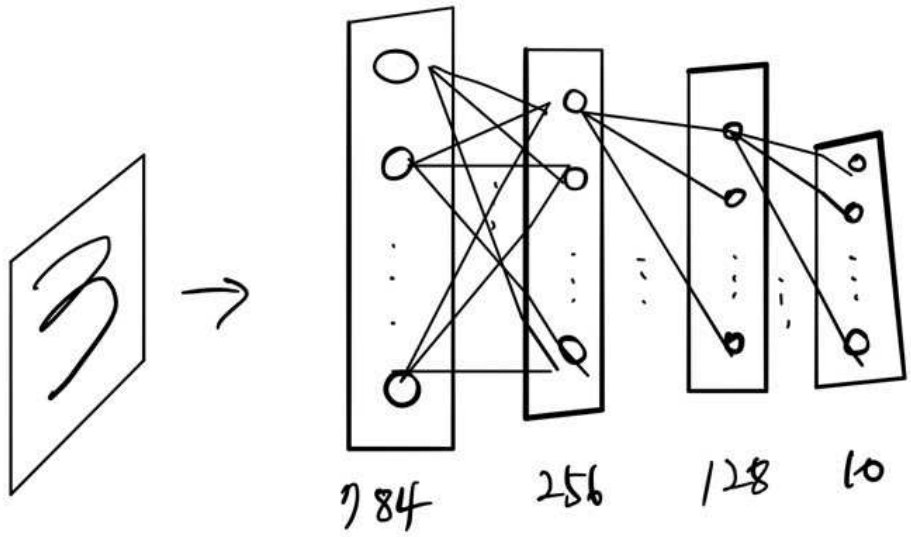
심층학습
[실습02] MLP-학습

SW융합학부 양희경

MLP 를 쉽게 구현하기

- 데이터셋: MNIST
- Multi-layered perceptron (MLP)
  - 2 hidden layers
- Forward propagation & backpropagation



```
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision.datasets as dset
import torchvision.transforms as transforms
from torch.utils.data import DataLoader
from torch.autograd import Variable
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
```

1. MNIST train, test dataset 가져오기

```
# "": 현재 폴더에 MNIST 있음
mnist_train=dset.MNIST("", train=True,transform=transforms.ToTensor(),   #train 용으로 쓰겠다.
                       target_transform=None, download=True)
mnist_test=dset.MNIST("", train=False,transform=transforms.ToTensor(),   #test 용으로 쓰겠다.
                       target_transform=None, download=True)
```

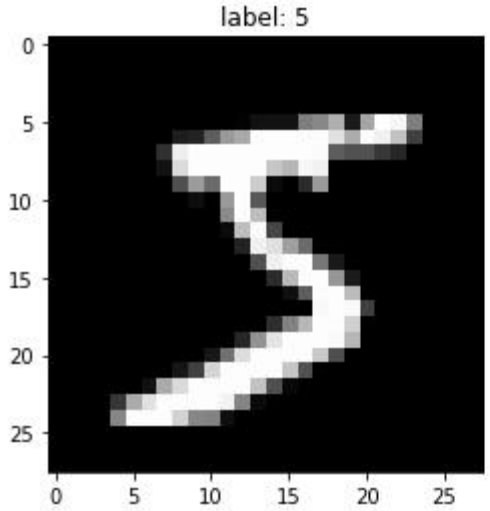
2. 대략적인 데이터 형태

```
print "mnist_train 길이:", len(mnist_train)
print "mnist_test 길이:", len(mnist_test)

# 데이터 하나 형태
image, label = mnist_train.__getitem__(0)  # 0번째 데이터
print "image data 형태:", image.size()
print "label: ", label

# 그리기
img = image.numpy()  # image 타입을 numpy 로 변환 (1,28,28)
plt.title("label: %d" %label )
plt.imshow(img[0], cmap='gray')
plt.show()
```

```
mnist_train 길이: 60000
mnist_test 길이: 10000
image data 형태: torch.Size([1, 28, 28])
label:  5
```



3. 데이터 로드함수

학습시킬 때 batch_size 단위로 끊어서 로드하기 위함

```
# hyper parameters
batch_size = 1024
learning_rate = 0.01  # 0.1, 0.01, 0.001, 0.0001, ...
num_epoch = 400
```

```
train_loader = torch.utils.data.DataLoader(mnist_train,
                                           batch_size=batch_size,  # mnist_train 를 트레인 시키자.
                                           shuffle=True, num_workers=2,
                                           drop_last=True)  # batch_size 만큼 나눌 때 나머지는 버려라
test_loader = torch.utils.data.DataLoader(mnist_test, batch_size=batch_size,
                                          shuffle=False, num_workers=2,
                                          drop_last=True)
```

데이터 로드함수 이해하기

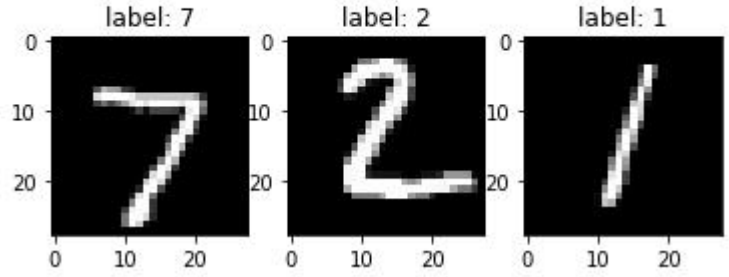
```
n = 3  # 샘플로 그려볼 데이터 개수
for i, [imgs, labels] in enumerate(test_loader):  # batch_size 만큼
    if i>5:
        break

    print "[%d]" %i
    print "한 번에 로드되는 데이터 크기:", len(imgs)

    # 그리기
    for j in range(n):
        img = imgs[j].numpy()  # image 타입을 numpy 로 변환 (1,28,28)
        img = img.reshape((img.shape[1], img.shape[2]))  # (1,28,28) -> (28,28)
        #print img.shape

        plt.subplot(1, n, j+1)  # (1,3) 형태 플랏의 j 번째 자리에 그리겠다
        plt.imshow(img, cmap='gray')
        plt.title("label: %d" %labels[j] )
    plt.show()
```

```
[0]
한 번에 로드되는 데이터 크기: 1024
```



4. 모델 선언

```
# 모델 선언
# * 퍼셉트론(2 hidden layer) *
model = nn.Sequential(
    nn.Linear(28*28,256),
    nn.Sigmoid(),   #nn.ReLU(),    # nn.Sigmoid()   91.89%
    nn.Linear(256,128),
    nn.Linear(128,10),
)
# 파라미터 보기
#print(list(model.parameters()))  # 초기 파라미터 출력
```

```
#model = model.cuda()
```

[실습01] 中

5. 모델 선언

```
# Multi-layered perceptron
# # of units in each layer: 28*28 - 256 - 128 - 10
class MyMLP:
    def __init__(self, n_input, n_hidden1, n_hidden2, n_output):
        # W^(1): layer1 -> layer2 에 매핑되는 Weight
        self.W1 = np.zeros((n_hidden1, n_input), dtype=np.float32)  # W1(256, 28*28)
        self.b1 = np.zeros((n_hidden1,), dtype=np.float32)

        self.W2 = np.zeros((n_hidden2, n_hidden1), dtype=np.float32)  # W2(128, 256)
        self.b2 = np.zeros((n_hidden2,), dtype=np.float32)

        self.W3 = np.zeros((n_output, n_hidden2), dtype=np.float32)  # W3(10, 128)
        self.b3 = np.zeros((n_output), dtype=np.float32) # b3

    def __call__(self, x):
        # (1,28,28) -> (28*28)
        x = x.reshape(-1)  # 일렬로 피기

        h1  = sigmoid(np.dot(self.W1, x) + self.b1)   # W1(256, 28*28), x(28*28), b1(256) -> h1(256)
        h2  = np.dot(self.W2, h1) + self.b2  # W2(128, 256), h1(256), b2(128) -> h2(128)
        out = np.dot(self.W3, h2) + self.b3  # W3(10, 128), h2(128), b3(10) -> out(10)

        return softmax(out)  # (10)
```

```python
def ComputeAccr(dloader, imodel):
    correct = 0
    total = 0

    for j, [imgs, labels] in enumerate(dloader):  # batch_size 만큼
        img = imgs  # x
        label = Variable(labels)  # y
        #label = Variable(labels).cuda()
        # .cuda() : GPU 에 로드되기 위함. 만약 CPU로 설정되어 있다면 에러남

        # (batch_size, 1, 28, 28) -> (batch_size, 28, 28)
        img = img.reshape((img.shape[0], img.shape[2], img.shape[3]))
        # (batch_size, 28, 28) -> (batch_size, 28*28)
        img = img.reshape((img.shape[0], img.shape[1]*img.shape[2]))
        img = Variable(img, requires_grad=False)
        #img = Variable(img, requires_grad=False).cuda()

        output = imodel(img)  # forward prop.
        _, output_index = torch.max(output, 1)

        total += label.size(0)
        correct += (output_index == label).sum().float()
    print("Accuracy of Test Data: {}".format(100*correct/total))
```

```python
ComputeAccr(test_loader, model)
```

```
Accuracy of Test Data: 10.2213544846
```

[실습01] 中

8. 테스트

```python
mysum = 0

m = len(mnist_test)
cnt = 0
for i in range(m):
    image, label = mnist_test.__getitem__(i)  # 0번째 데이터
    output = model(image)

    if (i%1000==0):
        img = image.numpy()  # image 타입을 numpy 로 변환 (1,28,28)
        pred_label = np.argmax(output)
        plt.title("pred: %d, label: %d" %(pred_label, label) )
        plt.imshow(img[0], cmap='gray')
        plt.show()

    cnt += 1
    mysum += (np.argmax(output) == label)
print "정확도: %.2f" %( (float(mysum) / cnt) * 100.0 )
```

5. loss, optimizer

```
1  loss_func = nn.CrossEntropyLoss() # logit(# of classes), target(1)
2  optimizer = optim.SGD(model.parameters(), lr=learning_rate)
```

6. 학습

```
num_epoch = 400
for i in range(num_epoch):
    for j, [imgs, labels] in enumerate(train_loader):  # batch_size 만큼
        img = imgs   # (batch_size, 1, 28, 28)
        label = Variable(labels)   # (batch_size)
        #label = Variable(labels).cuda()  # (batch_size)

        # (batch_size, 1, 28, 28) -> (batch_size, 28, 28)
        img = img.reshape((img.shape[0], img.shape[2], img.shape[3]))
        # (batch_size, 28, 28) -> (batch_size, 28*28)
        img = img.reshape((img.shape[0], img.shape[1]*img.shape[2]))
        img = Variable(img, requires_grad=True)
        #img = Variable(img, requires_grad=True).cuda()

        optimizer.zero_grad()
        output = model(img)  # forward prop.
        loss = loss_func(output, label)  # logit(# of classes), target(1)

        loss.backward()  # back prop.
        optimizer.step() # weight 조정

    if i%50==0:
        print("%d.." %i)
        ComputeAccr(test_loader, model)
        print loss
```

```
0..
Accuracy of Test Data: 12.3155384064
tensor(2.2944, grad_fn=<NllLossBackward>)
50..
Accuracy of Test Data: 81.239151001
tensor(0.6966, grad_fn=<NllLossBackward>)
100..
Accuracy of Test Data: 88.3355026245
tensor(0.3954, grad_fn=<NllLossBackward>)
```

7. 테스트

```
1  ComputeAccr(test_loader, model)   # 98.  %(ReLU), 92.48%(ReLU X)
```

```
Accuracy of Test Data: 91.9921875
```

8. 학습된 파라미터 저장

```
1  netname = './nets/mlp_weight.pkl'
2  torch.save(model, netname, )
3
4  #model = torch.load(netname)
```

9. (Optional) 실습1 에 쓰인 .npz 만드려면?

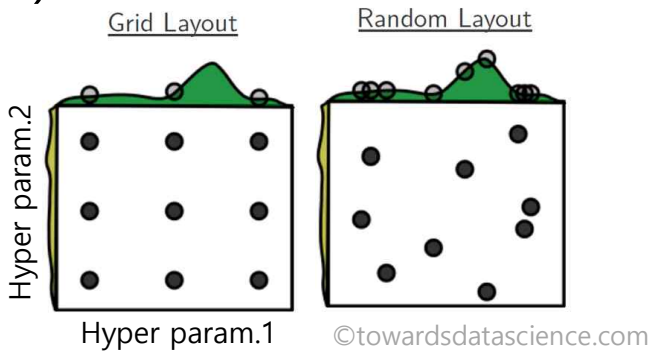
```
np.savez_compressed('./nets/mlp_weight.npz',
                    W1=W1, b1=b1,
                    W2=W2, b2=b2,
                    W3=W3, b3=b3)
```

Q. Hyper parameter 는 어떻게 정하나?

- Hyper parameter
  - Learning rate, # of epochs, hidden layers, hidden units, activation functions, …

- Training/ test/ validation(development) dataset 중 validation dataset 사용하여 성능 측정 후 선택

- Hyper parameter tuning 방법(교재 2.7)
  - 그리드 탐색Grid search
  - 랜덤 탐색Random search



Q. Hyper parameter 는 어떻게 정하나?

- 기계학습/심층학습 이론 공부할 때
  - 기본 개념 이해
  - 주어진 hyper parameter 로 실습
  - 변화를 줘가며 실험

- 나만의 프로젝트 만들 때
  - 나의 task 와 유사한 기존 연구의 hyper parameter 를 참고 (수업, 논문, 깃허브)
  - 유사한 연구가 없을 때?

Q. Hyper parameter 는 어떻게 정하나?

Data Structure		Train / Test	Wiki7K/40K													YM4K										Wiki4K + YM4K									
			Wiki_test1K					Hyper param.					Synth	YM_te	YM_te	net 이름		YM_test600				Synth	YM_te	Wiki_t	net 이름		YM_test600				Synth	YM_te	Wiki_t	net 이름	
			mAP	f1	val	test	train	# of no	dropou	depth	epoc						mAP	f1	val	test	train	epoch				mAP	f1	val	test	train	epoch					mAP
Baseline	AlexNet		0.74	0.67	0.66						40/100	0_Basic_d00						0.74																	
	VGGNet		0.72	0.68	0.66	0.68	0.98	1024	0.5	3	80/100	0_Basic_	0.37	0.47	0.48	0_Basic_b00		0.78								0.86									
	GoogLeNet		0.82	0.75	0.71	0.74	0.97	256	0.5	1	70/100	0_Basic_e00						0.83																	
	ResNet		0.81	0.72	0.72	0.72	0.97	128	0.5	2	100/10	0_Basic_	0.46	0.57	0.6	0_Basic_c00		0.82					0.81		media_r	0.91									
	DenseNet		0.82	0.75	0.71	0.74	0.88	64	0.5	2	20/100	0_Basic_	0.43	0.61	0.59	0_Basic_a00		0.85					0.91		media_d	0.93									
Baseline+Wsum	VGGNet			0.66									0.4	0.5	0.45																			
	ResNet			0.74									0.46	0.62	0.66			0.86					0.89		media_res152_RCrop_fixed									
	DenseNet			0.77									0.44	0.64	0.64			0.87					0.94		media_dense_RCrop_fixed									
[Sun17]	VGGNet		0.81	0.73	0.74	0.73	0.98	128	0.5	2	70/100		0.46	0.6	0.71	1_Gram_b00_o 1_Gram_b00_t		0.89				0.52	0.92		Gram_b00									
	ResNet		0.83	0.75	0.71	0.74	0.99	128	0.5	2	100/100		0.42	0.58	0.6	1_Gram_c00_o(del) 1_Gram_c00_t(del)		0.85				0.54	0.9		Gram_c00									
	DenseNet		0.86	**0.77**	0.77	0.77	0.97	64	0.5	3	100/100		0.46	0.65	0.67	1_Gram_a00_o(del) 1_Gram_a00_t(del)		0.86				0.54	0.87		Gram_a00									
[Sun17]+Wsum	VGGNet			0.74				128					0.46	0.62	0.7																			
	ResNet			0.73				128					0.43	0.59	0.6																			
	DenseNet			**0.79**				64	0.5	3			0.47	0.66	0.71																			
1_Sal_TrainTest "slocal"	DenseNet	"max"			0.66	0.71	0.75	48	0.5	3	80/100	~~1_Gram_a01_o(del)~~																							
		"min"			0.74	0.74	0.85	48			90/100																								
		"rand"			0.71	0.72	0.77	48																											
		"mid"			0.74	0.75	0.90	48			80/100																								
		"mid", only tmodel		0.75	0.75	**0.74**	0.85	48			100/10	~~1_TT00~~																							
1_Sal_TrainTest "wslocal"		"min"																																
		"mid"			0.57	0.58	0.59	48			80/100																								
		"mid", only tmodel																																
2_Sal_Test "slocal"		"mid"		0.76	0.75	**0.76**	0.93	64			100/10	~~2_T00~~																							
1. loss에 weight 추가					0.72	0.76	0.90				60/100	~~2_T01~~		1																				
	1) 3개->5개																																	
	2) test할 때 "wslo				0.72	0.76	0.89					~~2_T01~~		2																				
	2)-1 3개->5개																																	
2. only tmodel		sutils/Feed2I	Model 변경함		0.74	0.75	0.87				80/100																								
		5개		0.80	0.78	0.79	0.93				100/100			3																				
	76보다 잘나오면	3개			0.76	0.76	0.96																											
	79보다 못나와야	random		0.78	0.74	0.78	0.83				70/100																								
	Without Gram	"rand"																0.88	0.86	0.88	0.91	50/100			**3_c_ym03**									
		"max"	0.82	0.77	0.74	**0.77**	0.91				70/100	**3_c01**						0.90	0.90	0.90	0.98	90/100			**3_c_ym01**									
		"min"	0.84	0.79	0.74	**0.79**	0.93				100/10	**3_c02**						0.86	0.85	0.86	0.95	90/100			**3_c_ym02**									
		"med"	0.87	0.80	0.78	0.80	0.91				60/100	**3_c00**						0.91	0.90	0.91	0.97	60/100			**3_c_ym00**									
		"q1"																0.90	0.89	0.90	0.99	100/100			3_c_ym0	0.97								
		"q3"																0.90	0.89	0.90	0.97	50/100			3_c_ym0	0.97								
3_SalsTest RandSal Train 여러장 랜덤선	에러수정후(random sal_idx)			0.80	0.78	**0.79**	0.91				100/10	3_T00	weight 결정방법 더 고민할 필요 있음																						
		"rand"(base)	0.87	0.76	0.75	0.76	0.89				70/100	3_T01(del)						0.89	0.8903	0.8969	0.9945	100/100			3_T10(de	0.97								

오늘의 과제

- '[실습02] MLP-학습' 을 실습한다.
  - P.3 ~ 9
- 'HTML, ipynb' 파일을 다운받아 e-campus 에 제출한다.
- 마감: e-campus 과제 메뉴 참조