PyTorch 教學

1. 引入+檢查版本

```
import torch
print(torch.__version__)
```

2.常用數值型態 type

```
# dtype = torch.int32 or torch.int
# dtype = torch.float64 or torch.float
```

3. 矩陣/張量宣告列印

```
# 3 x 3 Matrix with Python List
torch.tensor([[1.,2.,3.],[4.,5.,6.],[7.,8.,9.]])

import numpy as np
# 3 x 3 Matrix with Python Numpy
torch.tensor(np.array([[1.,2.,3.],[4.,5.,6.],[7.,8.,9.]]))
# if A is numpy array
torch.from_numpy(A)

# 初始化 4x10 實零矩陣,整零矩陣
torch.zeros([4,10],dtype=torch.int)
torch.zeros([4,10],dtype=torch.float)

# 初始化 5x9 實1矩陣,整1矩陣
torch.ones([5,9],dtype=torch.int)
torch.ones([5,9],dtype=torch.float)

# 初始化 2x3 實隨機矩陣 Normal(0,1)
torch.randn(2,3)
```

4. 架設網路 y = Wx + b

```
class MyModel(nn.Module):
  def __init__(self):
    super(MyModel, self).__init__()
    # 架構線性 Layer
        self.func1 = nn.Linear(_dimIn,_dimfunc1,False)
        # 初始化權重矩陣
        self.func1.weight = torch.nn.Parameter() #...torch.tensor
        # 初始化b 向量
        self.func1.bias = torch.nn.Parameter()#...torch.tensor
  # 最後回傳出 y_pred (tensor)
    def forward(self, x):
    y_pred = self.fc1(x)
    return y_pred
  # 可使用此方法列印參數 (gradient descent)
  def printParas():
        for name, param in self.named_parameters():
      if param.requires_grad:
        print(name, param.data)
```

5. 設定常參數

```
with torch.no_grad():
    # 這邊的 tensor 不會受到 gradient descent 影響
#或是使用 [torch.tensor].requires_grad = False
```

6 架構 Loss 樣板函式

```
loss_func = nn.MSELoss()
#loss_func = nn.L1Loss()
#loss_func = nn.CrossEntropyLoss()
# y_pred 代表 model 的 output , y_target 為 output 資料 tensor
loss = loss_func(y_pred,y_target)
loss_value = loss.data.numpy()
```

7. 定義 optimizer

```
optimizer = torch.optim.Adam(autoencoder.parameters())
optimizer = torch.optim.SGD(autoencoder.parameters())

#_____________________________________________________________________#
optimizer.zero_grad()  # clear gradients for this training step
loss.backward()          # backpropagation, compute gradients
optimizer.step()
#_____________________________________________________________________#
```

8. 定義 epoch - iteration

```
for epoch in range(n_epochs):
    train_loader = Data.DataLoader(dataset=data,batch_size=n_batchsize,
shuffle=True)
    for _iteridx,x in enumerate(train_loader):
      y_pred = model(x)
    # loss is tensor form
        loss = loss_func(y_pred,y_target)
        optimizer.zero_grad()  # clear gradients for this training step

        loss.backward()
        optimizer.step()
```

9. Batch Output

```
#model = nn.Linear(#batch output)
```