hw3

October 1, 2021

1 Mathematical Foundations of Deep Neural Networks

1.1 Homework 3

1.1.1 2017-11362

1.2 Problem 1: *3-layer MLP to fit a univariate function.*

[1]:
```
import torch
import numpy as np
from torch import nn, optim
from torch.nn import functional as F
from torch.utils.data import TensorDataset, DataLoader
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

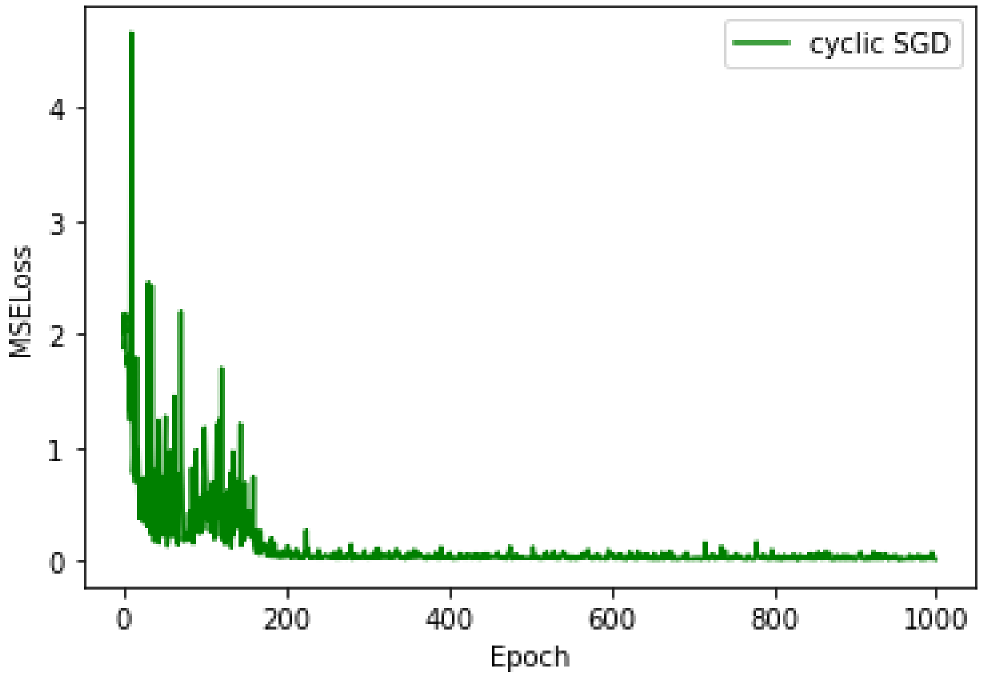
[2]:
```
alpha = 0.1
K = 1000
B = 128
N = 512

def f_true(x) :
    return (x-2) * np.cos(x*4)

np.random.seed(0)
X_train = torch.tensor(np.random.normal(loc = 0.0, scale = 1.0, size = N),␣
 ↪dtype=torch.float32)
y_train = f_true(X_train)
X_val = torch.tensor(np.random.normal(loc = 0.0, scale = 1.0, size = N//5),␣
 ↪dtype=torch.float32)
y_val = f_true(X_val)

train_dataloader = DataLoader(TensorDataset(X_train.unsqueeze(1), y_train.
 ↪unsqueeze(1)), batch_size=B, shuffle=True)
test_dataloader = DataLoader(TensorDataset(X_val.unsqueeze(1), y_val.
 ↪unsqueeze(1)), batch_size=B)
```

[3]:
```python
# Define model
class MLP(nn.Module):
    def __init__(self):
        super().__init__()
        self.linear1 = nn.Linear(1, 64, bias=True)
        self.linear2 = nn.Linear(64, 64, bias=True)
        self.linear3 = nn.Linear(64, 1, bias=True)

    def forward(self, x):
        x = torch.sigmoid(self.linear1(x))
        x = torch.sigmoid(self.linear2(x))
        x = self.linear3(x)
        return x
```

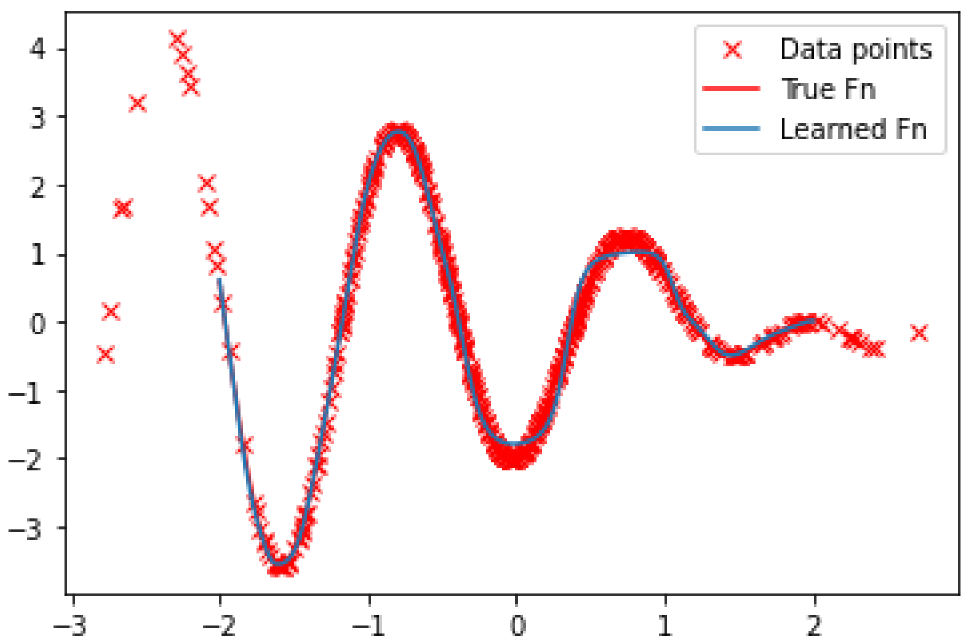
[4]:
```python
# Model construction
model = MLP()
loss_function = nn.MSELoss()
optimizer = torch.optim.SGD(model.parameters(), lr=alpha)

# Initialization
for layer in [model.linear1, model.linear2, model.linear3]:
    layer.weight.data = torch.normal(0, 1, layer.weight.shape)
    layer.bias.data = torch.full(layer.bias.shape, 0.03)
```

[5]:
```python
# Training
f_val = []

for epoch in range(K):
    for _ in range(N // B):
        for x_data, y_data in train_dataloader:
            # forward
            y_pred = model(x_data)
            # loss
            loss = loss_function(y_pred, y_data.float())

            # backward
            loss.backward()

            # update
            optimizer.step()
            optimizer.zero_grad()

    # save loss
    f_val.append(loss)

plt.plot(list(range(K)), f_val, color='green', label="cyclic SGD")
plt.xlabel('Epoch')
```

```
plt.ylabel('MSELoss')
plt.legend()
plt.show()
```



[6]:
```
# Plot
with torch.no_grad():
    xx = torch.linspace(-2,2,1024).unsqueeze(1)
    plt.plot(X_train,y_train,'rx',label='Data points')
    plt.plot(xx,f_true(xx),'r',label='True Fn')
    plt.plot(xx, model(xx),label='Learned Fn')
plt.legend()
plt.show()
```



1.3 Problem 2: *Deep learning operates under* $p \gg N$.

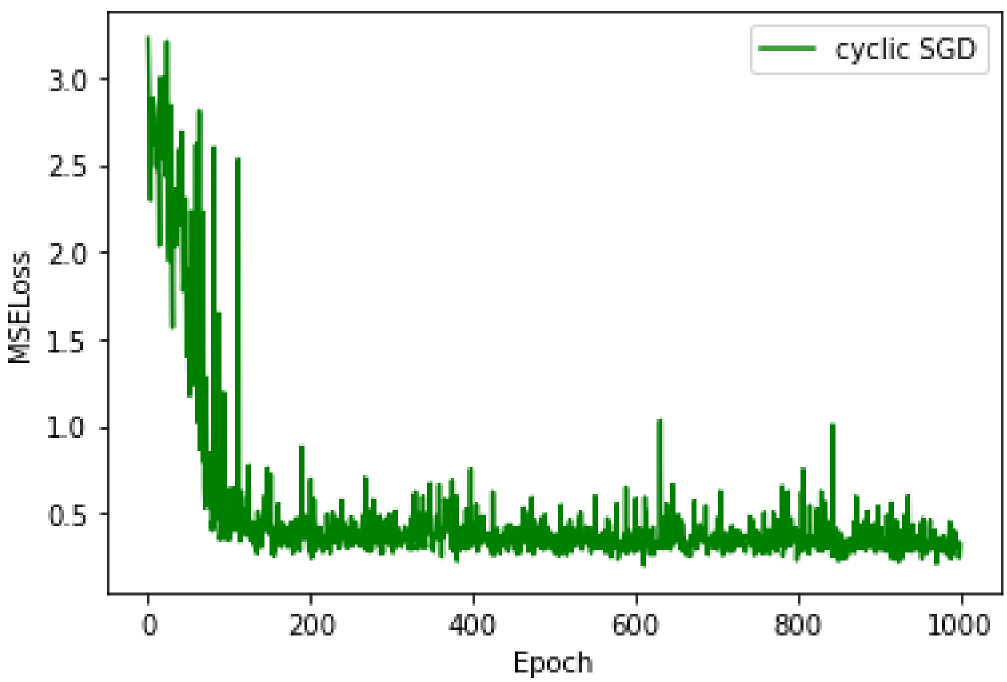
In previous problem, $p = 1 \times 64 + 64 + 64 \times 64 + 64 + 64 \times 1 + 1 = 4353$, $N = 512$.

[8]:
```
# Generate data-set
np.random.seed(0)
X_train = torch.tensor(np.random.normal(loc = 0.0, scale = 1.0, size = N),␣
↪dtype=torch.float32)
y_train = f_true(X_train) + torch.normal(0, 0.5, X_train.shape)
train_dataloader = DataLoader(TensorDataset(X_train.unsqueeze(1), y_train.
↪unsqueeze(1)), batch_size=B, shuffle=True)
```

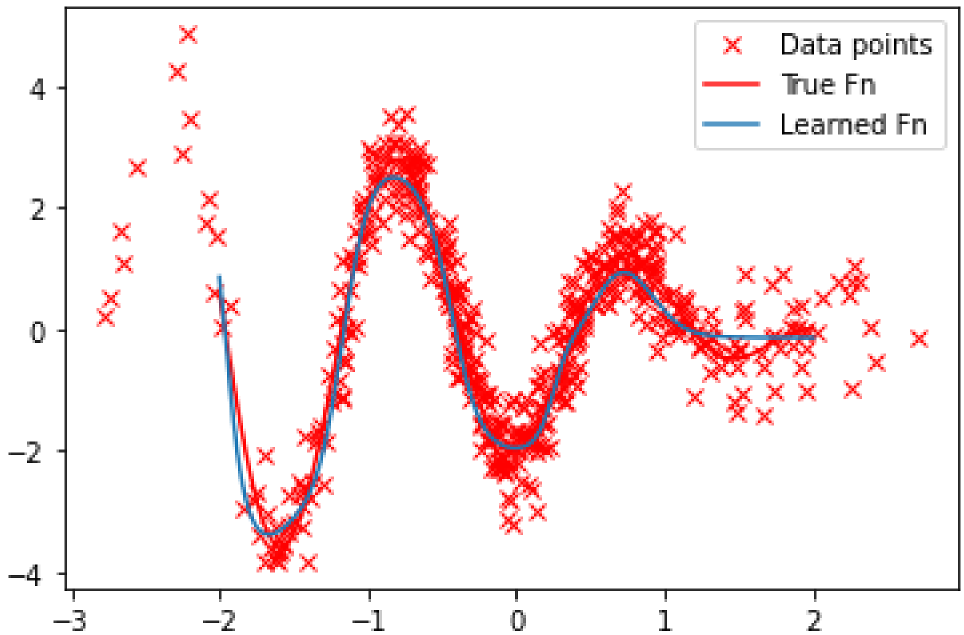
[9]:
```
# Model construction
model = MLP()
loss_function = nn.MSELoss()
optimizer = torch.optim.SGD(model.parameters(), lr=alpha)

# Initialization
for layer in [model.linear1, model.linear2, model.linear3]:
    layer.weight.data = torch.normal(0, 1, layer.weight.shape)
    layer.bias.data = torch.full(layer.bias.shape, 0.03)
```

[10]:
```
# Training
f_val = []

for epoch in range(K):
```

```
    for _ in range(N // B):
        for x_data, y_data in train_dataloader:
            # forward
            y_pred = model(x_data)
            # loss
            loss = loss_function(y_pred, y_data.float())

            # backward
            loss.backward()

            # update
            optimizer.step()
            optimizer.zero_grad()

    # save loss
    f_val.append(loss)

plt.plot(list(range(K)), f_val, color='green', label="cyclic SGD")
plt.xlabel('Epoch')
plt.ylabel('MSELoss')
plt.legend()
plt.show()
```



[11]:
```
# Plot
with torch.no_grad():
    xx = torch.linspace(-2,2,1024).unsqueeze(1)
    plt.plot(X_train,y_train,'rx',label='Data points')
    plt.plot(xx,f_true(xx),'r',label='True Fn')
    plt.plot(xx, model(xx),label='Learned Fn')
plt.legend()
plt.show()
```



1.4 Problem 3: *Basic properties of the CE loss.*

$\ell^{CE}(f, y) = -\log\left(\frac{\exp f_y}{\sum \exp(f_j)}\right)$, $f \in \mathbb{R}^k$, $y \in \{1, \cdots, k\}$.

(a) $0 < \ell^{CE}(f, y) < \infty$.

$$0 < \ell^{CE}(f, y) \iff 0 > \log\left(\frac{\exp(f_y)}{\sum \exp(f_j)}\right)$$

$$\iff 1 > \frac{\exp(f_y)}{\sum \exp(f_j)}$$

$$\iff \underline{\sum_{j=1}^{k} \exp(f_j)} > \exp(f_y)$$

$$\exp(f_j) > 0 \Rightarrow \sum_{j \neq y} \exp(f_j) > 0.$$

$$\ell^{CE}(f, y) < \infty \iff \log\left(\frac{\exp(f_y)}{\sum \exp(f_j)}\right) > -\infty$$

$$\iff \frac{\exp(f_y)}{\sum \exp(f_j)} > 0$$

$$\iff \exp(f_y) > 0.$$

(b) $\lambda e_y = (0, 0, \cdots, 0, \underset{\substack{\uparrow \\ y\text{th}}}{\lambda}, 0, \cdots, 0)^t$

$$\sum_{j=1}^{k} \exp(j\text{th element of } \lambda e_y) = \sum_{j \neq y} 1 + e^{\lambda} = e^{\lambda} + k - 1$$

$$\ell^{CE}(\lambda e_y, y) = -\log\left(\frac{e^{\lambda}}{e^{\lambda} + k - 1}\right)$$

$$\lim_{\lambda \to \infty} \ell^{CE}(\lambda e_y, y) = -\log\left(\lim_{\lambda \to \infty} \frac{1}{1 + e^{-\lambda}(k-1)}\right) = -\log 1 = 0.$$

1.5 Problem 4: *Derivative of max.*

Assume that $f'(x) \neq f_I'(x)$ for some x.

$f_I'(x) = \lim_{h\to 0} \frac{1}{h} (f_I(x+h) - f_I(x))$

$f'(x) = \lim_{h\to 0} \frac{1}{h} (\max\{f_1(x+h), \cdots, f_k(x+h)\} - f_I(x))$

$f_I'(x) \neq f'(x)$ means that $\lim_{h\to 0} \max\{f_1(x+h), \cdots, f_k(x+h)\} \neq f_I(x)$.

That is, $\exists J$ s.t $f_J(x+h) > f_I(x+h)$ for $0<|h|<\delta$ for some $\delta > 0$. $\cdots (*)$

Since $I = \underset{1\le i\le k}{\operatorname{argmax}} f_i(x)$ is unique, $f_I(x) > f_J(x)$.

Let $f_I(x) - f_J(x) = \varepsilon > 0$.

Then $\exists \delta > 0$ s.t $|h| < \delta \Rightarrow |f_I(x+h) - f_I(x)| < \frac{\varepsilon}{3}$, $|f_J(x+h) - f_I(x)| < \frac{\varepsilon}{3}$. ($\because f_I, f_J$ is diff $\Rightarrow$ cont.)

$f_I(x) - \frac{\varepsilon}{3} < f_I(x+h) < f_I(x) + \frac{\varepsilon}{3}$, $f_J(x) - \frac{\varepsilon}{3} < f_J(x+h) < f_J(x) + \frac{\varepsilon}{3}$

$\Rightarrow f_I(x+h) - f_J(x+h) > f_I(x) - \frac{\varepsilon}{3} - \left(f_J(x) + \frac{\varepsilon}{3}\right)$

$> \varepsilon - \frac{2}{3}\varepsilon = \frac{\varepsilon}{3} > 0$, which is contradict to $(*)$.

1.6 Problem 5: *Basic properties of activation functions.*

(a) $\sigma(z) = \max\{0, z\} = \begin{cases} 0, & z < 0 \\ z, & z \ge 0 \end{cases}$

$\sigma(\sigma(z)) = \max\{0, \sigma(z)\} = \begin{cases} 0, & \sigma(z) < 0 \\ \sigma(z), & \sigma(z) \ge 0 \end{cases} = \begin{cases} 0, & z < 0 \\ z, & z \ge 0 \end{cases} = \sigma(z)$.

(b) Lipschitz continuity : $\exists L$ s.t $|f(x) - f(y)| \le L|x-y| \Leftrightarrow \exists L$ s.t $|f'(x)| \le L$.

- $\sigma'(z) = \frac{e^z}{1+e^z} = \frac{1}{1+e^{-z}}$, $\sigma''(z) = \frac{e^{-z}}{(1+e^{-z})^2} < \frac{e^{-z}}{1+e^{-z}} < 1$.
- $\frac{d}{dz} \mathrm{ReLU}(z) = \begin{cases} 0, & z < 0 \\ 1, & z > 0 \end{cases}$

Let $x = -\varepsilon$, $y = \varepsilon$ $(\varepsilon > 0)$.

$|\mathrm{ReLU}'(x) - \mathrm{ReLU}'(y)| = 1$, but $|x-y| = 2\varepsilon$. (as $\varepsilon \to 0$, $L \to \infty$: not L-cont).

(c) $\rho(z) = \frac{1-e^{-2z}}{1+e^{-2z}} = \frac{2}{1+e^{-2z}} - 1 = 2\sigma(2z) - 1$

$\Rightarrow \rho(z)$ is scaled-version of $\sigma(z)$

$\therefore$ Result of MLP is equivalent.

1.7 Problem 6: *Vanishing gradients.*

We only have to prove that $a_j^1 X_i + b_j^1 < 0$ for all $i = 1, \cdots, N$ ($\because$ induction).

$\nabla \ell(f_\theta(X_i), Y_i) = \nabla f_\theta(X_i) \cdot \ell(f_\theta(X_i), Y_i)$

$a_j^1 = a_j^0 - \alpha \cdot \frac{\partial}{\partial a_j} u^T \sigma(a^0 X_i + b^0)$

$b_j^1 = b_j^0 - \alpha \frac{\partial}{\partial b_j} u^T \sigma(a^0 X_i + b^0)$.

$a_j^0 X_i + b_j^0 < 0 \Rightarrow \begin{cases} \frac{\partial}{\partial a_j} u_j \sigma(a_j^0 X_i + b_j^0) = 0 \\ \frac{\partial}{\partial b_j} u_j \sigma(a_j^0 X_i + b_j^0) = 0 \end{cases}$.

$\therefore a_j^1 = a_j^0, \quad b_j^1 = b_j^0$

$\therefore a_j^1 X_i + b_j^1 < 0$ for $\forall i = 1, \cdots, N$.

1.8 Problem 7: *Leaky ReLU.*

Assume that $a_j^0 X_i + b_j^0 < 0$.

$\frac{\partial}{\partial a_j} u_j \sigma(a_j X_i + b_j) = u_j X_i \alpha$

$\frac{\partial}{\partial b_j} u_j \sigma(a_j X_i + b_j) = u_j \alpha$

$\Rightarrow a_j^1 \neq a_j^0, \; b_j^1 \neq b_j^0$, which means $a_j^1 X_i + b_j^1$ differ from $a_j^0 X_i + b_j^0 < 0$.

$\Rightarrow$ There is a chance that $a_j^1 X_i + b_j^1 > 0$: not vanished gradients