ニューラルネットワーク実習05

浅川伸一

ニューラルネットワーク実習

```
import torch
import torch.nn as nn

class perceptron(nn.Module):
    def __init__(self, in_features=2, out_features=1):
        super().__init__()
        self.in_featuers = in_features
        self.out_features = out_features
        self.layer = nn.Linear(self.in_featuers, self.out_features)
        self.act_f = nn.Sigmoid()

    def forward(self, data):
        out = self.act_f(self.layer(data))
        return out
```

最適化と損失の定義

```
network = perceptron() # モデルの実体化（インスタンス化）
loss_f = nn.MSELoss() # 損失関数の定義

import torch.optim as optim
optimizer = optim.SGD(net.parameters(), lr=0.001) # 最適化の定義
```

データの定義

```
X = torch.Tensor([[0,0],[0,1],[1,0],[1,1]])
y = torch.Tensor([[0],[1],[1],[1]])
print(X,y)
```

上記は，論理和になります。下記のようすると論理積になります。

```
y = torch.Tensor([[0],[0],[0],[1]])
```

学習の実行

```
net = perceptron()
loss_f = nn.MSELoss()
optimizer = optim.SGD(net.parameters(), lr=0.001)

net.train()
iter_max = 10 ** 3
interval = iter_max >> 2
for i in range(iter_max):

    pred = net(X)
    loss = loss_f(pred, y)

    if (i % (iter_max >> 2)) == 0:
        print(f'{i:<5d} 損失: {loss.detach().numpy():.3f}')

    loss.backward()

    # Updating gradients
    optimizer.step()
```

まとめ

- PyTorch を用いた，簡単なニューラルネットワークの実習をしました

実習

学習で用いたファイルを変更して遊んでみてください