令和 2 年 11 月 19 日 (木) 伊藤　薫

岡田先生情報工学実験課題

1 XOR モデルのソースコードと学習率を変えた挙動の変化とそれぞれの勾配の最大値

Listing 1 に用いたソースコードを示す。図 1 図 6 に学習率 lr を 100,10,1,0.1,0.01,0.001 としたときの XOR モデルの挙動の変化とそれぞれの勾配の最大値を示す。

Listing 1: XOR モデルのソースコード

```
import torch
import numpy as np
import torch.nn as nn
import matplotlib.pyplot as plt

class MLP(nn.Module):
  def __init__(self,input_size,hidden_size,output_size):
    super(MLP,self).__init__()

    self.l1=nn.Linear(input_size,hidden_size)
    self.l2=nn.Linear(hidden_size,output_size)

  def forward(self,x):
    h=torch.sigmoid(self.l1(x))
    o=self.l2(h)
    return o

x=torch.tensor([[0.,0.],[0.,1.],[1.,0.],[1.,1.]])
y=torch.tensor([[0.],[1.],[1.],[0.]])

lr=100
max_epoch=50000

for i in range(6):
  history=[]
  history_l1=[]
  history_l2=[]
  model=MLP(2,3,1)
  params=list(model.parameters())
  optimizer = torch.optim.SGD(model.parameters(), lr=lr)
  criterion=nn.MSELoss()

  for j in range(max_epoch):
    pred = model(x)
    error = criterion(pred, y)
    history.append(error.item())
    optimizer.zero_grad()
    error.backward()
    history_l1.append(torch.max(params[0].grad).data)
    history_l2.append(torch.max(params[2].grad).data)
```

```
    optimizer.step()

epochs=np.arange(1,max_epoch+1)
history=np.array(history,dtype=np.float32)
history_l1=np.array(history_l1,dtype=np.float32)
history_l2=np.array(history_l2,dtype=np.float32)
epochs=np.arange(1,max_epoch+1)

fig,axes=plt.subplots(1,3,figsize=(15,5))

axes[0].plot(epochs,history)
axes[0].set_title("lr={}".format(lr))
axes[0].set_xlabel("epochs")
axes[0].set_ylabel("loss")
axes[1].plot(epochs,history_l1)
axes[1].set_title("max grad of l1")
axes[1].set_xlabel("epochs")
axes[2].plot(epochs,history_l2)
axes[2].set_title("max grad of l2")
axes[2].set_xlabel("epochs")
plt.show()
lr=lr/10
```

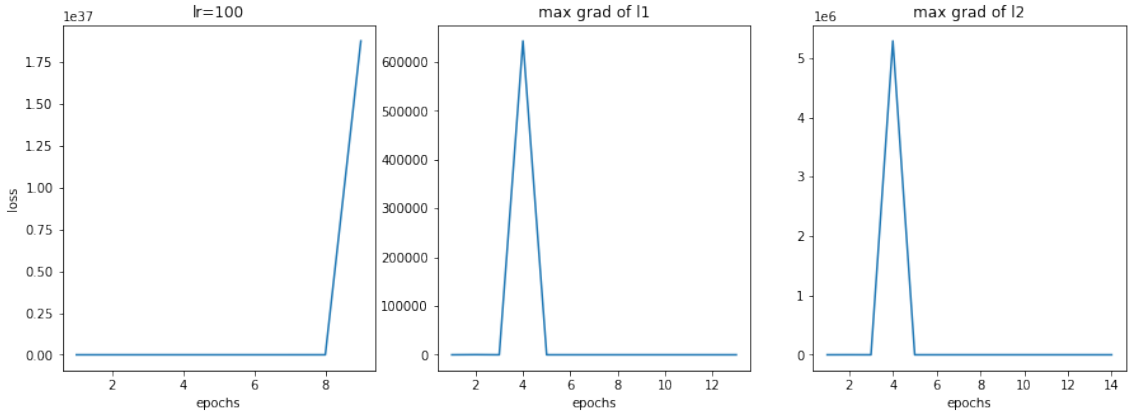

図 1:lr=100

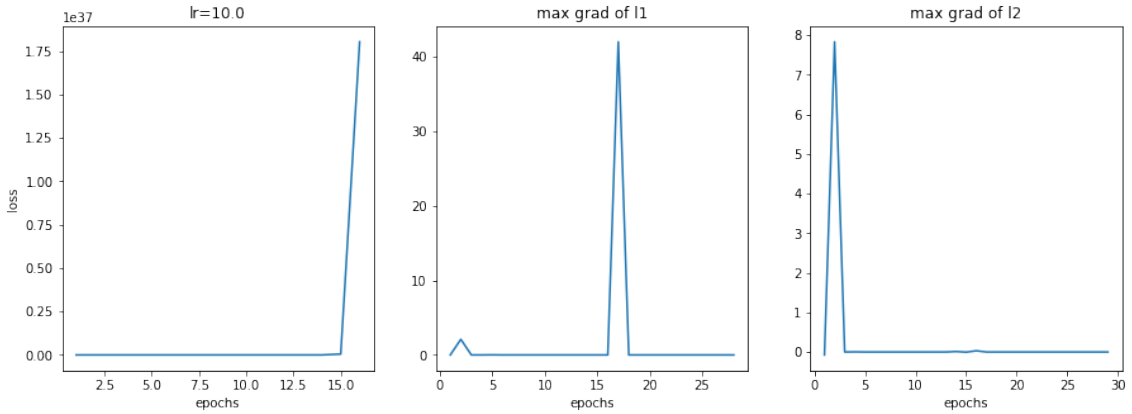

図 2:lr=10

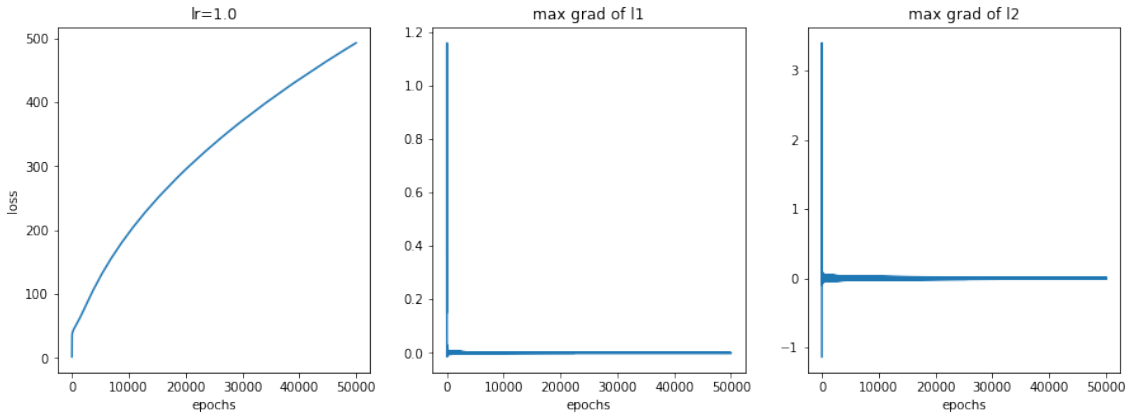

図 3:lr=1

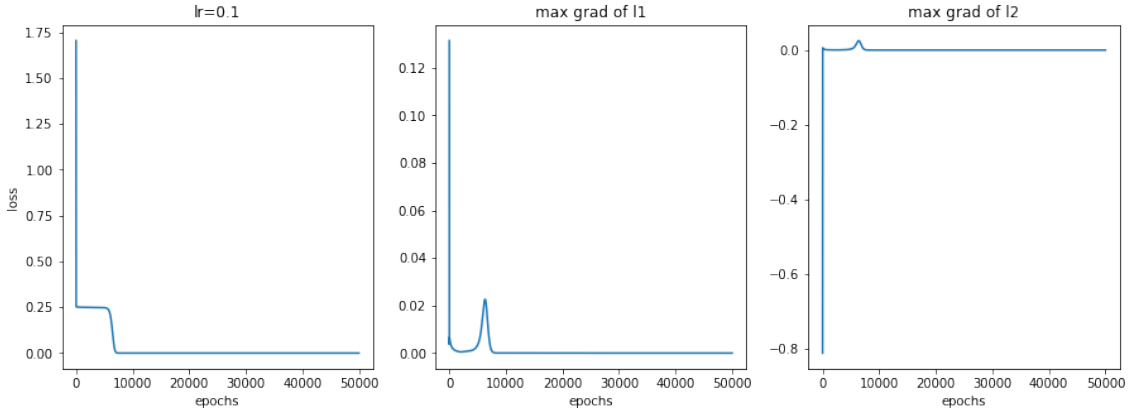

図 4:lr=0.1

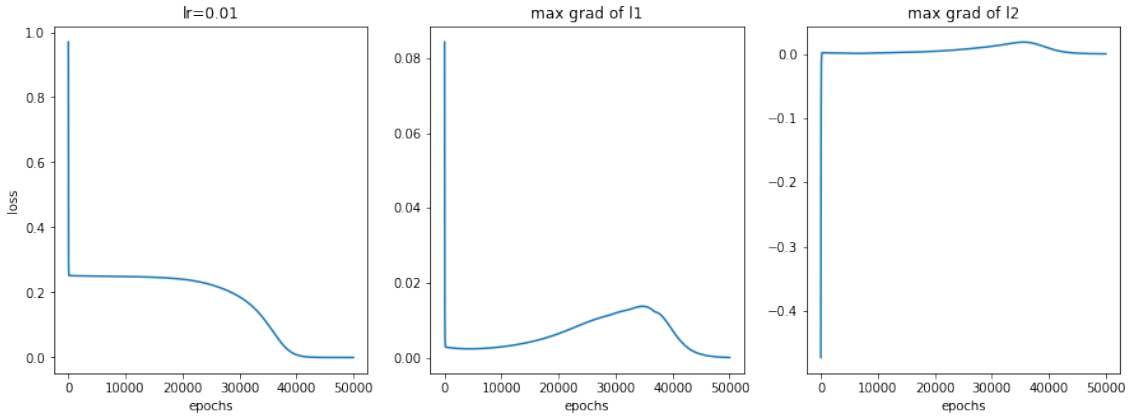

図 5:lr=0.01

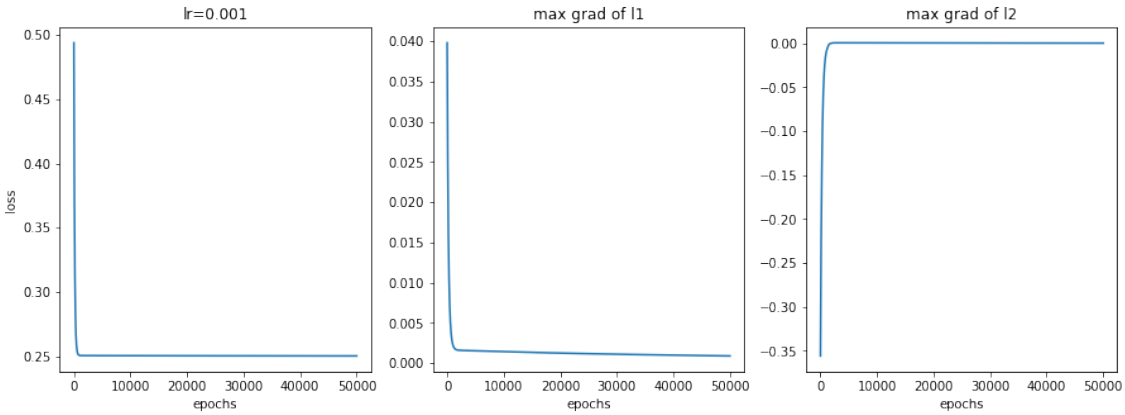

図 6:lr=0.001

2　交差エントロピーを誤差関数としたパリティビット予測

Listing 2 に用いたソースコードを示す。図 7 に Listing 2 を実行した結果を示す。

Listing 2: パリティビット予測モデルのソースコード

```
import torch
import numpy as np
import torch.nn as nn
import matplotlib.pyplot as plt

class MLP(nn.Module):
  def __init__(self, input_size, hidden_size, output_size):
    super(MLP, self).__init__()
    self.l1 = nn.Linear(input_size, hidden_size)
    self.l2 = nn.Linear(hidden_size, output_size)

  def forward(self, x):
    h = torch.sigmoid(self.l1(x))
    return torch.sigmoid(self.l2(h))

x = torch.tensor([[0, 0, 0], [0, 0, 1], [0, 1, 0], [0, 1, 1], [1, 0, 0], [1, 0, 1], [1, 1,
    0], [1, 1, 1]], dtype=torch.float)
y = torch.tensor([0,1,1,0,1,0,0,1])

model = MLP(3, 3, 2)
optimizer = torch.optim.SGD(model.parameters(), lr=1.0e-1)
criterion = nn.CrossEntropyLoss()

history = []
max_epoch = 100000

for epoch in range(max_epoch):
  pred = model(x)
  error = criterion(pred,y)
  history.append(error.item())
  model.zero_grad()
  error.backward()
  optimizer.step()
```

```
33
34 history = np.array(history, dtype=np.float32)
35 epochs = np.arange(1, max_epoch+1)
36 plt.plot(epochs,history,label="CrossEntropyLoss")
37 plt.xlabel("epochs")
38 plt.ylabel("loss")
39 plt.show()
```

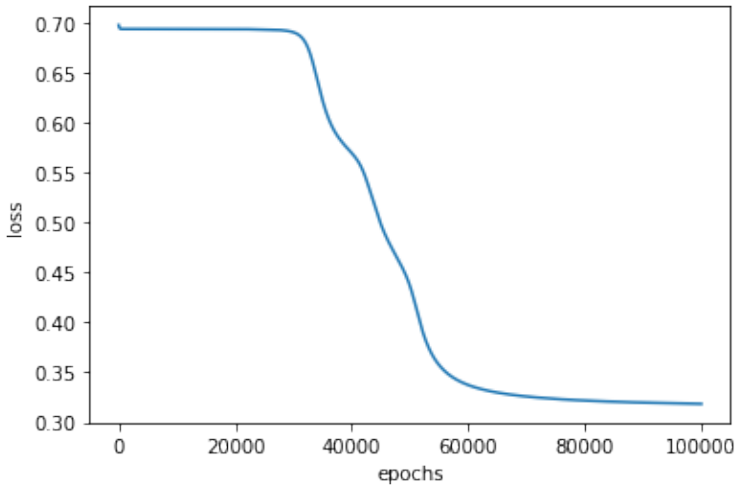

図 7:パリティビット予測モデルの誤差