[실습04] 심층신경망 훈련

201810909 컴퓨터과학과 김부용

CIFAR10을 CNN으로 학습하기. 여러 학습 방법으로 비교

In [2]:

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

#(8) learning rate decay
from torch.optim import lr_scheduler

batch_size=16
learning_rate=0.002
num_epoch=1 # 효과만 확인할 것이기 때문에 1
```

1. CIFAR10 train, test dataset 가져오기 (163MB)

10개의 카테고리에 대한 데이터
유명해서 pytorch dset에서 제공
cf) MNIST: 11MB

In [3]:

```
cifar_train=dset.CIFAR10("CIFAR10/",train=True, transform=transforms.ToTensor(), target_transfo
rm=None, download=True)
cifar_test=dset.CIFAR10("CIFAR10/",train=False, transform=transforms.ToTensor(), target_transfo
rm=None, download=True)
```

```
Files already downloaded and verified
Files already downloaded and verified
```

In [29]:

```
# (2) Data augmentation
cifar_train=dset.CIFAR10("CIFAR10/", train=True,
                         transform=transforms.Compose([
                             transforms.Scale(36),
                             transforms.CenterCrop(32),
                             transforms.RandomHorizontalFlip(),
                             transforms.Lambda(lambda x: x.rotate(90)),
                             transforms.ToTensor(),
                         ]))
cifar_test=dset.CIFAR10("CIFAR10/",train=False, transform=transforms.ToTensor(), target_transfo
rm=None, download=True)
```

```
/home/ec2-user/anaconda3/envs/pytorch_p27/lib/python2.7/site-packages/torchvision/
transforms/transforms.py:220: UserWarning: The use of the transforms.Scale transfo
rm is deprecated, please use transforms.Resize instead.
  "please use transforms.Resize instead.")

Files already downloaded and verified
```

In [47]:

```
# (4) Data normalization
cifar_train=dset.CIFAR10("CIFAR10/", train=True,
                         transform=transforms.Compose([
                             transforms.ToTensor(),
                             transforms.Normalize(mean=(0.5,0.5,0.5), std=(0.5,0.5,0.5)),
                         ])
                         , target_transform=None, download=True)
cifar_test=dset.CIFAR10("CIFAR10/", train=False,
                        transform=transforms.Compose([
                             transforms.ToTensor(),
                             transforms.Normalize(mean=(0.5,0.5,0.5), std=(0.5,0.5,0.5)),
                         ])
                         , target_transform=None, download=True)
```

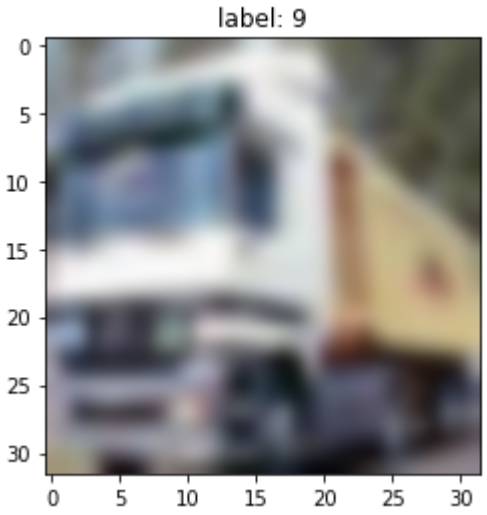
```
Files already downloaded and verified
Files already downloaded and verified
```

2. 대략적인 데이터 형태

In [3]:

```
print "cifar_train 길이:", len(cifar_train)
print "cifar_test 길이:", len(cifar_test)

# 데이터 하나 형태
image, label = cifar_train.__getitem__(1) # 1번째 데이터
print "image data 형태:", image.size()
print "label:", label

# 그리기
img = image.numpy() # image 타입을 numpy로 변환 (3, 32, 32)

# (3, 32, 32) -> (32, 32, 3)
r, g, b = img[0,:,:], img[1,:,:], img[2,:,:]
#img = img.reshape(img.shape[1], img.shape[2], img.shape[0])
img2 = np.zeros((img.shape[1],img.shape[2],img.shape[0])) # width, height, rgb 순서 바꾸기
img2[:,:,0], img2[:,:,1], img2[:,:,2] = r, g, b

plt.title("label: %d" %label)
plt.imshow(img2, interpolation='bicubic')
plt.show()
```

```
cifar_train 길이: 50000
cifar_test 길이: 10000
image data 형태: torch.Size([3, 32, 32])
label: 9
```



In [4]:

```
# test data에 대한 성능
def ComputeAccr(dloader, imodel):
    correct = 0
    total = 0

    for j, [imgs, labels] in enumerate(dloader): # batch_size 만큼
        img = Variable(imgs, volatile=True) # x
        label = Variable(labels)

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

        total += label.size(0) # 데이터 개수 label은 1024개로 이루어진 y값
        correct += (output_index == label).sum().float()
    print("Accuracy of Test Data: {}".format(100*correct/total)) # 만 장에 대한 정확도 측정
```

3. 데이터 로드함수

In [5]:

```
train_loader=torch.utils.data.DataLoader(list(cifar_train)[:], batch_size=batch_size, shuffle=Tr
ue, num_workers=2, drop_last=True)
test_loader=torch.utils.data.DataLoader(cifar_test, batch_size=batch_size, shuffle=False, num_w
orkers=2, drop_last=True)
```

4. 모델 선언

In [6]:

```
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.layer=nn.Sequential(
            nn.Conv2d(3, 16, 3, padding=1),
            nn.ReLU(),
            #nn.Dropout2d(0.2),   # (1) drop out
            #nn.BatchNorm2d(16),  # (5) Batch normalization
            nn.Conv2d(16, 32, 3, padding=1),
            nn.ReLU(),
            #nn.Dropout2d(0.2),   # (1) drop out
            #nn.BatchNorm2d(32),  # (5) Batch normalization
            nn.MaxPool2d(2, 2),
            nn.Conv2d(32, 64, 3, padding=1),
            nn.ReLU(),
            #nn.Dropout2d(0.2),   # (1) drop out
            #nn.BatchNorm2d(64),  # (5) Batch normalization
            nn.MaxPool2d(2, 2)
        )
        self.fc_layer=nn.Sequential(
            nn.Linear(64*8*8, 100),
            nn.ReLU(),
            #nn.Dropout2d(0.2),   # (1) drop out
            #nn.BatchNorm1d(100), # (5) Batch normalization
            nn.Linear(100, 10)
        )

        # (3) weight initialization
        #for m in self.modules():
        #    if isinstance(m, nn.Conv2d):
        #        init.kaiming_normal(m.weight.data) # ReLU일 때
        #        m.bias.data.fill_(0)
        #    if isinstance(m, nn.Linear):
        #        init.kaiming_normal(m.weight.data)
        #        m.bias.data.fill_(0)

    def forward(self, x):
        out=self.layer(x)
        out=out.view(batch_size, -1)
        out=self.fc_layer(out)

        return out
```

In [7]:

```
model=CNN()
```

5. loss, optimizer

In [9]:

```
loss_func=nn.CrossEntropyLoss()
optimizer=torch.optim.SGD(model.parameters(), lr=learning_rate)
```

In [18]:

```
loss_func=nn.CrossEntropyLoss()
optimizer=torch.optim.Adam(model.parameters(), lr=learning_rate) # (6) Adam optimizer
```

In [10]:

```
num_epoch = 60
scheduler = lr_scheduler.StepLR(optimizer, step_size=20, gamma=0.2) # (7) learning rate decay
```

(옵션) epoch 시각화

In [11]:

```
import ipywidgets as widgets
from ipywidgets import Layout,Box
from IPython.display import display
```

6. 학습

In [ ]:

```
bar = widgets.IntProgress(min=0, max=num_epoch,description='epoch:')
label = widgets.Label(value=str(bar.value), disabled=True)
mylink = widgets.jslink((bar, 'value'), (label, 'value'))
visualize = Box([bar, label], layout = Layout(display='flex', flex_flow='row', justify_content=
'flex-start'))
display(visualize)

for i in range(num_epoch):
    for j, [image, label] in enumerate(train_loader):
        x=Variable(image)
        y_=Variable(label)

        optimizer.zero_grad()
        output=model.forward(x)
        loss=loss_func(output,y_)
        loss.backward()
        optimizer.step()

        if j%1000==0:
            print(j, loss)

    bar.value += 1
```

```
(0, tensor(2.2966, grad_fn=<NllLossBackward>))
(1000, tensor(2.2825, grad_fn=<NllLossBackward>))
(2000, tensor(2.2825, grad_fn=<NllLossBackward>))
(3000, tensor(2.3233, grad_fn=<NllLossBackward>))
(0, tensor(2.2616, grad_fn=<NllLossBackward>))
(1000, tensor(1.9504, grad_fn=<NllLossBackward>))
(2000, tensor(1.7915, grad_fn=<NllLossBackward>))
(3000, tensor(2.1911, grad_fn=<NllLossBackward>))
(0, tensor(2.1343, grad_fn=<NllLossBackward>))
(1000, tensor(1.9143, grad_fn=<NllLossBackward>))
(2000, tensor(1.7704, grad_fn=<NllLossBackward>))
(3000, tensor(1.7660, grad_fn=<NllLossBackward>))
(0, tensor(2.2449, grad_fn=<NllLossBackward>))
(1000, tensor(1.5145, grad_fn=<NllLossBackward>))
(2000, tensor(1.5887, grad_fn=<NllLossBackward>))
(3000, tensor(1.6334, grad_fn=<NllLossBackward>))
(0, tensor(1.2081, grad_fn=<NllLossBackward>))
(1000, tensor(1.7744, grad_fn=<NllLossBackward>))
(2000, tensor(1.4192, grad_fn=<NllLossBackward>))
(3000, tensor(1.3809, grad_fn=<NllLossBackward>))
(0, tensor(1.3058, grad_fn=<NllLossBackward>))
(1000, tensor(1.4454, grad_fn=<NllLossBackward>))
(2000, tensor(1.7599, grad_fn=<NllLossBackward>))
(3000, tensor(1.3297, grad_fn=<NllLossBackward>))
(0, tensor(1.3309, grad_fn=<NllLossBackward>))
(1000, tensor(1.5062, grad_fn=<NllLossBackward>))
(2000, tensor(1.0912, grad_fn=<NllLossBackward>))
(3000, tensor(1.2145, grad_fn=<NllLossBackward>))
(0, tensor(1.3351, grad_fn=<NllLossBackward>))
(1000, tensor(1.1315, grad_fn=<NllLossBackward>))
(2000, tensor(1.7244, grad_fn=<NllLossBackward>))
(3000, tensor(1.2720, grad_fn=<NllLossBackward>))
(0, tensor(1.5475, grad_fn=<NllLossBackward>))
(1000, tensor(1.2135, grad_fn=<NllLossBackward>))
(2000, tensor(1.5721, grad_fn=<NllLossBackward>))
(3000, tensor(1.2064, grad_fn=<NllLossBackward>))
(0, tensor(1.7259, grad_fn=<NllLossBackward>))
(1000, tensor(1.6478, grad_fn=<NllLossBackward>))
(2000, tensor(1.0796, grad_fn=<NllLossBackward>))
(3000, tensor(1.4600, grad_fn=<NllLossBackward>))
(0, tensor(1.1740, grad_fn=<NllLossBackward>))
(1000, tensor(1.1920, grad_fn=<NllLossBackward>))
(2000, tensor(1.6026, grad_fn=<NllLossBackward>))
(3000, tensor(1.4698, grad_fn=<NllLossBackward>))
(0, tensor(1.2807, grad_fn=<NllLossBackward>))
(1000, tensor(1.4192, grad_fn=<NllLossBackward>))
(2000, tensor(1.3214, grad_fn=<NllLossBackward>))
(3000, tensor(1.5325, grad_fn=<NllLossBackward>))
(0, tensor(0.7617, grad_fn=<NllLossBackward>))
(1000, tensor(0.7898, grad_fn=<NllLossBackward>))
(2000, tensor(0.9401, grad_fn=<NllLossBackward>))
(3000, tensor(1.1435, grad_fn=<NllLossBackward>))
(0, tensor(1.2619, grad_fn=<NllLossBackward>))
(1000, tensor(1.1432, grad_fn=<NllLossBackward>))
(2000, tensor(1.2467, grad_fn=<NllLossBackward>))
(3000, tensor(0.7010, grad_fn=<NllLossBackward>))
(0, tensor(1.1759, grad_fn=<NllLossBackward>))
(1000, tensor(1.1717, grad_fn=<NllLossBackward>))
(2000, tensor(1.2318, grad_fn=<NllLossBackward>))
(3000, tensor(1.0660, grad_fn=<NllLossBackward>))
(0, tensor(0.7582, grad_fn=<NllLossBackward>))
```

```
(1000, tensor(0.9972, grad_fn=<NllLossBackward>))
(2000, tensor(0.5788, grad_fn=<NllLossBackward>))
(3000, tensor(1.2120, grad_fn=<NllLossBackward>))
(0, tensor(1.2145, grad_fn=<NllLossBackward>))
(1000, tensor(0.6989, grad_fn=<NllLossBackward>))
(2000, tensor(1.1583, grad_fn=<NllLossBackward>))
(3000, tensor(1.0407, grad_fn=<NllLossBackward>))
(0, tensor(0.8332, grad_fn=<NllLossBackward>))
(1000, tensor(0.7417, grad_fn=<NllLossBackward>))
(2000, tensor(1.0189, grad_fn=<NllLossBackward>))
(3000, tensor(1.1335, grad_fn=<NllLossBackward>))
(0, tensor(0.9724, grad_fn=<NllLossBackward>))
(1000, tensor(1.2877, grad_fn=<NllLossBackward>))
(2000, tensor(1.2478, grad_fn=<NllLossBackward>))
(3000, tensor(0.5694, grad_fn=<NllLossBackward>))
(0, tensor(1.0605, grad_fn=<NllLossBackward>))
(1000, tensor(0.9066, grad_fn=<NllLossBackward>))
(2000, tensor(1.4267, grad_fn=<NllLossBackward>))
(3000, tensor(0.6140, grad_fn=<NllLossBackward>))
(0, tensor(0.6598, grad_fn=<NllLossBackward>))
(1000, tensor(1.1578, grad_fn=<NllLossBackward>))
(2000, tensor(0.9150, grad_fn=<NllLossBackward>))
(3000, tensor(0.7155, grad_fn=<NllLossBackward>))
(0, tensor(0.4769, grad_fn=<NllLossBackward>))
(1000, tensor(0.7742, grad_fn=<NllLossBackward>))
(2000, tensor(1.0766, grad_fn=<NllLossBackward>))
(3000, tensor(0.7644, grad_fn=<NllLossBackward>))
(0, tensor(0.8709, grad_fn=<NllLossBackward>))
(1000, tensor(0.2205, grad_fn=<NllLossBackward>))
(2000, tensor(1.0201, grad_fn=<NllLossBackward>))
(3000, tensor(0.7476, grad_fn=<NllLossBackward>))
(0, tensor(0.6514, grad_fn=<NllLossBackward>))
(1000, tensor(0.7037, grad_fn=<NllLossBackward>))
(2000, tensor(1.1951, grad_fn=<NllLossBackward>))
(3000, tensor(0.9482, grad_fn=<NllLossBackward>))
(0, tensor(0.6724, grad_fn=<NllLossBackward>))
(1000, tensor(0.6060, grad_fn=<NllLossBackward>))
(2000, tensor(0.4699, grad_fn=<NllLossBackward>))
(3000, tensor(0.8104, grad_fn=<NllLossBackward>))
(0, tensor(0.6242, grad_fn=<NllLossBackward>))
(1000, tensor(0.4244, grad_fn=<NllLossBackward>))
(2000, tensor(0.6231, grad_fn=<NllLossBackward>))
(3000, tensor(0.7135, grad_fn=<NllLossBackward>))
(0, tensor(0.3240, grad_fn=<NllLossBackward>))
(1000, tensor(0.6956, grad_fn=<NllLossBackward>))
(2000, tensor(0.6362, grad_fn=<NllLossBackward>))
(3000, tensor(0.7800, grad_fn=<NllLossBackward>))
(0, tensor(1.2932, grad_fn=<NllLossBackward>))
(1000, tensor(0.4823, grad_fn=<NllLossBackward>))
(2000, tensor(0.5464, grad_fn=<NllLossBackward>))
(3000, tensor(0.3089, grad_fn=<NllLossBackward>))
(0, tensor(0.6214, grad_fn=<NllLossBackward>))
(1000, tensor(0.9002, grad_fn=<NllLossBackward>))
(2000, tensor(0.6581, grad_fn=<NllLossBackward>))
(3000, tensor(0.3788, grad_fn=<NllLossBackward>))
(0, tensor(0.7711, grad_fn=<NllLossBackward>))
(1000, tensor(0.7482, grad_fn=<NllLossBackward>))
(2000, tensor(0.9870, grad_fn=<NllLossBackward>))
(3000, tensor(0.7011, grad_fn=<NllLossBackward>))
(0, tensor(0.3888, grad_fn=<NllLossBackward>))
(1000, tensor(0.2362, grad_fn=<NllLossBackward>))
```

```
(2000, tensor(0.3800, grad_fn=<NllLossBackward>))
(3000, tensor(0.7369, grad_fn=<NllLossBackward>))
(0, tensor(0.1973, grad_fn=<NllLossBackward>))
(1000, tensor(0.3947, grad_fn=<NllLossBackward>))
(2000, tensor(0.4812, grad_fn=<NllLossBackward>))
(3000, tensor(0.4973, grad_fn=<NllLossBackward>))
(0, tensor(0.3782, grad_fn=<NllLossBackward>))
(1000, tensor(0.5072, grad_fn=<NllLossBackward>))
(2000, tensor(0.5450, grad_fn=<NllLossBackward>))
(3000, tensor(0.2652, grad_fn=<NllLossBackward>))
(0, tensor(0.3293, grad_fn=<NllLossBackward>))
(1000, tensor(0.7194, grad_fn=<NllLossBackward>))
(2000, tensor(0.3523, grad_fn=<NllLossBackward>))
(3000, tensor(0.4418, grad_fn=<NllLossBackward>))
(0, tensor(0.3098, grad_fn=<NllLossBackward>))
(1000, tensor(0.2641, grad_fn=<NllLossBackward>))
(2000, tensor(0.2085, grad_fn=<NllLossBackward>))
(3000, tensor(0.3929, grad_fn=<NllLossBackward>))
(0, tensor(0.5516, grad_fn=<NllLossBackward>))
(1000, tensor(0.0581, grad_fn=<NllLossBackward>))
(2000, tensor(0.3316, grad_fn=<NllLossBackward>))
(3000, tensor(0.3387, grad_fn=<NllLossBackward>))
(0, tensor(0.4070, grad_fn=<NllLossBackward>))
(1000, tensor(0.4430, grad_fn=<NllLossBackward>))
(2000, tensor(0.2175, grad_fn=<NllLossBackward>))
(3000, tensor(0.4337, grad_fn=<NllLossBackward>))
(0, tensor(0.3058, grad_fn=<NllLossBackward>))
(1000, tensor(0.0630, grad_fn=<NllLossBackward>))
(2000, tensor(0.1848, grad_fn=<NllLossBackward>))
(3000, tensor(0.3543, grad_fn=<NllLossBackward>))
(0, tensor(0.1128, grad_fn=<NllLossBackward>))
(1000, tensor(0.3203, grad_fn=<NllLossBackward>))
(2000, tensor(0.5544, grad_fn=<NllLossBackward>))
(3000, tensor(0.1920, grad_fn=<NllLossBackward>))
(0, tensor(0.2886, grad_fn=<NllLossBackward>))
(1000, tensor(0.2393, grad_fn=<NllLossBackward>))
(2000, tensor(0.4475, grad_fn=<NllLossBackward>))
(3000, tensor(0.1046, grad_fn=<NllLossBackward>))
(0, tensor(0.3719, grad_fn=<NllLossBackward>))
(1000, tensor(0.1989, grad_fn=<NllLossBackward>))
(2000, tensor(0.0464, grad_fn=<NllLossBackward>))
(3000, tensor(0.5605, grad_fn=<NllLossBackward>))
(0, tensor(0.2070, grad_fn=<NllLossBackward>))
(1000, tensor(0.1683, grad_fn=<NllLossBackward>))
(2000, tensor(0.1985, grad_fn=<NllLossBackward>))
(3000, tensor(0.1354, grad_fn=<NllLossBackward>))
(0, tensor(0.1249, grad_fn=<NllLossBackward>))
(1000, tensor(0.1547, grad_fn=<NllLossBackward>))
(2000, tensor(0.0863, grad_fn=<NllLossBackward>))
(3000, tensor(0.0700, grad_fn=<NllLossBackward>))
(0, tensor(0.0723, grad_fn=<NllLossBackward>))
(1000, tensor(0.1105, grad_fn=<NllLossBackward>))
(2000, tensor(0.1250, grad_fn=<NllLossBackward>))
(3000, tensor(0.3287, grad_fn=<NllLossBackward>))
(0, tensor(0.1535, grad_fn=<NllLossBackward>))
(1000, tensor(0.0248, grad_fn=<NllLossBackward>))
(2000, tensor(0.3202, grad_fn=<NllLossBackward>))
(3000, tensor(0.2457, grad_fn=<NllLossBackward>))
(0, tensor(0.1567, grad_fn=<NllLossBackward>))
(1000, tensor(0.0493, grad_fn=<NllLossBackward>))
(2000, tensor(0.0288, grad_fn=<NllLossBackward>))
```

```
(3000, tensor(0.3995, grad_fn=<NllLossBackward>))
(0, tensor(0.0126, grad_fn=<NllLossBackward>))
(1000, tensor(0.2284, grad_fn=<NllLossBackward>))
(2000, tensor(0.1295, grad_fn=<NllLossBackward>))
(3000, tensor(0.0512, grad_fn=<NllLossBackward>))
(0, tensor(0.0934, grad_fn=<NllLossBackward>))
(1000, tensor(0.0696, grad_fn=<NllLossBackward>))
(2000, tensor(0.0391, grad_fn=<NllLossBackward>))
(3000, tensor(0.1476, grad_fn=<NllLossBackward>))
(0, tensor(0.0889, grad_fn=<NllLossBackward>))
(1000, tensor(0.1789, grad_fn=<NllLossBackward>))
(2000, tensor(0.1048, grad_fn=<NllLossBackward>))
(3000, tensor(0.0645, grad_fn=<NllLossBackward>))
(0, tensor(0.0090, grad_fn=<NllLossBackward>))
(1000, tensor(0.0278, grad_fn=<NllLossBackward>))
(2000, tensor(0.0231, grad_fn=<NllLossBackward>))
(3000, tensor(0.0213, grad_fn=<NllLossBackward>))
(0, tensor(0.0392, grad_fn=<NllLossBackward>))
(1000, tensor(0.0127, grad_fn=<NllLossBackward>))
(2000, tensor(0.0260, grad_fn=<NllLossBackward>))
(3000, tensor(0.0062, grad_fn=<NllLossBackward>))
(0, tensor(0.0155, grad_fn=<NllLossBackward>))
(1000, tensor(0.0082, grad_fn=<NllLossBackward>))
(2000, tensor(0.0140, grad_fn=<NllLossBackward>))
(3000, tensor(0.0158, grad_fn=<NllLossBackward>))
(0, tensor(0.0932, grad_fn=<NllLossBackward>))
(1000, tensor(0.1255, grad_fn=<NllLossBackward>))
(2000, tensor(0.0359, grad_fn=<NllLossBackward>))
(3000, tensor(0.0170, grad_fn=<NllLossBackward>))
(0, tensor(0.0211, grad_fn=<NllLossBackward>))
(1000, tensor(0.0089, grad_fn=<NllLossBackward>))
(2000, tensor(0.0214, grad_fn=<NllLossBackward>))
(3000, tensor(0.0445, grad_fn=<NllLossBackward>))
(0, tensor(0.0092, grad_fn=<NllLossBackward>))
(1000, tensor(0.0098, grad_fn=<NllLossBackward>))
(2000, tensor(0.0567, grad_fn=<NllLossBackward>))
(3000, tensor(0.0143, grad_fn=<NllLossBackward>))
(0, tensor(0.0027, grad_fn=<NllLossBackward>))
(1000, tensor(0.0009, grad_fn=<NllLossBackward>))
(2000, tensor(0.0041, grad_fn=<NllLossBackward>))
(3000, tensor(0.0356, grad_fn=<NllLossBackward>))
(0, tensor(0.0043, grad_fn=<NllLossBackward>))
(1000, tensor(0.0039, grad_fn=<NllLossBackward>))
(2000, tensor(0.0306, grad_fn=<NllLossBackward>))
(3000, tensor(0.0019, grad_fn=<NllLossBackward>))
(0, tensor(0.0875, grad_fn=<NllLossBackward>))
(1000, tensor(0.0003, grad_fn=<NllLossBackward>))
(2000, tensor(0.0019, grad_fn=<NllLossBackward>))
(3000, tensor(0.0023, grad_fn=<NllLossBackward>))
(0, tensor(0.0015, grad_fn=<NllLossBackward>))
(1000, tensor(0.0048, grad_fn=<NllLossBackward>))
(2000, tensor(0.0068, grad_fn=<NllLossBackward>))
(3000, tensor(0.0010, grad_fn=<NllLossBackward>))
(0, tensor(0.0028, grad_fn=<NllLossBackward>))
(1000, tensor(0.0092, grad_fn=<NllLossBackward>))
(2000, tensor(0.0108, grad_fn=<NllLossBackward>))
(3000, tensor(0.0112, grad_fn=<NllLossBackward>))
```

(0) Naive Test

(0, tensor(2.3118, grad_fn=))
(1000, tensor(2.3132, grad_fn=))
(2000, tensor(2.2907, grad_fn=))
(3000, tensor(2.3037, grad_fn=))

In [10]:

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

```
/home/ec2-user/anaconda3/envs/pytorch_p27/lib/python2.7/site-packages/ipykernel/__
main__.py:7: UserWarning: volatile was removed and now has no effect. Use `with to
rch.no_grad():` instead.

Accuracy of Test Data: 17.5900001526
```

(1) drop out

(0, tensor(2.3044, grad_fn=))
(1000, tensor(2.2906, grad_fn=))
(2000, tensor(2.2922, grad_fn=))
(3000, tensor(2.2836, grad_fn=))

In [26]:

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

```
/home/ec2-user/anaconda3/envs/pytorch_p27/lib/python2.7/site-packages/ipykernel/__
main__.py:7: UserWarning: volatile was removed and now has no effect. Use `with to
rch.no_grad():` instead.

Accuracy of Test Data: 10.5500001907
```

(2) Data augmentation

(0, tensor(2.2904, grad_fn=))
(1000, tensor(2.3117, grad_fn=))
(2000, tensor(2.3024, grad_fn=))
(3000, tensor(2.2993, grad_fn=))

In [35]:

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

```
/home/ec2-user/anaconda3/envs/pytorch_p27/lib/python2.7/site-packages/ipykernel/__
main__.py:7: UserWarning: volatile was removed and now has no effect. Use `with to
rch.no_grad():` instead.

Accuracy of Test Data: 17.7299995422
```

(3) Weight initialization

(0, tensor(2.5656, grad_fn=))
(1000, tensor(1.7319, grad_fn=))
(2000, tensor(1.7721, grad_fn=))
(3000, tensor(1.7549, grad_fn=))

In [44]:

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

```
/home/ec2-user/anaconda3/envs/pytorch_p27/lib/python2.7/site-packages/ipykernel/__
main__.py:7: UserWarning: volatile was removed and now has no effect. Use `with to
rch.no_grad():` instead.

Accuracy of Test Data: 44.5400009155
```

(4) Data normalization

(0, tensor(2.3120, grad_fn=))
(1000, tensor(2.3053, grad_fn=))
(2000, tensor(2.2601, grad_fn=))
(3000, tensor(2.1691, grad_fn=))

In [55]:

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

```
/home/ec2-user/anaconda3/envs/pytorch_p27/lib/python2.7/site-packages/ipykernel/__
main__.py:7: UserWarning: volatile was removed and now has no effect. Use `with to
rch.no_grad():` instead.

Accuracy of Test Data: 25.3600006104
```

(5) Batch normalization

(0, tensor(2.5951, grad_fn=))
(1000, tensor(1.0384, grad_fn=))
(2000, tensor(0.7781, grad_fn=))
(3000, tensor(1.1698, grad_fn=))

In [13]:

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

```
/home/ec2-user/anaconda3/envs/pytorch_p27/lib/python2.7/site-packages/ipykernel/__
main__.py:7: UserWarning: volatile was removed and now has no effect. Use `with to
rch.no_grad():` instead.

Accuracy of Test Data: 61.4099998474
```

(6) Adam optimizer

(0, tensor(2.2960, grad_fn=))
(1000, tensor(1.4230, grad_fn=))
(2000, tensor(1.0869, grad_fn=))
(3000, tensor(0.7181, grad_fn=))

In [20]:

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

```
/home/ec2-user/anaconda3/envs/pytorch_p27/lib/python2.7/site-packages/ipykernel/__
main__.py:7: UserWarning: volatile was removed and now has no effect. Use `with to
rch.no_grad():` instead.

Accuracy of Test Data: 60.2000007629
```

(7) learning rate decay

In [ ]:

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

```
/home/ec2-user/anaconda3/envs/pytorch_p27/lib/python2.7/site-packages/ipykernel/__
main__.py:7: UserWarning: volatile was removed and now has no effect. Use `with to
rch.no_grad():` instead.

Accuracy of Test Data: 65.1299972534
```

학습된 파라미터 저장

In [ ]:

```
case = 7
```

In [ ]:

```
netname = './nets/mlp_weight0'+str(case)+'.pkl'
torch.save(model, netname, )
```

```
/home/ec2-user/anaconda3/envs/pytorch_p27/lib/python2.7/site-packages/torch/serial
ization.py:360: UserWarning: Couldn't retrieve source code for container of type C
NN. It won't be checked for correctness upon loading.
  "type " + obj.__name__ + ". It won't be checked "
```

저장된 파라미터 로드

In [ ]:

```
#lcase = # case 번호 입력
#lnetname = './net/mynet0%d.pkl' %lcase
#model = torch.load(lnatname)
```