

Deep Learning을 이용한 이미지 인식

CONTENTS

Objectives and Members

Activities by week by week

week1. pytorch 기초 문법

week2. 선형, 로지스틱, 소프트맥스 회귀

week3. 인공 신경망(퍼셉트론과 역전파)

week4. 다층 퍼셉트론을 이용한 MNIST 분류

week5. 합성곱과 풀링

week6. CNN을 이용한 MNIST분류

Outcomes and Future Plans

Objectives and Members

Objectives

겨울방학에 진행한 딥러닝 이론 스터디의 내용을 구현하는 것이 목표이다.

1. Python, Pytorch를 사용하여 다양한 활성화 함수와 회귀를 구현한다.
2. 역전파와 매개변수 최적화를 사용하여 학습 정확도를 향상시킨다.
 - 매개변수 최적화 : SGD, Momentum, AdaGrad
3. MNIST dataset을 이용해 training과 test를 진행하고 직접 구현한 신경망의 성능을 확인한다.
4. 다른 오픈소스의 학습 모델과 비교하여 직접 구현한 신경망의 성능을 확인하고 향상 시킨다.



PYTHON



PYTORCH

Members

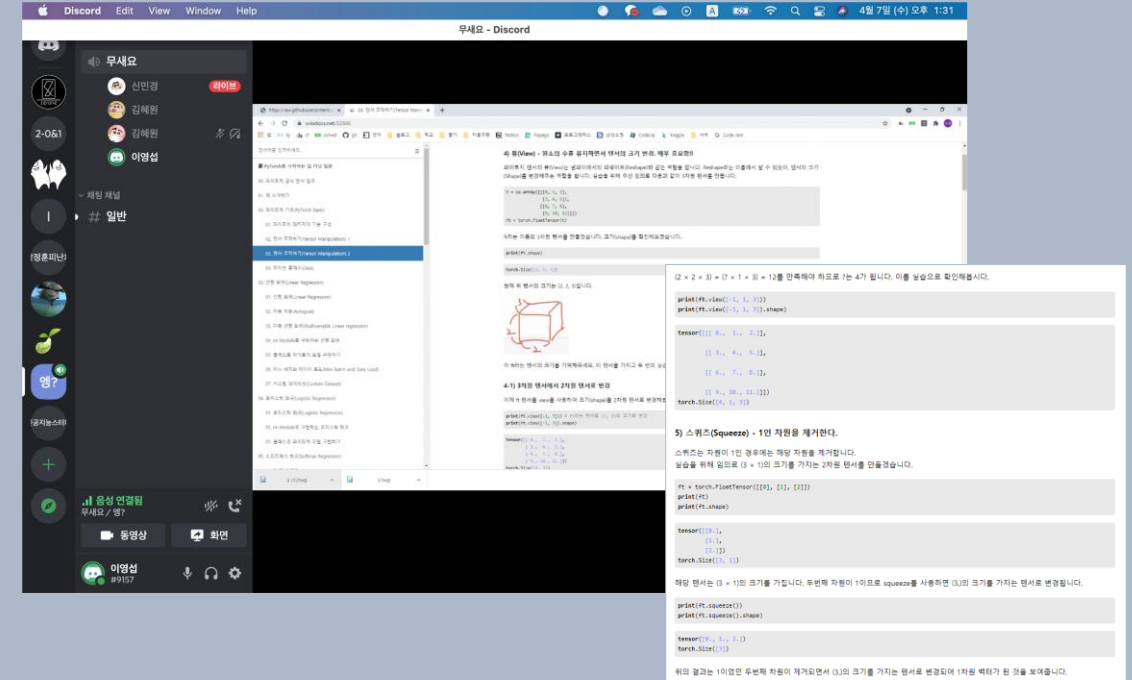
2019028313 - 김혜원

2019037129 - 신민경

2017012351 - 이영섭

Activities by week by week

Week 1. pytorch 기초 문법



Activities by week by week

Week 1. pytorch 기초 문법

week2. 선형, 로지스틱, 소프트맥스 회귀

week3. 인공 신경망(퍼셉트론과 역전파)

week4. 다층 퍼셉트론을 이용한 MNIST 분류

week5. 합성곱과 풀링

week6. CNN을 이용한 MNIST분류

Discord chat window showing a list of members: 김혜원, 김혜원, 신민경, 이영섭.

Discord channel window showing a code editor with the following Python code:

```
def softmax(a):  
    c = np.max(a) # a의 최대값  
    exp_a = np.exp(a-c)  
    sum_exp_a = np.sum(exp_a)  
    y = exp_a / sum_exp_a  
    return y
```

Slide 4. 출력층 설계하기 (softmax function):

$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^n \exp(a_i)}$$

Slide 4. 출력층 설계하기 (softmax function):

$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^n \exp(a_i)}$$

1. 선형 함수와 비선형 함수의 차이 파악
2. 다양한 연산 게이트의 구현
3. 계단함수, 시그모이드 함수, ReLU 함수 구현 및 차이 비교
4. 소프트맥스 함수의 구현 및 적용

Activities by week by week

Week 1. pytorch 기초 문법

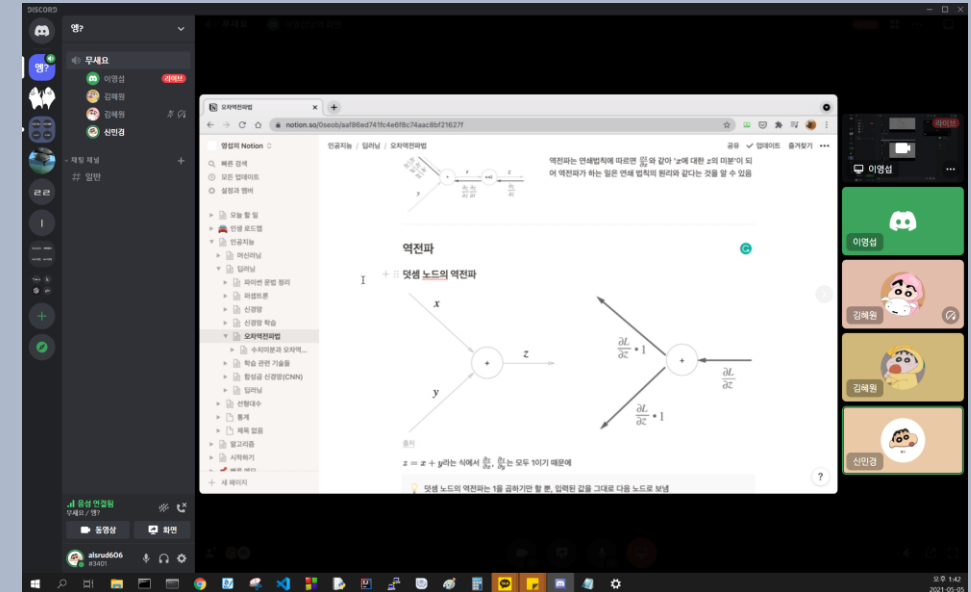
week2. 선형, 로지스틱, 소프트맥스 회귀

week3. 인공 신경망(퍼셉트론과 역전파)

week4. 다층 퍼셉트론을 이용한 MNIST 분류

week5. 합성곱과 풀링

week6. CNN을 이용한 MNIST분류



1. 신경망과 퍼셉트론의 차이 파악
2. 손실 함수에 대한 이해 및 설정과 계산법에 대하여 학습
3. 역전파의 개념을 이해하고 여러 함수의 계산그래프를 학습
4. 수치 미분과 오차역전파법의 공통점과 차이점을 비교

Activities by week by week

Week 1. pytorch 기초 문법

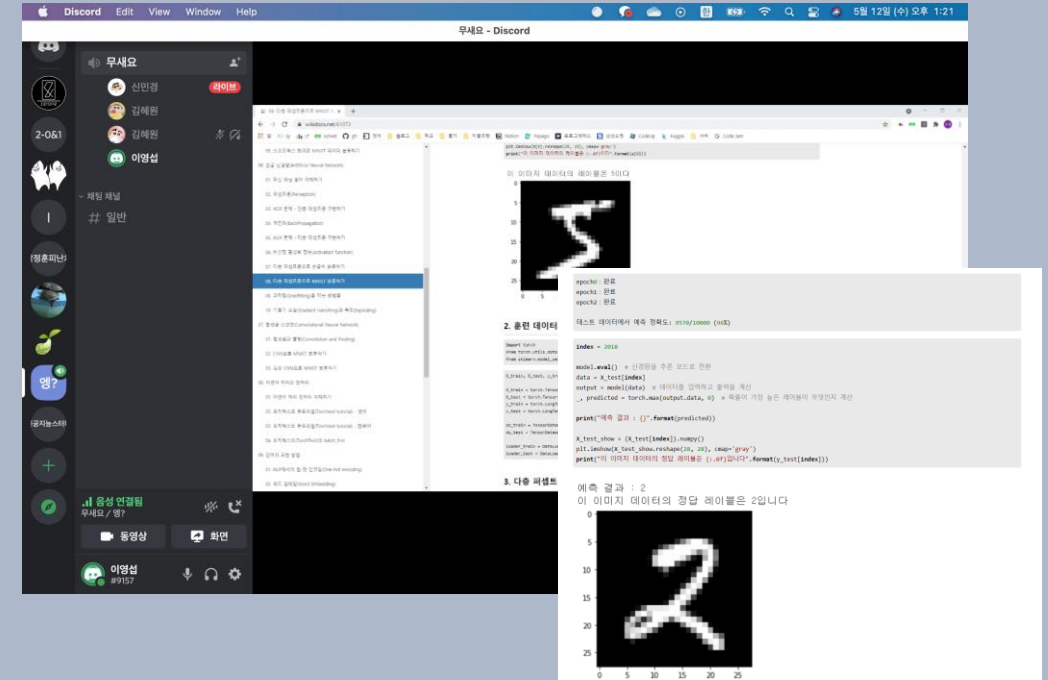
week2. 선형, 로지스틱, 소프트맥스 회귀

week3. 인공 신경망(퍼셉트론과 역전파)

week4. 다층 퍼셉트론을 이용한 MNIST 분류

week5. 합성곱과 풀링

week6. CNN을 이용한 MNIST분류



1. 입력층과 출력 층 사이에 은닉층을 추가한 다층 퍼셉트론 구현
2. 구현한 신경망 모델을 이용한 MNIST데이터 학습 및 분류
3. overfitting을 막기위한 dropout 및 가중치 규제 공부 및 구현

Activities by week by week

Week 1. pytorch 기초 문법

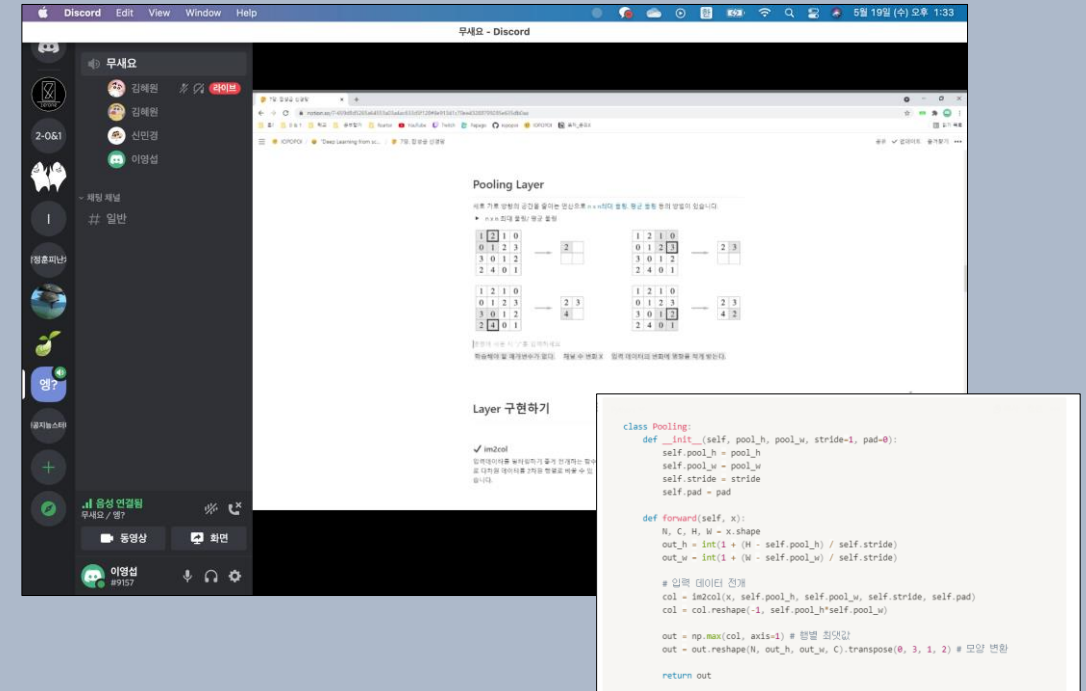
week2. 선형, 로지스틱, 소프트맥스 회귀

week3. 인공 신경망(퍼셉트론과 역전파)

week4. 다층 퍼셉트론을 이용한 MNIST 분류

week5. 합성곱과 풀링

week6. CNN을 이용한 MNIST분류



무새요 - Discord

Pooling Layer

이웃 개체들의 공집합을 취하는 연산으로 $n \times n$ 의 행렬 혹은 행렬들의 집합이 있습니다.

→ $n \times n$ 크기의 행렬 / 행렬 집합

Layer 구현하기

✓ `nn.MaxPool2d`

입력데이터를 배치단위로 끊어 내려가는 방식
로 입력층 데이터들 크기에 맞추어 사용할 수
있습니다.

```
class Pooling:
    def __init__(self, pool_h, pool_w, stride=1, pad=0):
        self.pool_h = pool_h
        self.pool_w = pool_w
        self.stride = stride
        self.pad = pad

    def forward(self, x):
        N, C, H, W = x.shape
        out_h = int((H - self.pool_h) / self.stride)
        out_w = int((W - self.pool_w) / self.stride)

        # 입력 데이터 전개
        col = nn.unfold(x, self.pool_h, self.pool_w, self.stride, self.pad)
        col = col.reshape(-1, self.pool_h * self.pool_w)

        out = np.max(col, axis=-1) # 행렬 최대값
        out = out.reshape(N, out_h, out_w, C).transpose(0, 3, 1, 2) # 모양 변환

        return out
```

1. Convolution layer, Pooling layer의 작동원리 및 기능 공부
2. Convolutin, Pooling layer의 구현 및 활용 방안 공부
3. 구현한 계층을 활용한 간단한 CNN 구현 및 학습

Activities by week by week

Week 1. pytorch 기초 문법

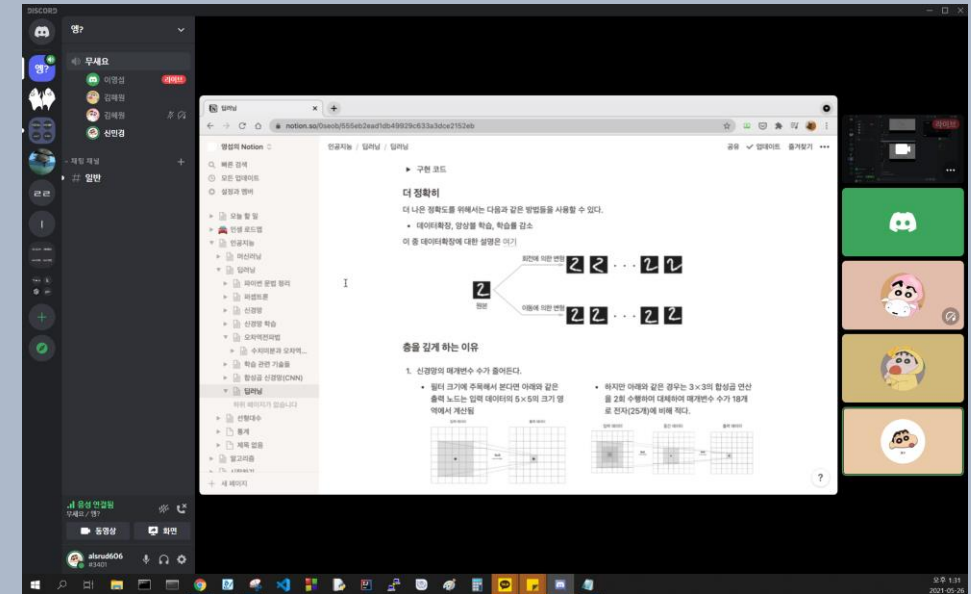
week2. 선형, 로지스틱, 소프트맥스 회귀

week3. 인공 신경망(퍼셉트론과 역전파)

week4. 다층 퍼셉트론을 이용한 MNIST 분류

week5. 합성곱과 풀링

week6. CNN을 이용한 MNIST분류



1. Convolution layer와 Pooling layer를 이용하여 복잡한 CNN 구현
2. MNIST 데이터셋 분류의 정확도 향상을 위한 방법을 공부
 1. 활성화 함수로 ReLU를 사용
 2. He 초기값으로 초기값 설정

Outcomes and Feature plans

Outcomes

```
class Relu:
    def __init__(self):
        self.mask = None

    def forward(self, x):
        self.mask = (x <= 0) # 입력값이 0이하이면
        out = x.copy()
        out[self.mask] = 0

        return out

    def backward(self, dout):
        dout[self.mask] = 0 # 순전파에서 입력이 0이하면 입력을
        dx = dout
        return dx

# cnn을 구성하는 계층을 생성한다. [conv - relu - pool - affine]

# 계층 생성
self.layers = OrderedDict()
self.layers['Conv1'] = Convolution(self.params['W1'], self.params['b1'],
                                   conv_param['stride'], conv_param['pool'])

self.layers['Relu1'] = Relu()
self.layers['Pool1'] = Pooling(pool_h=2, pool_w=2, stride=2)
self.layers['Affine1'] = Affine(self.params['W2'], self.params['b2'])
self.layers['Relu2'] = Relu()
self.layers['Affine2'] = Affine(self.params['W3'], self.params['b3'])

self.last_layer = SoftmaxWithLoss()
```

```
w = np.random.randn(node_num, node_num) * np.sqrt(2.0 / node_num)
```

```
class Dropout:
    def __init__(self, dropout_ratio=0.5):
        self.dropout_ratio = dropout_ratio
        self.mask = None

    def forward(self, x, train_flg = True):
        if train_flg:
            self_mask = np.random.rand(*x.shape) > self.dropout_ratio
            return x * self_mask
        else:
            return x * (1.0 - self.dropout_ratio)

    def backward(self, dout):
        return dout * self_mask
```

```
test accuracy : 0.113319
test accuracy : 0.747621
test accuracy : 0.974833
```

다양한 활성화 함수와 회귀를 구현하고 매개변수 최적화, 역전파 등을 통해 신경망 (인공지능)의 성능을 향상시켰다.

Feature plans

이번 학기에 구현한 신경망을 기반으로 더 다양한 데이터를 training 및 test하여 더 다양한 데이터를 분류할 수 있는 신경망을 만드는 것을 목표로 한다.

1. 인터넷을 통해 더 다양한 학습데이터를 구축하고 신경망을 학습시킨다.
2. 학습결과를 활용한 프로그램을 개발한다.
 - 사물을 인식 및 분류 할 수 있는 프로그램을 개발한다.
 - 더욱 정확하게 사람의 얼굴을 학습하고 구분할 수 있는 신경망을 구축하기 위해 신경망을 개선한다.
3. 다양한 오픈소스를 활용 및 참조하여 더욱 개선된 인공지능 프로그램을 만든다.

Deep Learning from scratch & Pytorch

End of Document;