Autoencoder PyTorch MNIST 實作

金門大學資工系 馮玄明整理

實作專案 Autoencoder 實作 MNIST

- 對象:大學與研究所初學者
- 目的:學習 Python 設計Autoencoder完成數位手寫影像壓縮與還原的應用
- 來源 https://morvanzhou.github.io/tutorials/

Autoencoder 實作 MNIST

程式碼解說1

```
"""
來源 visit my tutorial page: https://morvanzhou.github.io/tutorials/
My Youtube Channel: https://www.youtube.com/user/MorvanZhou
Dependencies:torch: 0.4 matplotlib numpy
展示 AUtoencoder 與 Decoder MNIST 手寫資料集的應用
"""
import torch
import torch.nn as nn
import torch.utils.data as Data
import torchvision
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
import numpy as np

# torch.manual_seed(1)    # reproducible
""" Hyper Parameters 參數設定 """
EPOCH = 10
BATCH_SIZE = 64       # 批次處理大小
LR = 0.005            # learning rate 學習率
DOWNLOAD_MNIST = True
N_TEST_IMG = 5

# # 下述為下載 MNIST dataset 轉換成 DataLoader  Mnist digits dataset
train_data = torchvision.datasets.MNIST(
    root='./mnist/',
```

10 Autoencoder 實作 MNIST

程式碼解說2

```python
    root='./mnist/',
    train=True,                                     # this is training data
    transform=torchvision.transforms.ToTensor(),    # Converts a PIL.Image or numpy.ndarray to
                                                    # torch.FloatTensor of shape (C x H x W) and normalize in the
    download=DOWNLOAD_MNIST,                        # download it if you don't have it
)

# plot one example 畫出一個影像範例檔
print(train_data.train_data.size())     # 訓練資料集大小 (60000, 28, 28)
print(train_data.train_labels.size())   # 訓練資料label 大小 (60000)
plt.imshow(train_data.train_data[2].numpy(), cmap='gray')
plt.title('%i' % train_data.train_labels[2])
plt.show()
# 轉換為最小批次量訓練資料集的 型態
# Data Loader for easy mini-batch return in training, the image batch shape will be (50, 1, 28, 28)
#轉換為訓練資料載體 DataLoader
train_loader = Data.DataLoader(dataset=train_data, batch_size=BATCH_SIZE, shuffle=True)

# 利用 nn.Linear 與 nn.Tanh 功能完成 Encoder程序
class AutoEncoder(nn.Module):
    def __init__(self):
        super(AutoEncoder, self).__init__()

        self.encoder = nn.Sequential(
            nn.Linear(28*28, 128),
            nn.Tanh(),
```

程式碼解說3

```python
            nn.Linear(128, 64),
            nn.Tanh(),
            nn.Linear(64, 12),
            nn.Tanh(),
            nn.Linear(12, 3),   # 壓縮到3維的特徵集方便可視化的 3D圖展現
        ) #下述是連串的 decoder將3維度code還原為原始大小的影像圖形
        self.decoder = nn.Sequential(
            nn.Linear(3, 12),
            nn.Tanh(),
            nn.Linear(12, 64),
            nn.Tanh(),
            nn.Linear(64, 128),
            nn.Tanh(),
            nn.Linear(128, 28*28), # 還原為 28*28 的原始影像大小
            nn.Sigmoid(),       # compress to a range (0, 1)
        )

    def forward(self, x):
        encoded = self.encoder(x) # 影像資料 X 輸入到 encoder模組產生 encoded後的code
        decoded = self.decoder(encoded) # Code 輸入 decoder 模組 decoded 後的 原始影像檔
        return encoded, decoded

""" 模組參數最佳化的程序開始 """
autoencoder = AutoEncoder()

optimizer = torch.optim.Adam(autoencoder.parameters(), lr=LR)
```

Autoencoder 實作 MNIST

程式碼解說 4

```
loss_func = nn.MSELoss()

"""# 初始化圖形 initialize figure"""
f, a = plt.subplots(2, N_TEST_IMG, figsize=(5, 2))
plt.ion()   # 連續的出來 continuously plot

"""# original data (first row) for viewing  初始化圖形的展示"""
view_data = train_data.train_data[:N_TEST_IMG].view(-1, 28*28).type(torch.FloatTensor)/255.
for i in range(N_TEST_IMG):
    a[0][i].imshow(np.reshape(view_data.data.numpy()[i], (28, 28)), cmap='gray'); a[0][i].set_xticks(()); a[0][i]

for epoch in range(EPOCH):
    for step, (x, b_label) in enumerate(train_loader):
        b_x = x.view(-1, 28*28)   # batch x, shape (batch, 28*28) 每一批次輸入訓練資料的大小
        b_y = x.view(-1, 28*28)   # batch y, shape (batch, 28*28) 每一批次輸出訓練資料的大小

        encoded, decoded = autoencoder(b_x) # b_x 輸入訓練模組

        loss = loss_func(decoded, b_y)      # 真實輸出 b_y與 訓練輸出的誤差值 mean square error
        optimizer.zero_grad()               # 清除 gradude 的值 clear gradients for this training step
        loss.backward()                     # 計算倒傳遞的Grade大小 backpropagation, compute gradients
        optimizer.step()                    # 輸入倒傳遞的Grade apply gradients
                       # 顯示輸出訓練誤差的大小
        if step % 100 == 0:
            print('Epoch: ', epoch, '| train loss: %.4f' % loss.data.numpy())
```

Autoencoder 實作 MNIST

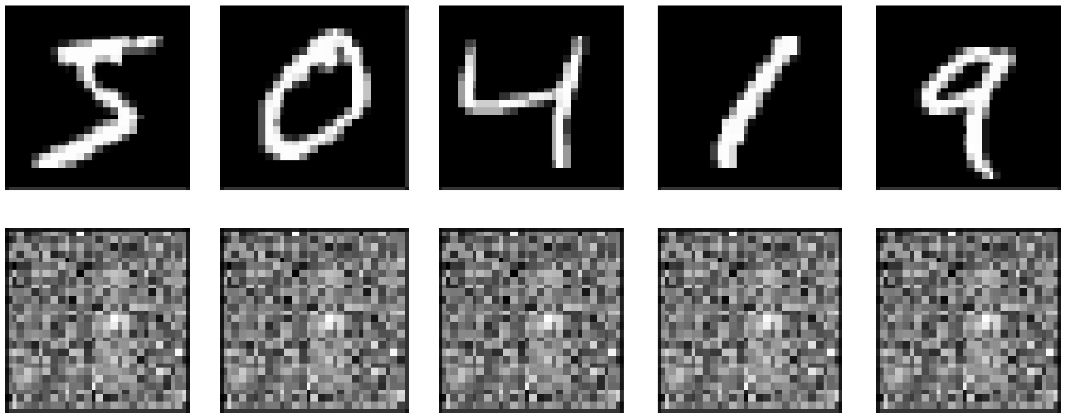
程式碼解說 5

```
                                            # 顯示輸出訓練誤差的大小
            if step % 100 == 0:
                print('Epoch: ', epoch, '| train loss: %.4f' % loss.data.numpy())

                # plotting decoded image (second row) 顯示 還原後影像值
                _, decoded_data = autoencoder(view_data)
                for i in range(N_TEST_IMG):
                    a[1][i].clear()
                    a[1][i].imshow(np.reshape(decoded_data.data.numpy()[i], (28, 28)), cmap='gray')
                    a[1][i].set_xticks(()); a[1][i].set_yticks(())
                plt.draw(); plt.pause(0.05)

plt.ioff()
plt.show()

# visualize in 3D plot 3D顯示方式 可視化的方式展示出來
view_data = train_data.train_data[:200].view(-1, 28*28).type(torch.FloatTensor)/255.
encoded_data, _ = autoencoder(view_data)
fig = plt.figure(2); ax = Axes3D(fig)
X, Y, Z = encoded_data.data[:, 0].numpy(), encoded_data.data[:, 1].numpy(), encoded_data.data[:, 2].numpy()
values = train_data.train_labels[:200].numpy()
for x, y, z, s in zip(X, Y, Z, values):
    c = cm.rainbow(int(255*s/9)); ax.text(x, y, z, s, backgroundcolor=c)
ax.set_xlim(X.min(), X.max()); ax.set_ylim(Y.min(), Y.max()); ax.set_zlim(Z.min(), Z.max())
plt.show()
```

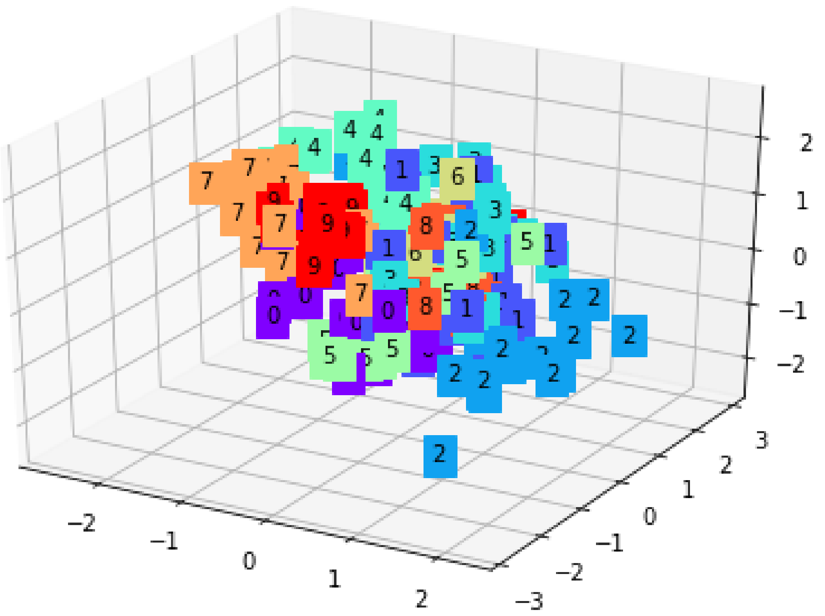
Autoencoder 實作 MNIST

程式輸出

原始訓練用資料



產生的壓縮碼



還原產生的分佈群族

thank you!

謝謝聆聽

THANK YOU FOR YOUR ATTENTION