[실습] PyTorch 의 MLP 프로그래밍

Untitled0.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

댓글 공유 B

+ 코드 + 텍스트

실행 완료 시 알림을 받으려면 설정에서 브라우저 알림을 사용 설정하세요.

확인 아니요

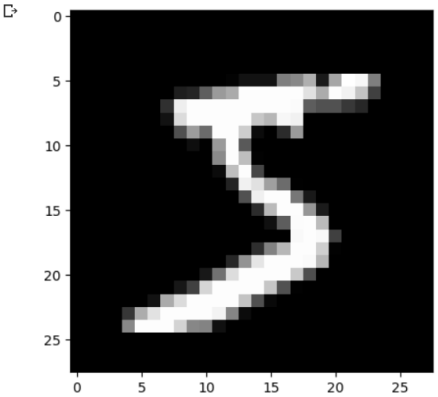
```
# -*- coding: utf-8 -*-
from sklearn.datasets import fetch_openml
mnist = fetch_openml('mnist_784', version=1, cache=True, as_frame=False)

X = mnist.data/255
y = mnist.target

import matplotlib.pyplot as plt
plt.imshow(X[0].reshape(28, 28), cmap='gray')
plt.show()
print("이미지 레이블 :{}".format(y[0]))

import torch
from torch.utils.data import TensorDataset, DataLoader
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1/7., random_state=0)
X_train = torch.Tensor(X_train)
X_test = torch.Tensor(X_test)
y_train = torch.LongTensor(list(map(int, y_train)))
y_test = torch.LongTensor(list(map(int, y_test)))

ds_train = TensorDataset(X_train, y_train)
ds_test = TensorDataset(X_test, y_test)

loader_train = DataLoader(ds_train, batch_size=64, shuffle=True)
loader_test = DataLoader(ds_test, batch_size=64, shuffle=False)

from torch import nn
model = nn.Sequential()
model.add_module('fc1', nn.Linear(in_features=28*28*1, out_features=100))
model.add_module('relu1', nn.ReLU())
```

38 초

```
model.add_module('fc2', nn.Linear(in_features=100, out_features=100))
model.add_module('relu2', nn.ReLU())
model.add_module('fc3', nn.Linear(in_features=100, out_features=10))

from torch import optim
loss_fn = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.01)

def train(epoch):
  model.train()
  for data, target in loader_train:
    optimizer.zero_grad()
    outputs = model(data)
    loss = loss_fn(outputs, target)
    loss.backward()
    optimizer.step()
  print('epoch {}:완료'.format(epoch))

def test(record_property):
  model.eval()
  correct = 0
  with torch.no_grad():
    for data, targets in loader_test:
      outputs = model(data)
      _,predicted = torch.max(outputs.data, 1)
      correct += predicted.eq(targets.data.view_as(predicted)).sum()
  data_num = len(loader_test.dataset)
  print('accuracy = ', 100.*correct/data_num)

for epoch in range(3):
  train(epoch)
  test('학습중')
```

48 초

확인 아니요



```
이미지 레이블 :5
epoch 0:완료
accuracy =  tensor(95.1500)
epoch 1:완료
accuracy =  tensor(95.4900)
epoch 2:완료
accuracy =  tensor(95.7000)
```

[실습]CNN모델을 이용한 MNIST 데이터 분류

```
/usr/local/lib/python3.10/dist-packages/sklearn/datasets/_openml.py:968: FutureWarning: The default value of `parser` will change from `'liac-arff'` to `'auto'` in 1.4. You can set `parser='auto'`
  warn(
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-6-ff68bd04f5c3> in <cell line: 12>()
     10
     11 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1/7, random_state=0)
---> 12 X_train = torch.Tensor(X_train)
     13 X_test = torch.Tensor(X_test)
     14 y_train = torch.LongTensor(list(map(int, y_train)))

ValueError: could not determine the shape of object type 'DataFrame'

SEARCH STACK OVERFLOW
```

오류 발생

```
import numpy as np

X_train = torch.Tensor(np.array(X_train))
X_test = torch.Tensor(np.array(X_test))
```

해당코드 추가(DataFrame 객체를 numpy 배열로 변환)

```
# -*- coding: utf-8 -*-
from sklearn.datasets import fetch_openml
mnist = fetch_openml('mnist_784', version=1, cache=True)
X = mnist.data
y = mnist.target

import torch
from torch.utils.data import TensorDataset, DataLoader
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1/7, random_state=0)
import numpy as np

X_train = torch.Tensor(np.array(X_train))
X_test = torch.Tensor(np.array(X_test))
X_train = torch.Tensor(X_train)
X_test = torch.Tensor(X_test)
y_train = torch.LongTensor(list(map(int, y_train)))
y_test = torch.LongTensor(list(map(int, y_test)))

import torch.nn as nn
import torch.nn.functional as F
from torch import optim
from torch.autograd import Variable

X_train = X_train.view(-1, 1, 28, 28).float()
X_test = X_test.view(-1, 1, 28, 28).float()
print(X_train.shape)
print(X_test.shape)
```

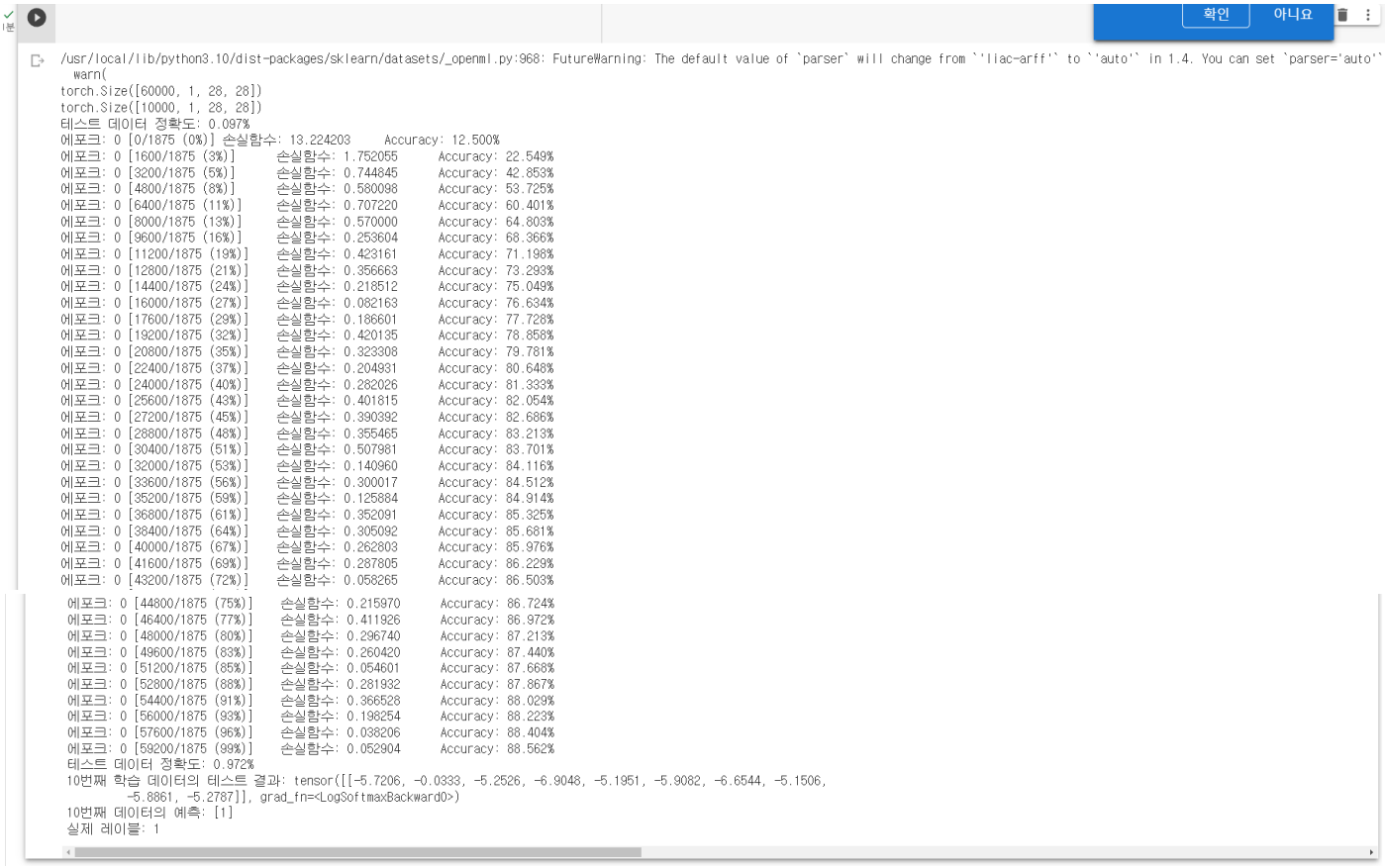
```
train = TensorDataset(X_train, y_train)
test = TensorDataset(X_test, y_test)
BATCH_SIZE = 32
loader_train = DataLoader(train, batch_size=BATCH_SIZE, shuffle=False)
loader_test = DataLoader(test, batch_size=BATCH_SIZE, shuffle=False)


class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, kernel_size=5)
        self.conv2 = nn.Conv2d(32, 32, kernel_size=5)
        self.conv3 = nn.Conv2d(32, 64, kernel_size=5)
        self.fc1 = nn.Linear(3 * 3 * 64, 256)
        self.fc2 = nn.Linear(256, 10)

        self.loss_fn = nn.CrossEntropyLoss()
        self.optimizer = optim.Adam(self.parameters(), lr=0.01)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = F.relu(F.max_pool2d(self.conv2(x), 2))
        x = F.dropout(x, p=0.5, training=self.training)
        x = F.relu(F.max_pool2d(self.conv3(x), 2))
        x = F.dropout(x, p=0.5, training=self.training)
        x = x.view(-1, 3 * 3 * 64)
        x = F.relu(self.fc1(x))
        x = F.dropout(x, training=self.training)
        x = self.fc2(x)
        return F.log_softmax(x, dim=1)


def fit(model, loader_train):
    optimizer = torch.optim.Adam(model.parameters())
```

```
    error = nn.CrossEntropyLoss()
    EPOCHS = 1
    model.train()
    for epoch in range(EPOCHS):
        correct = 0
        for batch_idx, (X_batch, y_batch) in enumerate(loader_train):
            var_X_batch = Variable(X_batch).float()
            var_y_batch = Variable(y_batch)
            optimizer.zero_grad()
            output = model(var_X_batch)
            loss = error(output, var_y_batch)
            loss.backward()
            optimizer.step()
            predicted = torch.max(output.data, 1)[1]
            correct += (predicted == var_y_batch).sum()
            if batch_idx % 50 == 0:
                print('에포크: {} [{}/{} ({:.0f}%)]₩t손실함수: {:.6f}₩tAccuracy: {:.3f}%'.format(
                    epoch, batch_idx * len(X_batch), len(loader_train),
                    100. * batch_idx / len(loader_train),
                    loss.data,
                    correct * 100. / (BATCH_SIZE * (batch_idx + 1))
                ))


def evaluate(model):
    correct = 0
    for test_imgs, test_labels in loader_test:
        test_imgs = Variable(test_imgs).float()
        output = model(test_imgs)
        predicted = torch.max(output, 1)[1]
        correct += (predicted == test_labels).sum()
    print("테스트 데이터 정확도: {:.3f}%".format(float(correct) / (len(loader_test) * BATCH_SIZE)))
```

```
cnn = CNN()
evaluate(cnn)
fit(cnn, loader_train)
cnn.eval()  # 모델 테스트 모드로 전환
evaluate(cnn)
index = 10  # 테스트 데이터 중에서 확인해볼 데이터의 인덱스
data = X_test[index].view(-1, 1, 28, 28).float()
output = cnn(data)  # 모델 적용
print('{}번째 학습 데이터의 테스트 결과: {}'.format(index, output))
_, predicted = torch.max(output, 1)
print('{}번째 데이터의 예측: {}'.format(index, predicted.numpy()))
print('실제 레이블: {}'.format(y_test[index]))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/datasets/_openml.py:968: FutureWarning: The default value of `parser` will change from `'liac-arff'` to `'auto'` in 1.4. You can set `parser='auto'`
  warn(
torch.Size([60000, 1, 28, 28])
torch.Size([10000, 1, 28, 28])
테스트 데이터 정확도: 0.095%
에포크: 0 [17600/1875 (29%)]    손실함수: 0.197504      Accuracy: 76.236%
에포크: 0 [19200/1875 (32%)]    손실함수: 0.469353      Accuracy: 77.433%
에포크: 0 [20800/1875 (35%)]    손실함수: 0.645111      Accuracy: 78.413%
에포크: 0 [22400/1875 (37%)]    손실함수: 0.127778      Accuracy: 79.351%
에포크: 0 [24000/1875 (40%)]    손실함수: 0.198131      Accuracy: 80.143%
에포크: 0 [25600/1875 (43%)]    손실함수: 0.515630      Accuracy: 80.965%
에포크: 0 [27200/1875 (45%)]    손실함수: 0.801752      Accuracy: 81.661%
에포크: 0 [28800/1875 (48%)]    손실함수: 0.181151      Accuracy: 82.277%
에포크: 0 [30400/1875 (51%)]    손실함수: 0.201943      Accuracy: 82.840%
에포크: 0 [32000/1875 (53%)]    손실함수: 0.175012      Accuracy: 83.304%
에포크: 0 [33600/1875 (56%)]    손실함수: 0.127306      Accuracy: 83.736%
에포크: 0 [35200/1875 (59%)]    손실함수: 0.049058      Accuracy: 84.171%
에포크: 0 [36800/1875 (61%)]    손실함수: 0.710536      Accuracy: 84.543%
에포크: 0 [38400/1875 (64%)]    손실함수: 0.196238      Accuracy: 84.924%
에포크: 0 [40000/1875 (67%)]    손실함수: 0.162745      Accuracy: 85.279%
에포크: 0 [41600/1875 (69%)]    손실함수: 0.135767      Accuracy: 85.540%
에포크: 0 [43200/1875 (72%)]    손실함수: 0.104871      Accuracy: 85.830%
에포크: 0 [44800/1875 (75%)]    손실함수: 0.248443      Accuracy: 86.084%
에포크: 0 [46400/1875 (77%)]    손실함수: 0.243570      Accuracy: 86.378%
에포크: 0 [48000/1875 (80%)]    손실함수: 0.580205      Accuracy: 86.611%
에포크: 0 [49600/1875 (83%)]    손실함수: 0.136951      Accuracy: 86.863%
에포크: 0 [51200/1875 (85%)]    손실함수: 0.177566      Accuracy: 87.080%
에포크: 0 [52800/1875 (88%)]    손실함수: 0.083187      Accuracy: 87.271%
에포크: 0 [54400/1875 (91%)]    손실함수: 0.225709      Accuracy: 87.469%
에포크: 0 [56000/1875 (93%)]    손실함수: 0.100058      Accuracy: 87.653%
에포크: 0 [57600/1875 (96%)]    손실함수: 0.084580      Accuracy: 87.866%
에포크: 0 [59200/1875 (99%)]    손실함수: 0.083116      Accuracy: 88.049%
테스트 데이터 정확도: 0.962%
10번째 학습 데이터의 테스트 결과: tensor([[-6.8075, -0.0423, -5.4806, -7.2606, -5.6500, -5.9576, -7.2568, -4.3070,
         -4.7368, -5.0527]], grad_fn=<LogSoftmaxBackward0>)
10번째 데이터의 예측: [1]
실제 레이블: 1
```

2차 테스트



10 번째 학습 데이터의 테스트 결과: tensor([[-5.7206, -0.0333, -5.2526, -6.9048, -5.1951, -5.9082, -6.6544, -5.1506,
-5.8861, -5.2787]], grad_fn=<LogSoftmaxBackward0>)
10 번째 데이터의 예측: [1]
실제 레이블: 1