밑바닥 부터 시작하는 딥러닝1

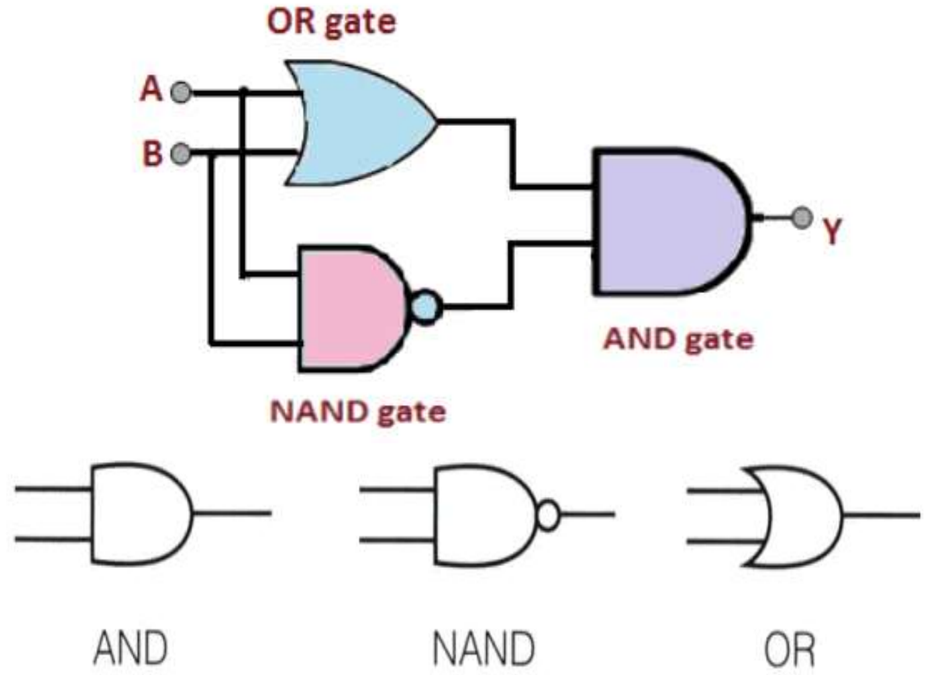
3.6장 MNIST 데이터셋

학습자: 이관형

목차

2 장 퍼 셉 트 론 문 제 해 결

		NAND	OR	AND
x_1	x_2	s_1	s_2	y
0	0	1	0	0
1	0	1	1	1
0	1	1	1	1
1	1	0	1	0



경우의 수(OR과 NAND의 순서를 바꾸는건 됨)
1. (AND, NAND) - > (OR)일때
2. (AND, OR) -> (NAND)일때
3. (OR, NAND) -> (AND) -> 이건 XOR구현의 기본 원리
따라서 3가지 중 2가지만 보면 됨.

경우의 수: 3 * 2 * 1(비복원 추출)

but 은닉층을 담당하는 게이트 두개는 순서 바뀌어도 상관 없음. 따라서 3가지 경우의 수만 발생(중복 제거)

2 장 퍼 셉 트 론 문 제 해 결

```
for i in np.array([[0, 0],
                   [1, 0],
                   [0, 1],
                   [1, 1]]):
    y=XOR(i[0], i[1])
    print(str(i), '->', str(y))
```

```
[0 0] -> 1
[1 0] -> 1
[0 1] -> 1
[1 1] -> 1
```

경우의 수(OR과 NAND의 순서를 바꾸는건 됨)
1. (AND, NAND) - > (OR)일때
2. (AND, OR) -> (NAND)일때
3. (OR, NAND) -> (AND) -> 이건 XOR구현의 기본 원리
따라서 3가지 중 2가지만 보면 됨.

```
    for i in np.array([[0, 0],
                       [1, 0],
                       [0, 1],
                       [1, 1]]):
        y=XOR(i[0], i[1])
        print(str(i), '->', str(y))
```

```
[0 0] -> 1
[1 0] -> 1
[0 1] -> 1
[1 1] -> 0
```

아니면 실제로 3게이트를
만족하는 가중치 편향이 모두
다름 -> 바꿔도 될 수가 없음

MNIST 데이터

데이터 셋 파악

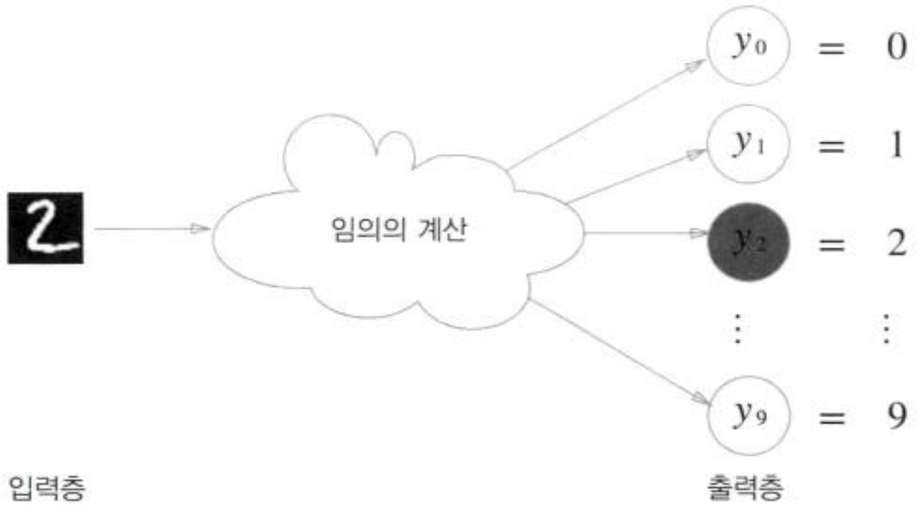
시각화

신경망 구현

배치 처리

MNIST 데이터

그림 3-23 출력층의 뉴런은 각 숫자에 대응한다.



신경망 분류 문제

이미지 행렬
컬러: 4차원(rgb)
흑백: 3차원

훈련: 60,000 테스트:10,000
흑백 한장 28 * 28* 1 행렬

총 4차원: 60,000 *28*28*1
(train 경우)
출력층 확률

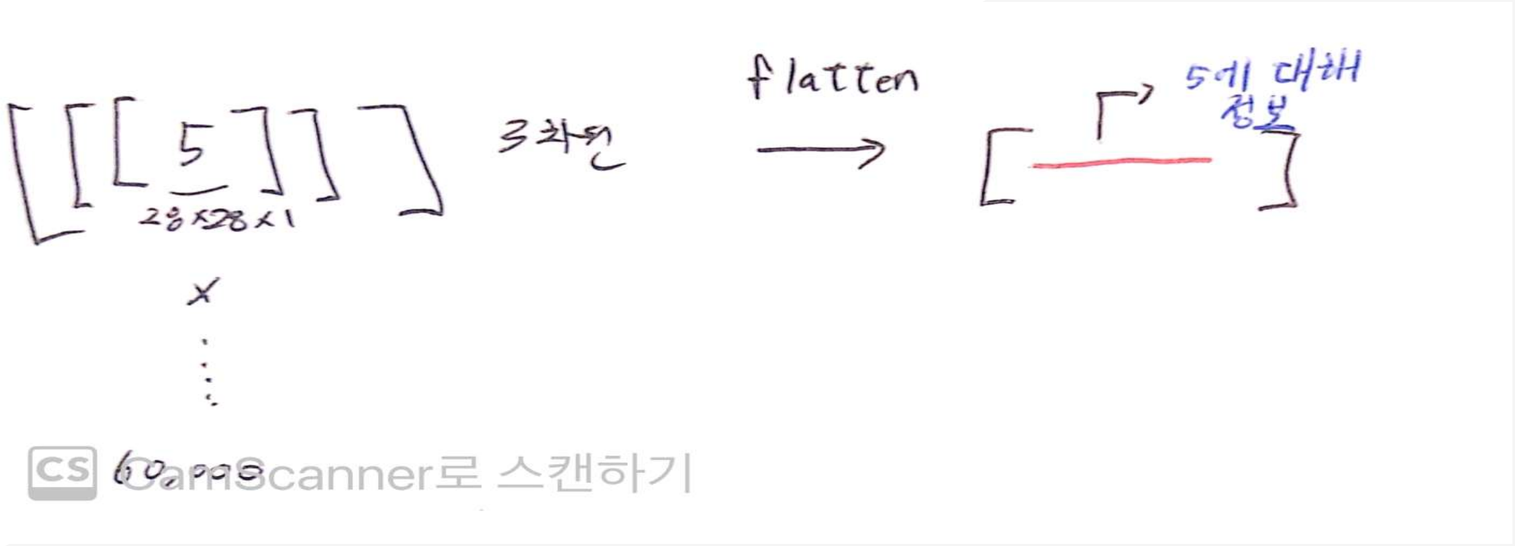
MNIST 데이터

인자 3개

```
(x_trian, t_train), (t_test, t_test) = ##

load_mnist(flatten= False, normalize = False, one_hot_label= False)
```

MNIST인자 탐색

```
(x_trian, t_train), (t_test, t_test) = ㅐ
load_mnist(flatten= False, normalize = False, one_hot_label= False)
```



인자 3개

flatten -> 2차원 차원 축소

one_got_label -> 확률값으로 변환

신경망 마지막 출력

[[[5]]] -> 0~9까지의 확률값으로 반환됨(소프트맥스)

label(정답): ex)5이런식 이것을 변환하여 확률값 5인 확률이 1인 확률값이 됨

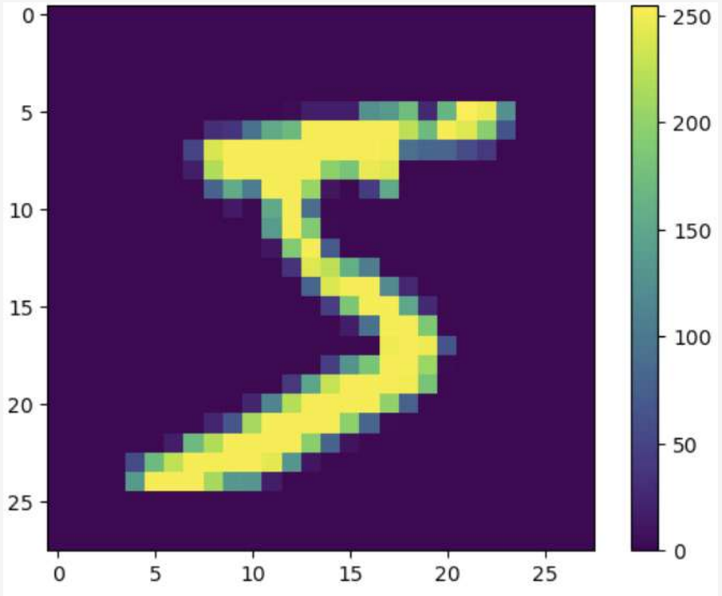
MNIST 시각화

PIL 라이브러리

```
from PIL import Image #이번 시간에 사용하고 안씀, 0

def img_show(img):
    pil_img = Image.fromarray(np.uint8(img))
    pil_img.show()
```

matplotlib 라이브러리

```
(x_trian, t_train), (t_test, t_test) = load_mnist(flatten= False, normalize = False)

plt.imshow(x_trian[0][0])
plt.colorbar() #옆 옆에 컬러바가 떠버리네
plt.show()
```



신 경 망 의　추 론　처 리

key point !

학습을 구현하는 것이 아님!
우리가 구해야할 변수는 w, b but 여기서
불러옴 학습 w, b를 신경망 구현

```python
def init_network():
    with open("sample_weight.pkl", 'rb') as f:
        # 학습된 가중치 매개변수가 담긴 파일
        # 학습 없이 바로 추론을 수행
        network = pickle.load(f)

    return network
```

입력: 784(28*28) -> 은닉층-> 출력 (10) 확률값

```python
def predict(network, x):
    w1, w2, w3 = network['W1'], network['W2'], network['W3']
    b1, b2, b3 = network['b1'], network['b2'], network['b3']

    a1 = np.dot(x, w1) + b1
    z1 = sigmoid(a1)
    a2 = np.dot(z1, w2) + b2
    z2 = sigmoid(a2)
    a3 = np.dot(z2, w3) + b3
    y = softmax(a3)

    return y #10가지의 추론 값들 확률
```

accuracy 코드 구현

```
x, t= get_data() #테스트 데이터
network = init_network()
```

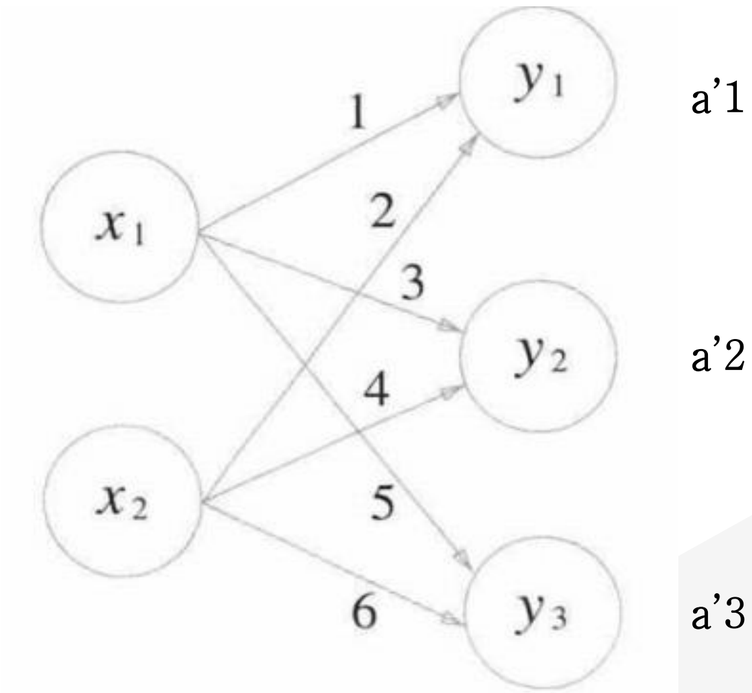
```
accuracy_cnt = 0

for i in range(len(x)):
    y = predict(network, x[i])
    p = np.argmax(y) #각각 10가지 중 확률 값이 나오는

    if p == t[i]:
        accuracy_cnt += 1

print('정확도:', float(accuracy_cnt / len(x)))
```

```
정확도: 0.9352
```

len(x) -> 테스터 데이터 수 10,000(리스트 별 파악)

np.argmax -> 인덱스 반환
ex) 확률 : [0.05, 0.05, 0.05, 0.05, 0.05, 0.05,0.05,0.05,0.05, 0.55] ->index: 9 ->예측한 라벨값

배 치　처 리



ex) 한번에 두장

[[a1 a2 a3],
[a'1, a'2, a'3]] = [[x1, x2],　(w11, w12, w13,)　+　(b1, b2, b3)
[x'1, x'2]]　w21, w22, w23　b1, b2, b3

브로드 캐스트

배 치　　처 리

	0	1	2	3	4	5	6	7	8	9	...	775	776	777	778	779	780	781	782	783	target
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	7
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2
9996	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3
9997	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4
9998	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5
9999	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	6

x, t 데이터 프레임

배치처리

```
accuracy_cnt = 0
for i in range(0, len(x), 100):
    x_batch = x[i:i+100]

    y_batch = predict(network, x_batch)

    p = np.argmax(y_batch, axis = 1)
    accuracy_cnt += np.sum(p == t[i:i+100])


print( 'acc: ' + str(float(accuracy_cnt) / len(x)))
```

```
acc: 0.9352
```