.utils.data import DataLoader, TensorDataset
```

```
X_train_tensor = torch.tensor(X_train, dtype=torch.float32)
y_train_tensor = torch.tensor(y_train, dtype=torch.float32)
X_test_tensor = torch.tensor(X_test, dtype=torch.float32)
y_test_tensor = torch.tensor(y_test, dtype=torch.float32)
```

```
train_dataset = TensorDataset(X_train_tensor, y_train_tensor)
train_dataloader = DataLoader(train_dataset, batch_size = 64)
```

```

test_dataset = TensorDataset(X_test_tensor, y_test_tensor)
test_dataloader = DataLoader(test_dataset, batch_size=64)

class NeuralNetwork(nn.Module):
    def __init__(self, hn):
        super().__init__()
        self.flatten = nn.Flatten()
        self.linear_relu_stack = nn.Sequential(
            nn.Linear(7, hn),
            nn.ReLU(),
            nn.Linear(hn, hn),
            nn.ReLU(),
            nn.Linear(hn, 2),
        )

    def forward(self, x):
        x = self.flatten(x)
        logits = self.linear_relu_stack(x)
        return logits

```

I then use the `train_loop` and the `test_loop` from the PyTorch website. These loops will help with understanding the performance on the training data, as well as on the test data.

```

[13]: from sklearn.metrics import f1_score

def train_loop(dataloader, model, loss_fn, optimizer):
    size = len(dataloader.dataset)

    model.train()
    for batch, (X, y) in enumerate(dataloader):

        y=y.long()
        pred = model(X)
        loss = loss_fn(pred, y)

        loss.backward()
        optimizer.step()
        optimizer.zero_grad()

        if batch % 100 == 0:
            loss, current = loss.item(), batch * batch_size + len(X)

def test_loop(dataloader, model, loss_fn):

```

```

model.eval()
size = len(dataloader.dataset)
num_batches = len(dataloader)
test_loss, correct = 0, 0

f1s=[]

with torch.no_grad():
    for X, y in dataloader:
        y=y.long()
        pred = model(X)
        test_loss += loss_fn(pred, y).item()
        correct += (pred.argmax(1) == y).type(torch.float).sum().item()
        cf1=f1_score(y.cpu(), pred.argmax(1).cpu(), average='weighted')
        f1s.append(cf1)

af1= sum(f1s) / num_batches
test_loss /= num_batches
correct /= size
print(f"Test Error: \n Accuracy: {(100*correct):>0.1f}%, F1 Score: {af1}, \n
↪Avg loss: {test_loss:>8f} \n")

```

0.2.1 Model 1

For the first model, I will use 100 neurons per hidden layer. the learning rate will be 1e-3, the batch size will be 64, and I will have 5 epochs.

```

[14]: learning_rate = 1e-3
      batch_size = 64
      epochs = 5
      hn = 100

      model = NeuralNetwork(hn)

```

```

[15]: loss_fn = nn.CrossEntropyLoss()

```

```

[16]: optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)

```

```

[17]: loss_fn = nn.CrossEntropyLoss()
      optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)

      for t in range(epochs):
          print(f"Epoch {t+1}\n-----")

```

```
train_loop(train_dataloader, model, loss_fn, optimizer)
test_loop(test_dataloader, model, loss_fn)
print("Done!")
```

Epoch 1

Test Error:

Accuracy: 92.3%, F1 Score: 0.9160726763778203, Avg loss: 0.223735

Epoch 2

Test Error:

Accuracy: 92.9%, F1 Score: 0.923822153040678, Avg loss: 0.182242

Epoch 3

Test Error:

Accuracy: 93.5%, F1 Score: 0.9288238461945655, Avg loss: 0.160974

Epoch 4

Test Error:

Accuracy: 94.3%, F1 Score: 0.9385039195805155, Avg loss: 0.138886

Epoch 5

Test Error:

Accuracy: 95.3%, F1 Score: 0.950725192053029, Avg loss: 0.121787

Done!

0.2.2 Model 2

For the second model, I will use 50 neurons per hidden_layer, a batch_size of 100, and 50 epochs. I will leave the learning rate the same.

```
[18]: learning_rate = 1e-3
      batch_size = 100
      epochs = 50
      hn = 50

      model = NeuralNetwork(hn)
```

```
[19]: loss_fn = nn.CrossEntropyLoss()
```

```
[20]: optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)
```

```
[21]: loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)

for t in range(epochs):
    print(f"Epoch {t+1}\n-----")
    train_loop(train_dataloader, model, loss_fn, optimizer)
    test_loop(test_dataloader, model, loss_fn)
print("Done!")
```

Epoch 1

Test Error:

Accuracy: 92.3%, F1 Score: 0.9132063914546926, Avg loss: 0.200240

Epoch 2

Test Error:

Accuracy: 93.0%, F1 Score: 0.9226235391808396, Avg loss: 0.165299

Epoch 3

Test Error:

Accuracy: 93.5%, F1 Score: 0.9284519850762067, Avg loss: 0.148936

Epoch 4

Test Error:

Accuracy: 94.4%, F1 Score: 0.9390439548902482, Avg loss: 0.130812

Epoch 5

Test Error:

Accuracy: 95.2%, F1 Score: 0.949355832975581, Avg loss: 0.116275

Epoch 6

Test Error:

Accuracy: 95.6%, F1 Score: 0.9544338363229107, Avg loss: 0.104019

Epoch 7

Test Error:

Accuracy: 96.1%, F1 Score: 0.9591953420480193, Avg loss: 0.093551

Epoch 8

Test Error:

Accuracy: 96.4%, F1 Score: 0.9634182365142981, Avg loss: 0.085939

Epoch 9

Test Error:

Accuracy: 96.6%, F1 Score: 0.9652521850493413, Avg loss: 0.079909

Epoch 10

Test Error:

Accuracy: 96.8%, F1 Score: 0.9678637326844176, Avg loss: 0.076111

Epoch 11

Test Error:

Accuracy: 97.0%, F1 Score: 0.9706895906526415, Avg loss: 0.073264

Epoch 12

Test Error:

Accuracy: 97.2%, F1 Score: 0.9725246888522892, Avg loss: 0.070227

Epoch 13

Test Error:

Accuracy: 97.4%, F1 Score: 0.9748575221745271, Avg loss: 0.064727

Epoch 14

Test Error:

Accuracy: 97.5%, F1 Score: 0.9751878905518522, Avg loss: 0.063259

Epoch 15

Test Error:

Accuracy: 97.8%, F1 Score: 0.9787445076346779, Avg loss: 0.059120

Epoch 16

Test Error:

Accuracy: 97.9%, F1 Score: 0.9791231415936769, Avg loss: 0.057238

Epoch 17

Test Error:

Accuracy: 97.9%, F1 Score: 0.9793056718451277, Avg loss: 0.056256

Epoch 18

Test Error:

Accuracy: 97.9%, F1 Score: 0.9795504174867682, Avg loss: 0.056023

Epoch 19

Test Error:

Accuracy: 98.2%, F1 Score: 0.982315364964906, Avg loss: 0.051916

Epoch 20

Test Error:

Accuracy: 98.2%, F1 Score: 0.982187012946666, Avg loss: 0.053296

Epoch 21

Test Error:

Accuracy: 98.0%, F1 Score: 0.9805450132548946, Avg loss: 0.050344

Epoch 22

Test Error:

Accuracy: 98.3%, F1 Score: 0.9831015023112061, Avg loss: 0.045974

Epoch 23

Test Error:

Accuracy: 98.5%, F1 Score: 0.9851329637338113, Avg loss: 0.044682

Epoch 24

Test Error:

Accuracy: 98.3%, F1 Score: 0.983126386841809, Avg loss: 0.045271

Epoch 25

Test Error:

Accuracy: 98.5%, F1 Score: 0.9853138899429756, Avg loss: 0.042274

Epoch 26

Test Error:

Accuracy: 98.4%, F1 Score: 0.9843099309057519, Avg loss: 0.042933

Epoch 27

Test Error:

Accuracy: 98.4%, F1 Score: 0.9847836854087625, Avg loss: 0.044477

Epoch 28

Test Error:

Accuracy: 98.5%, F1 Score: 0.9859161609136025, Avg loss: 0.041434

Epoch 29

Test Error:

Accuracy: 98.5%, F1 Score: 0.9855231985343685, Avg loss: 0.040554

Epoch 30

Test Error:

Accuracy: 98.4%, F1 Score: 0.9849782514459564, Avg loss: 0.040986

Epoch 31

Test Error:

Accuracy: 98.9%, F1 Score: 0.9890392173414442, Avg loss: 0.040421

Epoch 32

Test Error:

Accuracy: 99.0%, F1 Score: 0.9894246893447735, Avg loss: 0.035417

Epoch 33

Test Error:

Accuracy: 98.9%, F1 Score: 0.989154000380651, Avg loss: 0.035672

Epoch 34

Test Error:

Accuracy: 98.9%, F1 Score: 0.9897297163755264, Avg loss: 0.035281

Epoch 35

Test Error:

Accuracy: 98.9%, F1 Score: 0.98969990894516, Avg loss: 0.034250

Epoch 36

Test Error:

Accuracy: 98.9%, F1 Score: 0.9896103473092636, Avg loss: 0.033170

Epoch 37

Test Error:

Accuracy: 99.1%, F1 Score: 0.99095657155999, Avg loss: 0.034581

Epoch 38

Test Error:

Accuracy: 99.3%, F1 Score: 0.9927376872614276, Avg loss: 0.030884

Epoch 39

Test Error:

Accuracy: 99.0%, F1 Score: 0.9901989634513203, Avg loss: 0.032929

Epoch 40

Test Error:

Accuracy: 98.9%, F1 Score: 0.9894375101485104, Avg loss: 0.033249

Epoch 41

Test Error:

Accuracy: 99.1%, F1 Score: 0.9913511412120248, Avg loss: 0.030650

Epoch 42

Test Error:

Accuracy: 99.3%, F1 Score: 0.9925677619474407, Avg loss: 0.027472

Epoch 43

Test Error:

Accuracy: 99.3%, F1 Score: 0.9925659823947043, Avg loss: 0.026487

Epoch 44

Test Error:

Accuracy: 99.3%, F1 Score: 0.9932099928835201, Avg loss: 0.026834

Epoch 45

Test Error:

Accuracy: 99.4%, F1 Score: 0.994034604599448, Avg loss: 0.025856

Epoch 46

Test Error:

Accuracy: 99.4%, F1 Score: 0.9934338583376484, Avg loss: 0.024943

Epoch 47

Test Error:

Accuracy: 99.5%, F1 Score: 0.994762596644231, Avg loss: 0.024202

Epoch 48

Test Error:

Accuracy: 99.4%, F1 Score: 0.9934740447645186, Avg loss: 0.024598

Epoch 49

Test Error:

Accuracy: 99.3%, F1 Score: 0.9934613049869524, Avg loss: 0.026757

Epoch 50

Test Error:

Accuracy: 99.4%, F1 Score: 0.994350325256214, Avg loss: 0.023354

Done!

0.2.3 Model 3

For my final model, I will use 100 neurons per hidden layer, a batch_size of 64, 10 epochs, and a learning rate of 1e-4.

```
[22]: learning_rate = 1e-4
      batch_size = 64
      epochs = 10
      hn = 100

      model = NeuralNetwork(hn)
```

```
[23]: loss_fn = nn.CrossEntropyLoss()
```

```
[24]: optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)
```

```
[25]: loss_fn = nn.CrossEntropyLoss()
      optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)

      for t in range(epochs):
          print(f"Epoch {t+1}\n-----")
          train_loop(train_dataloader, model, loss_fn, optimizer)
          test_loop(test_dataloader, model, loss_fn)
      print("Done!")
```

Epoch 1

Test Error:

Accuracy: 91.5%, F1 Score: 0.882261473644765, Avg loss: 0.387805

Epoch 2

Test Error:
Accuracy: 91.7%, F1 Score: 0.8921798371616224, Avg loss: 0.292035

Epoch 3

Test Error:
Accuracy: 91.9%, F1 Score: 0.8984379999682679, Avg loss: 0.243960

Epoch 4

Test Error:
Accuracy: 92.0%, F1 Score: 0.9029252995191686, Avg loss: 0.221302

Epoch 5

Test Error:
Accuracy: 92.2%, F1 Score: 0.9064863137793205, Avg loss: 0.208320

Epoch 6

Test Error:
Accuracy: 92.3%, F1 Score: 0.908930732209424, Avg loss: 0.200694

Epoch 7

Test Error:
Accuracy: 92.4%, F1 Score: 0.9106543366623544, Avg loss: 0.194982

Epoch 8

Test Error:
Accuracy: 92.5%, F1 Score: 0.9121233702504971, Avg loss: 0.188377

Epoch 9

Test Error:
Accuracy: 92.6%, F1 Score: 0.9138915418022129, Avg loss: 0.184001

Epoch 10

Test Error:
Accuracy: 92.7%, F1 Score: 0.9154081861548743, Avg loss: 0.180168

Done!

The highest F1 score is obtained for the second model and is around 99.4. The highest accuracy is also obtained for the second model and is around 99.4. None of the models perform better than the decision tree from last week's homeworks.