PyTorch

PyTorch를 이용한 딥러닝 기초 실습

충남대학교 메카트로닉스공학과
지능차량시스템실험실 홍효성

https://github.com/peytonhong/pytorch_cnu

충남대학교 CHUNGNAM NATIONAL UNIVERSITY

목차

- MNIST Dataset 소개
- PyTorch 코드의 필수 구성 요소
- 코딩 실습
  - Multi Layer Perceptron (MLP)를 이용한 이미지 분류 예제 실습
  - Convolutional Neural Network (CNN)을 이용한 이미지 분류 예제 실습

MNIST Dataset

- Handwritten 숫자 이미지 데이터셋
  - 0~9까지의 숫자가 하나씩 그려진 28x28 픽셀 크기의 이미지 데이터 셋
  - 학습용 데이터: 60,000장
  - 검증용 데이터: 10,000장

LeCun et al. (1999): The MNIST Dataset Of Handwritten Digits (Images)

PyTorch 코드의 필수 구성 요소

- **DataLoader**: 사용할 데이터셋 로드
- **Model**: 네트워크 모델 정의 (Class 형태로 구현)
- **Optimizer**: SGD, Adam optimizer 등 정의 (learning rate)
- **Loss**: MSE, Negative Log Likelihood, Cross Entropy 등
- **For loop**
  - **Train()**: 학습 수행 ⟶

1. **Model.train()** : 모델을 학습 모드로 전환 (BN, Dropout 등 작동)
2. **Optimizer.zero_grad()** : Gradient 버퍼 0으로 초기화
3. **Outputs = Model(inputs)** : 모델 실행
4. **Loss = Loss_func(Outputs, Labels)** : Loss 계산
5. **Loss.backward()** : Gradient 자동 계산
6. **Optimizer.step()** : Gradient 적용하여 Weight 업데이트

  - **Test()**: 검증 수행 ⟶

1. **Model.eval()** : 모델을 검증 모드로 전환 (BN, Dropout 등 미작동)
2. **Outputs = Model(inputs)** : 모델 실행
3. **Loss = Loss_func(Outputs, Labels)** : Loss 계산

DataLoader

- Dataset을 관리하는 기능을 갖는 객체

```
DataLoader(dataset, batch_size=1, shuffle=False, sampler=None,
           batch_sampler=None, num_workers=0, collate_fn=None,
           pin_memory=False, drop_last=False, timeout=0,
           worker_init_fn=None)
```

  - Dataset: 사용할 Dataset의 정보를 담고 있는 객체 입력
  - Batch_size: 한 번에 학습시킬 데이터 개수 정의
  - Shuffle: Dataset에서 Batch_size 개수만큼 샘플링 할 때 랜덤 샘플링 여부 정의
  - Num_workers: Multi-process data loading시 사용 (잘 모르는 경우 0으로 설정하기 권장)

- 활용 예시:

```
train_loader = torch.utils.data.DataLoader(dataset=mnist_train, batch_size=64, shuffle=True)
test_loader = torch.utils.data.DataLoader(dataset=mnist_test, batch_size=1000, shuffle=True)
```

https://pytorch.org/docs/stable/data.html

Model 정의

- Pytorch의 **nn.Module**을 상속받는 클래스 형태로 정의함

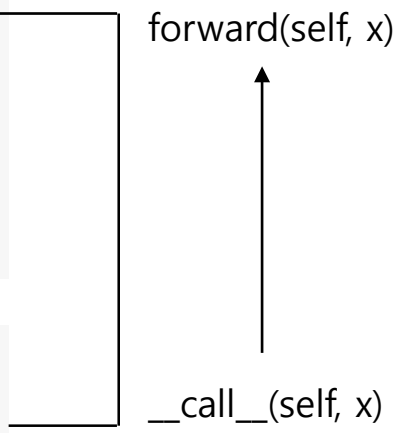
```
class Net_name(nn.Module):
    def __init__(self):
        super(Net_name, self).__init__()
        사용할 Layer 선언
    def forward(self, x):
        Layer 순서대로 작성
        return output
```

클래스 형태로 구성된 네트워크 모델의 구조

```
class MLP_Model(nn.Module):
    def __init__(self):
        super(MLP_Model, self).__init__()
        self.fc1 = nn.Linear(784, 128)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        x = torch.flatten(x, 1)
        x = self.fc1(x)
        x = F.relu(x)
        x = self.fc2(x)
        output = F.log_softmax(x, dim=1)
        return output

model = MLP_Model()
output = model(input_data)
```

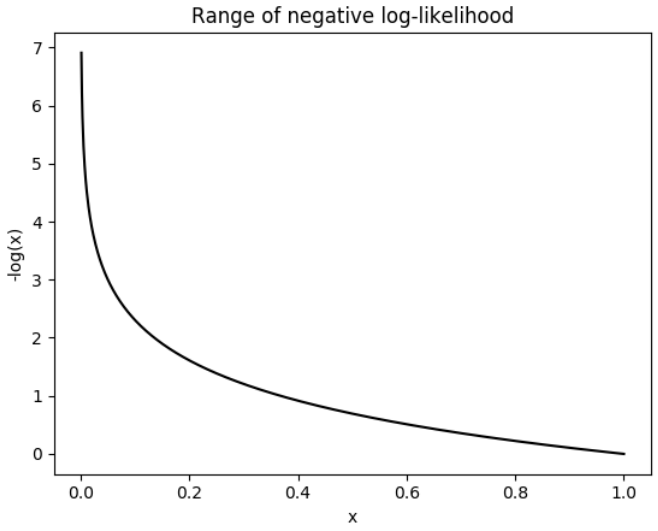


실제 코딩 예시 (MLP 네트워크)

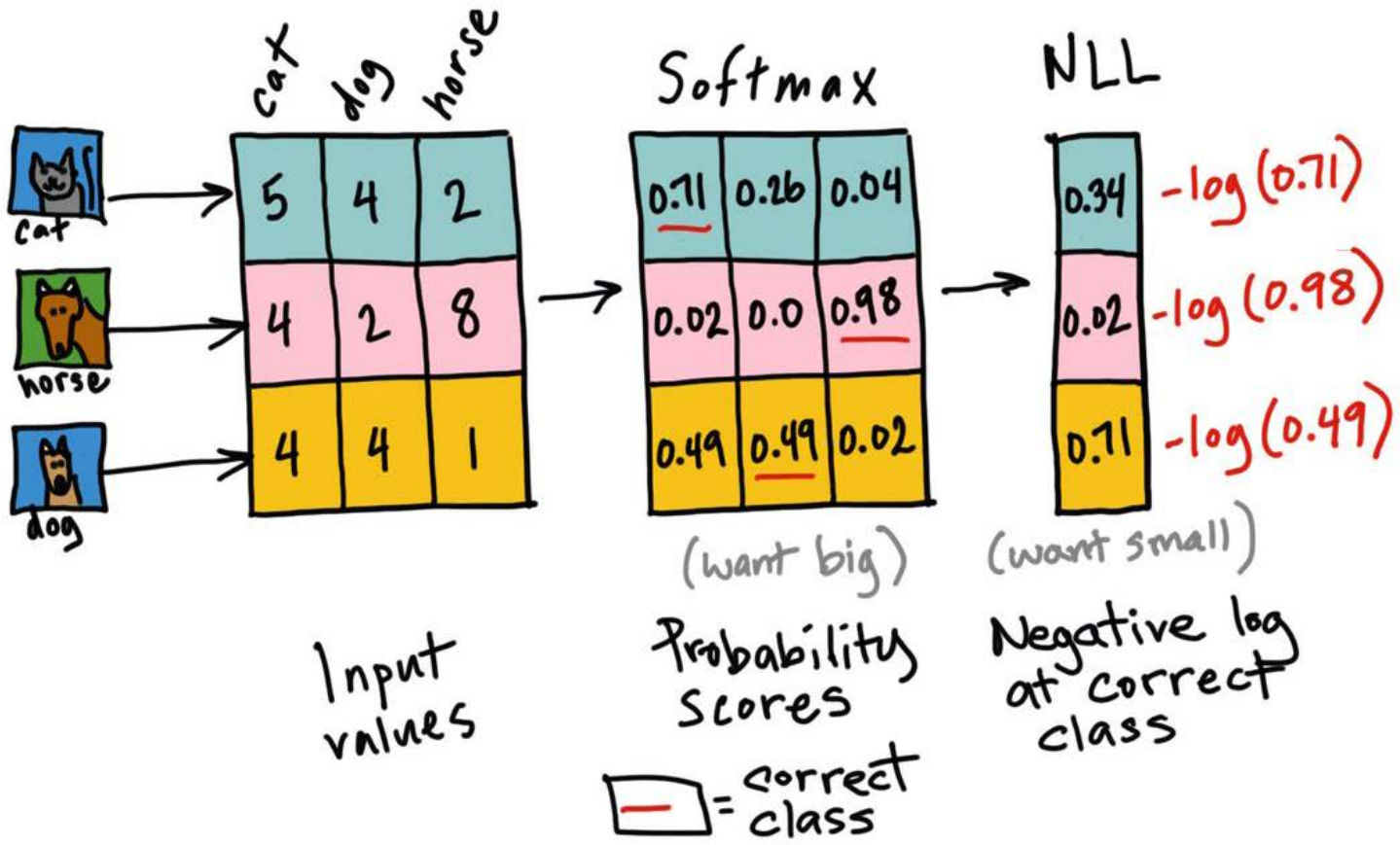
Loss 정의 (Negative Log Likelihood)

$$S(f_{y_i}) = \frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$$

<Softmax 수식>



-log(x) 그래프

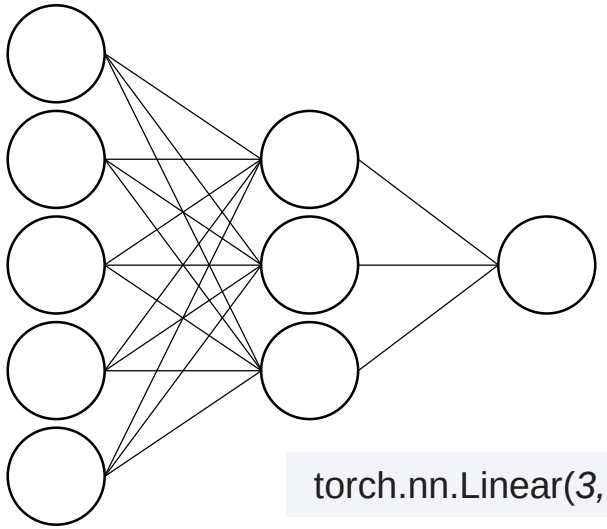


https://ljvmiranda921.github.io/notebook/2017/08/13/softmax-and-the-negative-log-likelihood/

MNIST 예제 실습

- MLP (Fully Connected)

torch.nn.Linear(*in_features*, *out_features*, *bias=True*)

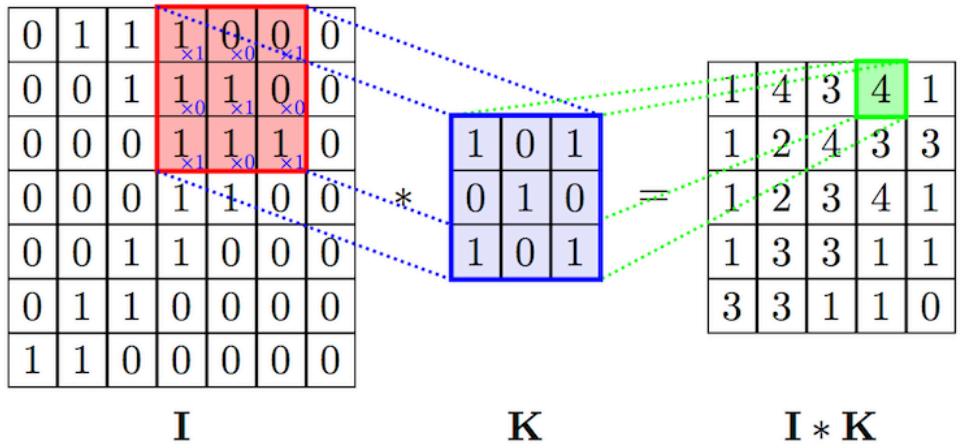


torch.nn.Linear(*3, 1*)

torch.nn.Linear(*5, 3*)

- CNN

torch.nn.Conv2d(*in_channels*, *out_channels*, *kernel_size*, *stride=1*, *padding=0*, *dilation=1*, *groups=1*, *bias=True*, *padding_mode='zeros'*)



torch.nn.Conv2d(1, 32, 3, 1)

https://tex.stackexchange.com/questions/437007/drawing-a-convolution-with-tikz?rq=1