머리말

인공지능, 머신러닝은 많이 들어본 단어이지만 초보자에게는 쉽게 이해하기 힘든 내용이었다. 어떻게 하면 쉽게 머신러닝의 원리를 이해 할 수 있을까 해서 이 책을 쓰게 되었다. 많은 내용을 다루는 것보다 핵심적인 내용을 단기간에 배울 수 있도록 구성하였다. 구체적으로는 MNIST 손글씨 데이터를 인식하는 알고리즘을 집중적으로 소개할 것이다. 좀 더 자세하고 깊이 있는 내용은 관련 참고문헌을 찾아서 학습하면 좋겠다. 이 책은 고려대학교 BK21 PLUS 사업의 지원을 받아서 집필되었다.

차 례

장 1

MATLAB 기초

이 장에서는 본문에서 주로 사용하는 MATLAB 기본 표현 및 함수들을 소개하겠다.

제 1 절 연산자

기본 연산자

기본 연산자로는 단순 계산은 물론, 벡터와 행렬에서 모두 적용 가능한 더하기 (+), 빼기 (−), 곱하기 (∗), 나누기 (/)가 있다.

`A + B`	A에 B를 더하다
`A - B`	A에서 B를 빼다
`A * B`	A에 B를 곱하다
`A / B`	A를 B로 나누다

```
>> x = [1 2 3 4 5]; y = [5 4 3 2 1];
>> x < y
  ans =  1     1     0     0     0
>> x <= y
  ans =  1     1     1     0     0
>> x == y
  ans =  0     0     1     0     0
>> x >= y
  ans =  0     0     1     1     1
>> x > y
  ans =  0     0     0     1     1
```

논리연산자

다양한 논리연산자 중 자주 사용하는 몇 가지에 대해서 알아보도록 하자.

`A & B`	A와 B가 둘 다 동시에 참일 때만 참이고, 그 외에는 모두 거짓
`A | B`	A 또는 B 중 적어도 하나만 참이면 참이고, 그 외에는 거짓

if else 문

여러 가지 조건에 따라 각각 다른 명령을 실행하고자 할 때, 'if ∼ else ∼ end'문을 사용한다. 아래 보기처럼 **조건 1**이 참이면 **문장 1**이 수행되고, **조건 1**이 모두 거짓이면, 'if'문을 빠져 나와 **문장 2**가 수행된다.

```
a=3;
if a<1
    b=a+1
else
    c=a+2
end
```

위의 코드는 a의 초깃값을 3으로 지정하게 된다. if 구문을 살펴보면 처음 조건은 a가 1보다 작을 경우에 b값을 a+1로 출력하게 되며, 그렇지 않을 경우 c값을 a+2로 출력하게 되는 것을 알 수 있다. 따라서 a=3은 첫 번째 조건을 만족하지 않으므로 결과 값으로 c=5값을 출력하는 것을 알 수 있다.

```
c = 5
```

while 문

'while'문은 'end'문과 짝을 이루어 사용된다. 'while'문과 같은 행에 있는 조건이 참이면 'while' 문과 'end'문 사이에 있는 문장의 명령을 반복적으로 수행한다.

```
a=1; b=2, c=3;
b=2
```

명령문 연속 ...

...은 줄을 넘겨주는 역할을 한다. 명령어가 너무 길어 다음 줄로 넘기고 싶을 때 사용한다. 편집기에서뿐만 아니라 작업창에서도 사용가능하다. 주의 할 점은 ' '의 중간을 ...으로 줄 바꿈을 할 경우 오류가 발생한다. 따라서, ' '다음에 ...을 사용해야한다.

```
plot(x,y,'--rs','LineWidth',2,'MarkerEdgeColor','k', ...
      'MarkerSize',10)
```

inline

inline을 이용하면 간편하게 함수를 정의할 수 있다.

```
f = inline('x^3+6*x-2','x');
f(3)
ans = 43
```

- `plot(X,Y,S)`

  S는 선의 종류, 심볼(symbol) 또는 색을 나타낼 수 있는 옵션값이다.(자세한 내용은 표 1.1 참고.)

- `plot(X1,Y1,S1,X2,Y2,S2,X3,Y3,S3,...)`

  여러 개의 벡터를 한 번에 같이 나타낼 수 있다.

색상		모양		라인	
b	Blue	.	Point	-	Solid
g	Green	o	Circle	:	Dotted
r	Red	x	x mark	-.	Dashdot
c	Cyan	+	Plus	--	Dashed
m	Magenta	*	Star	(none)	No line
y	Yellow	s	Square		
k	Black	d	Diamond		
w	white	v	Triangle(down)		
		^	Triangle(up)		

표 1.1: plot 명령어의 옵션

예를 들어, `plot(x,sin(x),'k--',x,cos(x),'ko')`를 실행하면, 그림 1.1을 얻게 된다. 이는 y 축의 값을 $\sin(x)$, $\cos(x)$로 하는 두 개의 그래프를 나타내며, 첫 번째 $\sin(x)$는 검은색 점선으로, $\cos(x)$는 검은색 원으로 표현된다.
다음은 부가적인 명령어이다.

- `title`

  그래프의 제목 넣기

- `xlabel, ylabel`

  x축에 이름 넣기, y축에 이름 넣기

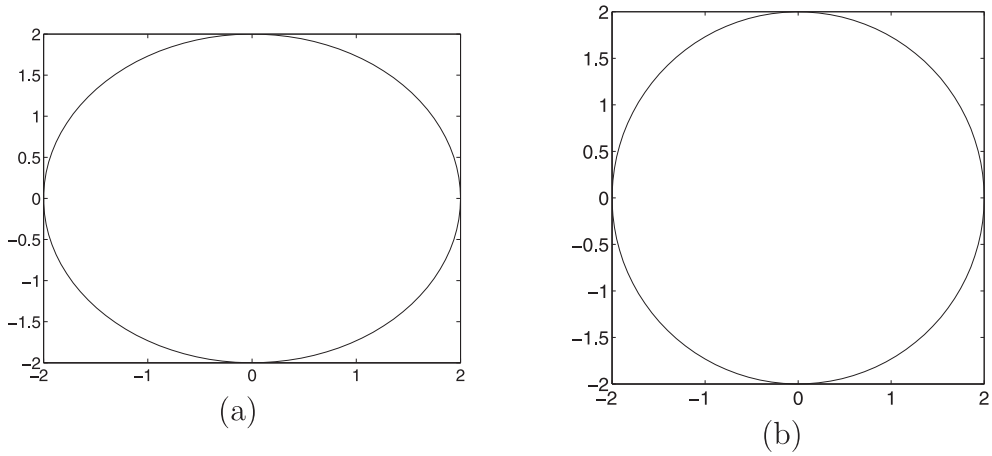

그림 1.2: (a) 좌표계가 default인 경우. (b) 좌표계가 square인 경우.

Aspect ratio의 옵션을 "square"로 하면, Figure window의 모양을 직사각형 모양의 좌표계가 아닌 정사각형의 모양의 좌표계로 바꾼다. 그림 1.2(b)는 그림 1.2(a)보다 훨씬 더 원하는 원의 모양을 갖고 있다. 그러나 이것은 어디까지나 Figure window를 정사각형 모양으로 바꾸어 준 결과로 나타난 것이다. 결국 각 좌표축의 units는 무시한 결과이다.

```
t=linspace(0,2*pi,100); x=2*cos(t); y=2*sin(t);
plot(x,y); axis square;
```

만일 실제 사물의 크기 비율(aspect ratio)와 같게 하려면, 각각의 좌표축에 대한 units는 동등한 크기를 가져야 한다. 이때 이용되는 aspect ratio로는 "equal"과 "image"가 있다. 해당 그래프에 대한 데이터의 범위까지만 display하는 경우가 "image"에 해당한다 (그림 1.3 참조).

zeros

각 원소 값들이 0인 정방행렬을 생성한다.

```
>> zeros(2)
ans = 0     0
      0     0
```

length

length로 벡터의 길이를 알 수 있다.

```
>> C=[1 2 3]; length(C)
ans = 3
```

sum

행렬 또는 벡터의 원소들의 합을 구할 수 있다. 벡터의 경우 전체를 더해주게 되며, 행렬의 경우 각 열의 합을 계산하게 된다.

```
fprintf(fp, '%e %e\n', 100, 1000);    %파일에 100 1000 쓰기
fclose(fp); %파일 close
```

```
1 2
3.500000 4.500000
1.000000e+002 1.000000e+003
```

load

load 함수는 파일에 저장된 데이터를 가져올 수 있다. 단, 파일에 문자가 들어 있으면 안 된다.

```
a = load('test.m');
```

```
a =  1.0e+003 *
    0.0010    0.0020
    0.0035    0.0045
    0.1000    1.0000
```

function

function은 임의의 함수를 정의할 때 사용하는 방법이다. 예를 들어, 시그모이드 함수 $y = \frac{1}{1+e^{-x}}$를 정의하고자 한다면 다음과 같은 코드의 M-file을 생성하면 된다.

```
function y=sigmoid(x)
y = 1.0./(1.0+exp(-x));
end
```

여기서 x는 함수의 독립변수를 의미한다. 주의해야 할 점은 위에서 함수의 이름('sigmoid')와 M-file의 이름('sigmoid.m')이 같아야 하며, 이 function을 사용할 때에는 function 파일이 저장되어 있는 디렉터리에서 사용이 가능하다. 저장된 function을 사용하는 방법은 다음과 같다.

```
x = linspace(-2,2,5);
sigmoid(x)


ans = 0.1192    0.2689    0.5000    0.7311    0.8808
```

장 2

머신러닝 (Machine Learning)

머신러닝(machine learning) 또는 기계학습은 인공지능의 한 분야로, 컴퓨터가 학습할 수 있도록 하는 알고리즘을 개발하는 분야를 말한다. 머신러닝의 구체적인 예제로 손글씨 숫자($0 \sim 9$)인식에 대해서 알아보겠다.

다음은 주요참고문헌 리스트이다.

[1] 기초부터 활용까지 패턴인식과 기계학습, 박혜영, 이관용, 이한출판사, 2011.

[2] 패턴인식, 오일석, 교보문고, 2008.

[3] 밑바닥부터 시작하는 딥러닝, 사이토 고키 저, 개앞맵시 역, 한빛미디어, 2017.

[4] 딥러닝 제대로 시작하기, 오카타니 타카유키 저, 심효섭 역, 제이펍, 2016.

[5] 신경망 첫걸음, 타리크 라시드 저, 송교석 역, 한빛미디어, 2017.

여기서 Δt는 시간간격이고 $w_i^{(n+1)}$은 n번째 반복했을때의 매개변수의 값이다. 특히, $w_i^{(1)}$은 초기 추측 값이다. 식 (2.4)의 양변에 Δt를 곱하고 다시 정리하면,

$$w_i^{(n+1)} = w_i^{(n)} - \Delta t \frac{\partial E(\mathbf{w}^{(n)})}{\partial w_i},\ i = 1, 2, \cdots, N,\ n = 1, 2, \cdots . \tag{2.5}$$

따라서, $\eta = \Delta t$ 라 하면 경사 하강법을 얻는다.

$$w_i^{(n+1)} = w_i^{(n)} - \eta \frac{\partial E(\mathbf{w}^{(n)})}{\partial w_i},\ i = 1, 2, \cdots, N,\ n = 1, 2, \cdots . \tag{2.6}$$

여기서 η는 학습률이라고 하며, 기울기의 반대 방향으로 얼마나 이동할지를 결정하는 역할을 한다. 이제 다음 예제를 통해, 학습률의 크기에 따라 어떤 현상이 일어나는지 알아보자.

예제1: 다음 주어진 비용함수 $E(w)$를 최소로 만드는 w를 구하는 문제에 대해서 학습률(η)을 크게 설정하여 경사 하강법 알고리즘을 작성해보자.

$$E(w) = 3(w-2)^2 + 5,$$
$$\eta = 0.4,\ w^{(1)} = 5.$$

초기 비용함수를 구해보면, $E(w^{(1)}) = 32$ 이다.

Step 1. 식 (2.6)을 적용하기 위해 $\dfrac{dE}{dw}$을 구하면 다음과 같다.

$$\frac{dE}{dw} = 6(w-2)$$

```
 w(i+1) = w(i) - eta*dE(w(i));
 fprintf('w(%d)=%f, E(w(%d))=%f \n',i+1,w(i+1),i+1,E(w(i+1)));
end
```

다음 결과를 보면 함숫값 $E(w^{(n)})$이 점점 커지는 것을 알 수 있다.

```
w(1)=5.000000, E(w(1))=32.000000
w(2)=-2.200000, E(w(2))=57.920000
w(3)=7.880000, E(w(3))=108.723200
w(4)=-6.232000, E(w(4))=208.297472
w(5)=13.524800, E(w(5))=403.463045
w(6)=-14.134720, E(w(6))=785.987568
w(7)=24.588608, E(w(7))=1535.735634
w(8)=-29.624051, E(w(8))=3005.241843
w(9)=46.273672, E(w(9))=5885.474012
w(10)=-59.983140, E(w(10))=11530.729064
w(11)=88.776396, E(w(11))=22595.428965
```

[그림 2.1](a)는 학습률이 클 경우 주어진 $w^{(1)}$에서의 비용함수의 기울기를 빗변으로 하고 밑변의 길이가 학습률 η인 직각삼각형을 생각해보자. 이때, 삼각형의 높이는 $\eta\dfrac{dE}{dw}\left(w^{(1)}\right)$이 된다. 따라서 $w^{(2)}$의 값은 현재값 $w^{(1)}$에서 $\eta\dfrac{dE}{dw}\left(w^{(1)}\right)$를 뺀 값이 된다. 이는 비용함수를 최소화하는 값으로부터 더 멀어지게 되고 이를 반복하면 [그림 2.1](b)처럼 점점 더 멀어진다. 따라서 $w^{(n)}$은

Step 2. 식 (2.6)을 10번 반복 적용하여 보자.

$$w^{(2)} = w^{(1)} - \eta \frac{dE}{dw}\left(w^{(1)}\right) = 5 - 0.01 \times 6 \times (5-2) = 4.82$$

비용함수를 구해보면, $E(w^{(2)}) = 28.8572$

$$w^{(3)} = w^{(2)} - \eta \frac{dE}{dw}\left(w^{(2)}\right) = 4.82 - 0.01 \times 6 \times (4.82-2) = 4.6508$$

비용함수를 구해보면, $E(w^{(3)}) = 26.080222$

$$\vdots$$

$$\begin{aligned} w^{(11)} &= w^{(10)} - \eta \frac{dE}{dw}\left(w^{(10)}\right) = 3.718984 - 0.01 \times 6 \times (3.718984-2) \\ &= 3.615845 \end{aligned}$$

비용함수를 구해보면, $E(w^{(11)}) = 12.832869$

학습률이 작을 경우를 다음 MATLAB 코드를 작성해서 알아보자.

```
%%% eta_small_book.m %%%
clear;clc
E = inline('3*(xx-2).^2+5','xx');
dE = inline('6*(xx-2)','xx');
eta = 0.01; w(1) = 5;
fprintf('w(1)=%f, E(w(1))=%f \n',w(1),E(w(1)));
for i=1:10
```

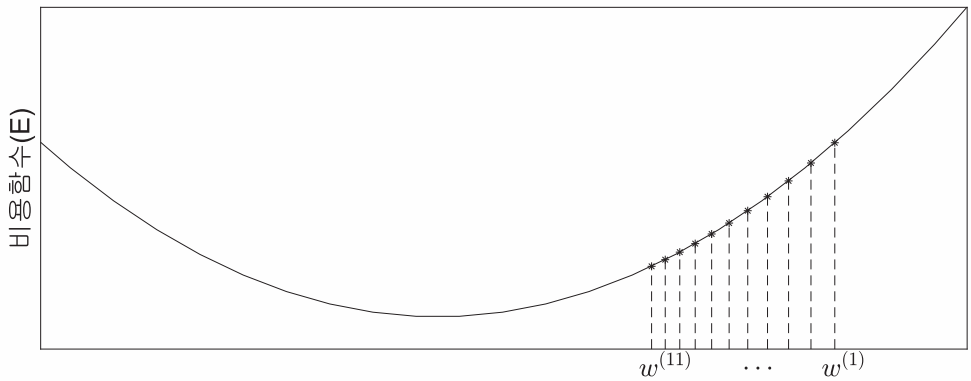

그림 2.2: 학습률($\eta = 0.01$)이 작을 경우.

예제3: 다음 주어진 비용함수 $E(w)$와 w를 이용하여 비용함수의 최솟값을 구하는 알고리즘을 작성해보자.

$$E(w) = 3(w-2)^2 + 5,$$
$$\eta = 0.1,\ w^{(0)} = 5.$$

초기 비용함수를 구해보면, $E(w^{(1)}) = 32$ 이다.

Step 1. 식 (2.6)을 적용하기 위해 $\frac{dE}{dw}$을 구하면 다음과 같다.

$$\frac{dE}{dw} = 6(w-2)$$

Step 2. 식 (2.6)을 반복하여 적용하여 보자.

$$w^{(2)} = w^{(1)} - \eta \frac{dE}{dw}\left(w^{(1)}\right) = 5 - 0.1 \times 6 \times (5-2) = 3.2$$

다음 결과와 [그림 2.3]을 보면 ⟨예제 2⟩보다 $w^{(n)}$이 빠르고 정확하게 수렴하고 있음을 알 수 있다.

```
w(1)=5.000000, E(w(1))=32.000000
w(2)=3.200000, E(w(2))=9.320000
w(3)=2.480000, E(w(3))=5.691200
w(4)=2.192000, E(w(4))=5.110592
w(5)=2.076800, E(w(5))=5.017695
w(6)=2.030720, E(w(6))=5.002831
w(7)=2.012288, E(w(7))=5.000453
w(8)=2.004915, E(w(8))=5.000072
w(9)=2.001966, E(w(9))=5.000012
w(10)=2.000786, E(w(10))=5.000002
w(11)=2.000315, E(w(11))=5.000000
```

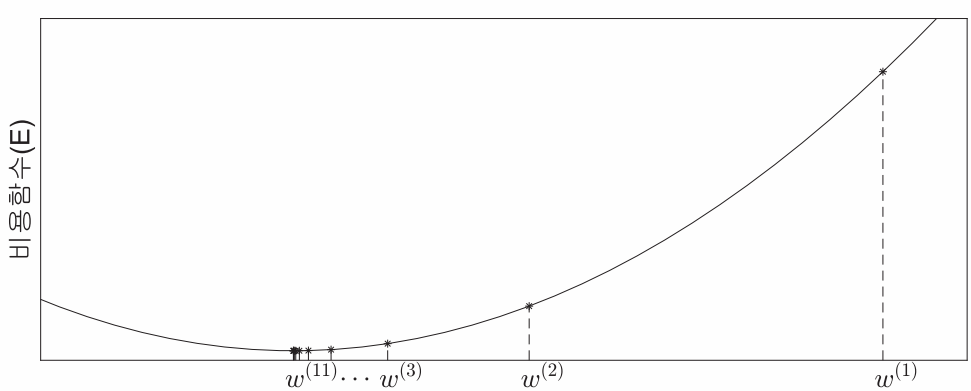

그림 2.3: 학습률($\eta = 0.1$)이 적절할 경우.

앞 예제들을 통해, 1변수 비용함수의 최솟값을 만족하는 변수를 구하는 문제

```
%%% two_variable_book.m %%%
clear;
w1(1) = 8; w2(1) = 10; eta = 0.1;
E = inline('3*(xx-2)^2+2*(yy-3)^2+3','xx','yy');
dEdw1 = inline('6*(xx-2)','xx');
dEdw2 = inline('4*(yy-3)','yy');
fprintf('(w1(1), w2(1)) = (%f, %f),E(w1(1),w2(1))=%f \n',
w1(1),w2(1),E(w1(1),w2(1)));
for i=1:10
    w1(i+1) = w1(i) - eta*dEdw1(w1(i));
    w2(i+1) = w2(i) - eta*dEdw2(w2(i));
fprintf('(w1(%d), w2(%d))=(%f, %f), E(w1(%d),w2(%d))=%f  \n',
i+1,i+1,w1(i+1),w2(i+1),i+1,i+1,E(w1(i+1),w2(i+1)));
end
```

여기서, 주의할 점은 ' '의 중간을 ...으로 줄 바꿈을 할 경우 오류가 발생한다. 따라서 ' ' 다음에 ...을 사용해야한다.

```
(w1(1), w2(1)) = (8.00000, 10.0000), E(w1(1),w2(1))=209.0000
(w1(2), w2(2)) = (4.40000, 7.20000), E(w1(2),w2(2))=55.56000
(w1(3), w2(3)) = (2.96000, 5.52000), E(w1(3),w2(3))=18.46560
(w1(4), w2(4)) = (2.38400, 4.51200), E(w1(4),w2(4))=8.01465
(w1(5), w2(5)) = (2.15360, 3.90720), E(w1(5),w2(5))=4.71680
(w1(6), w2(6)) = (2.06144, 3.54432), E(w1(6),w2(6))=3.60389
```

을 구하는 알고리즘을 작성해보자.

$$E(\mathbf{w}) = E(w_1, w_2, w_3) = 3(w_1 - 2)^2 + 2(w_2 - 3)^2 + 4(w_3 - 1)^2 + 3,$$
$$\eta = 0.1,\ \mathbf{w}^{(1)} = (w_1^{(1)}, w_2^{(1)}, w_3^{(1)}) = (8, 10, 5).$$

식 (2.6)에 적용하기 위해 $\frac{\partial E}{\partial w_1}$, $\frac{\partial E}{\partial w_2}$와 $\frac{\partial E}{\partial w_3}$을 구하면 다음과 같다.

$$\frac{\partial E}{\partial w_1} = 6(w_1 - 2),\ \frac{\partial E}{\partial w_2} = 2(w_2 - 3),\ \frac{\partial E}{\partial w_3} = 8(w_3 - 1).$$

주어진 비용함수는 $w_1 = 2, w_2 = 3, w_3 = 1$에서 최솟값을 갖는다.

$$\mathbf{w}^{(2)} = \mathbf{w}^{(1)} - \eta\left(\frac{\partial E}{\partial w_1}\left(\mathbf{w}^{(1)}\right), \frac{\partial E}{\partial w_2}\left(\mathbf{w}^{(1)}\right), \frac{\partial E}{\partial w_3}\left(\mathbf{w}^{(1)}\right)\right) = (4.4, 7.2, 1.8),$$
$$\mathbf{w}^{(3)} = \mathbf{w}^{(2)} - \eta\left(\frac{\partial E}{\partial w_1}\left(\mathbf{w}^{(2)}\right), \frac{\partial E}{\partial w_2}\left(\mathbf{w}^{(2)}\right), \frac{\partial E}{\partial w_3}\left(\mathbf{w}^{(2)}\right)\right) = (2.96, 5.52, 1.16),$$
$$\vdots$$
$$\mathbf{w}^{(11)} = \mathbf{w}^{(10)} - \eta\left(\frac{\partial E}{\partial w_1}\left(\mathbf{w}^{(10)}\right), \frac{\partial E}{\partial w_2}\left(\mathbf{w}^{(10)}\right), \frac{\partial E}{\partial z}\left(\mathbf{w}^{(10)}\right)\right)$$
$$= (2.000629, 3.042326, 1.000000).$$

3변수로 비용함수가 정의되었을 경우를 MATLAB 코드로 작성해서 알아보자.

```
(w1(2), w2(2), w3(2))= (4.400000, 7.200000, 1.800000),
 E(w1(2),w2(2),w3(2))=58.120000
(w1(3), w2(3), w3(3)) = (2.960000, 5.520000,1.160000),
 E(w1(3),w2(3),w3(3))=18.568000
(w1(4), w2(4), w3(4)) = (2.384000, 4.512000,1.032000),
 E(w1(4),w2(4),w3(4))=8.018752
(w1(5), w2(5), w3(5)) = (2.153600, 3.907200,1.006400),
 E(w1(5),w2(5),w3(5))=4.716966
(w1(6), w2(6), w3(6)) = (2.061440, 3.544320,1.001280),
 E(w1(6),w2(6),w3(6))=3.603900
(w1(7), w2(7), w3(7)) = (2.024576, 3.326592,1.000256),
 E(w1(7),w2(7),w3(7))=3.215137
(w1(8), w2(8), w3(8)) = (2.009830, 3.195955,1.000051),
 E(w1(8),w2(8),w3(8))=3.077087
(w1(9), w2(9), w3(9)) = (2.003932, 3.117573,1.000010),
 E(w1(9),w2(9),w3(9))=3.027693
(w1(10), w2(10), w3(10)) = (2.001573, 3.070544,1.000002),
 E(w1(10),w2(10),w3(10))=3.009960
(w1(11), w2(11), w3(11)) = (2.000629, 3.042326,1.000000),
 E(w1(11),w2(11),w3(11))=3.003584
```

근삿값이 $w_1 = 2, w_2 = 3, w_3 = 1$로 근사되는 것을 확인할 수 있다.

```
N = 10; w = 0.5*ones(1,N); eta = 0.7;
E = inline('0.5*sum((w-[1:N]/N).^2)','w','N');
dEdw = inline('xx-tt','xx','tt');
fprintf('W(1) = (%f, %f, %f, %f, %f, %f, %f, %f, %f, %f),
E(W(1))=%f \n',w,E(w,N));
for k=1:10
for i=1:N
  new_w(i) = w(i) - eta*dEdw(w(i),i/N);
end
 w =new_w;
 fprintf('W(%d) = (%f, %f, %f, %f, %f, %f, %f, %f, %f, %f),
E(W(%d))=%f \n',k+1,w,k+1,E(w,N));
end
```

```
W(1) = (0.50000, 0.50000, 0.50000, 0.50000, 0.50000, 0.50000,
 0.50000, 0.50000, 0.50000, 0.50000), E(W(1))=0.42500
W(2) = (0.22000, 0.29000, 0.36000, 0.43000, 0.50000, 0.57000,
 0.64000, 0.71000, 0.78000, 0.85000), E(W(2))=0.03825
W(3) = (0.13600, 0.22700, 0.31800, 0.40900, 0.50000, 0.59100,
 0.68200, 0.77300, 0.86400, 0.95500), E(W(3))=0.00344
W(4) = (0.11080, 0.20810, 0.30540, 0.40270, 0.50000, 0.59730,
 0.69460, 0.79190, 0.88920, 0.98650), E(W(4))=0.00031
W(5) = (0.10324, 0.20243, 0.30162, 0.40081, 0.50000, 0.59919,
```

$$x_2 = -1, \quad t_2 = 0$$
$$x_3 = 0, \quad t_3 = 1$$
$$x_4 = 1, \quad t_4 = 1$$
$$x_5 = 2, \quad t_5 = 1$$

여기서 X는 분류하고자 하는 데이터이고, T는 각 데이터에 해당하는 목표값이다.

분류기를 구축한다는 것은 궁극적으로는 다음과 같은 함수를 만드는 것이다.

$$y = f(x) = \phi(u) = \phi(w_0 + wx)$$

여기서 w를 가중값(Weight), w_0를 편향(Bias), u는 가중합, 그리고 ϕ는 활성화 함수라 한다. 본 절에서는 시그모이드(Sigmoid) 함수를 사용하겠다. 시그모이드 함수와 시그모이드의 도함수는 다음과 같다 ([그림 2.5] 참조).

$$\phi(u) = \frac{1}{1 + e^{-u}}, \tag{2.7}$$
$$\phi'(u) = \frac{e^{-u}}{(1 + e^{-u})^2}. \tag{2.8}$$

편의상 $W = [w_0 \; w]$를 가중값이라 하자. 일반적으로 처음부터 최적의 가중값을 알지 못하기 때문에 처음에는 임의의 값으로 시작한다. 가중값이 임의의 값이므로 함수 f는 우리가 원하는 함수가 아니다. 따라서 우리가 원하는 함수를 얻기 위해 가중값을 수정해야 하며, 수정하기 위해 비용함수를 다음과 같이

현재 설정되어 있는 가중값을 이용하여 함숫값(y)을 계산한다.

$$y = \phi(u) = \phi(w_0 + wx)$$

과정 2-2. 기울기를 구한다.

$$\frac{\partial E_m}{\partial w_0} = \frac{\partial E_m}{\partial u}\frac{\partial u}{\partial w_0} = (y-t)\phi'(u),$$
$$\frac{\partial E_m}{\partial w} = \frac{\partial E_m}{\partial u}\frac{\partial u}{\partial w} = (y-t)\phi'(u)x.$$

과정 2-3. 계산된 기울기와 학습률을 이용하여 가중값을 수정한다.

$$w_0^{(n+1)} = w_0^{(n)} - \eta\frac{\partial E_m}{\partial w_0},$$
$$w^{(n+1)} = w^{(n)} - \eta\frac{\partial E_m}{\partial w}.$$

과정 3. 5개의 데이터에 대해여 과정 2를 모두 수행한 후 수행 전 오차와 수행 후 오차의 차이(Residual)를 계산한다.

$$|E(X, W^{(n+1)}) - E(X, W^{(n)})|$$

과정 4. 과정 3에서 계산한 차이의 크기가 우리가 설정한 오차의 최솟값보다 크거나 설정한 학습 횟수보다 작으면 $n = n+1$로 설정하고 과정 2, 3을 반복한다. 그렇지 않다면 학습을 끝낸다.

```
    E2 = E(y,t);
    Resid = abs(E2-E1);
    E1=E2;
    n=n+1;
    if mod(n,100)==0
    fprintf('y(%d) = {%f, %f, %f, %f, %f}, E(y,t) = %f \n',  n,
y(1),y(2),y(3),y(4),y(5),E(p(u),t))
    end
end
fprintf('y(%d) = {%f, %f, %f, %f, %f}, E(y,t) = %f \n',  n,
y(1),y(2),y(3),y(4),y(5),E(p(u),t))
```

```
 y(1) = {0.644986, 0.650505, 0.655984, 0.661421, 0.666817},
 E(y,t) = 0.118316
 y(100) = {0.059742, 0.261893, 0.664588, 0.917115, 0.984074},
 E(y,t) = 0.019178
 y(200) = {0.029273, 0.216741, 0.717450, 0.958848, 0.995345},
 E(y,t) = 0.012938
 y(300) = {0.017793, 0.190763, 0.754153, 0.975561, 0.998079},
 E(y,t) = 0.009775
 y(400) = {0.011946, 0.171809, 0.780671, 0.983889, 0.999047},
 E(y,t) = 0.007803
 y(500) = {0.008559, 0.157047, 0.800821, 0.988606, 0.999466},
```

하므로 다음과 같이 표현하여 사용한다.

$$y = \begin{cases} 0 & -b + wx \leq 0\text{인 경우,} \\ 1 & -b + wx > 0\text{인 경우.} \end{cases}$$

여기서, $-b$는 w_0와 같다. 만약 편향을 수정하지 않고 고정하면, 즉, $b = 0$이라면 [그림 2.7]과 같이 x_3을 제대로 분류를 못하고 있음을 확인할 수 있다. 그 이유는 항상 원점을 지나므로 분류를 제대로 하지 못하게 되는 것이다.

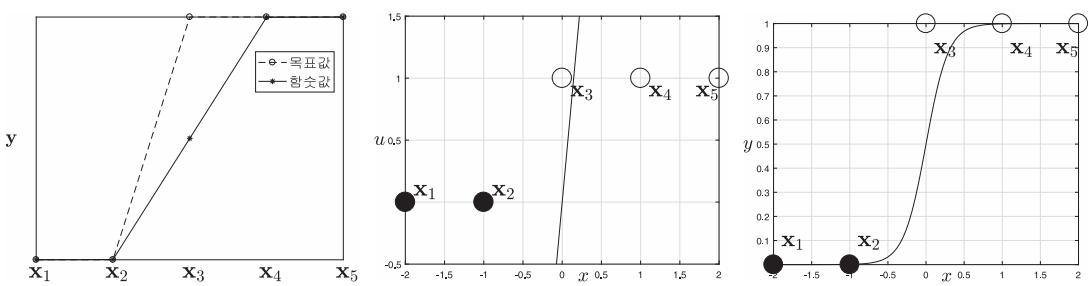

그림 2.7: 편향이 없는 경우: (a) 목표값과 함숫값 비교, (b) 추정한 가중값을 이용하여 구한 결정 경계, (c) 예제의 결과.

활성화 함수는 2가지의 의미를 가지고 있다. 첫 번째는 비선형의 성질을 갖는 것이다. 이는 복잡한 분류를 할 때 유용하게 활용될 수 있다. 두 번째로, 분류 할 때의 특성을 가지고 있다. 위 예제와 같이 0, 1을 분류할 때에는 시그모이드 함수와 같이 함수 값이 $0 \sim 1$ 갖는 함수를 사용하면 된다. 따라서, 분류하는 특성에 맞는 적절한 활성화 함수를 사용하면 된다.

제 3 절 은닉층 (Hidden Layer)

앞 절에서 ⟨예제 7⟩을 통해 단순하게 입력층과 출력층으로 구성되어 있는 분

```
E = inline('sum(0.5*(yy-tt).^2)/6','yy','tt');
y=p(u); E1 = E(y,t);
fprintf('y(1)={%1.3f,%1.3f,%1.3f,%1.3f,%1.3f,%1.3f},
E(y,t)=%1.3f\n', y(1),y(2),y(3),y(4),y(5),y(6),E(p(u),t))
n=1;
while Resid>=Tol && n<=MaxIter
    for i=1:length(x)
        u(i) = w0+w*x(i);
        y(i) = p(u(i));
        dEdw0 = (y(i)-t(i))*Dp(u(i));
        dEdw = (y(i)-t(i))*Dp(u(i))*x(i);
        w0 = w0 - eta*dEdw0;
        w = w - eta*dEdw;
    end
    u = w0+w*x;
    y = p(u);
    E2 = E(y,t);
    Resid = abs(E2-E1);
    E1=E2;
    n=n+1;
    if mod(n,200)==0
fprintf('y(%d)={%1.3f,%1.3f,%1.3f,%1.3f,%1.3f,%1.3f},
E(y,t)=%1.3f\n', n, y(1),y(2),y(3),y(4),y(5),y(6),E(p(u),t))
```

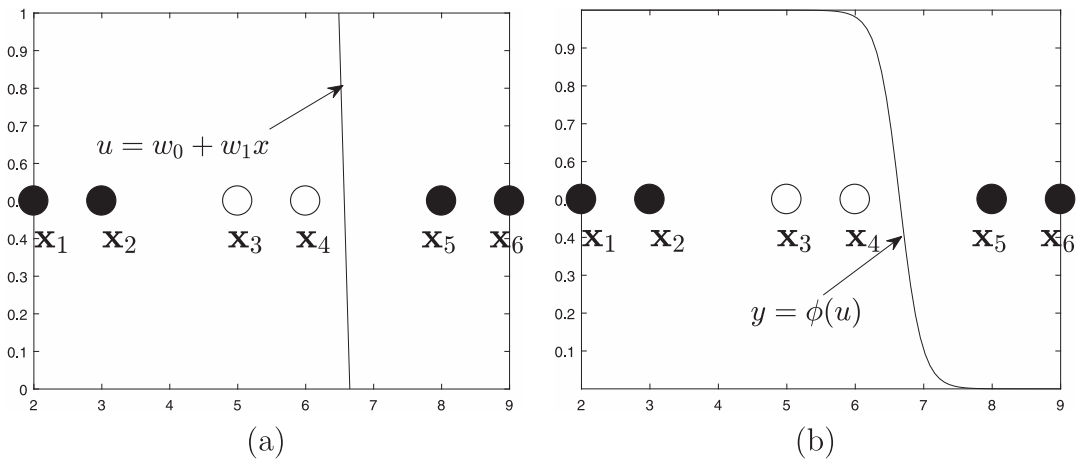

그림 2.8: (a) 선형 결정 경계, (b) 출력층의 출력값.

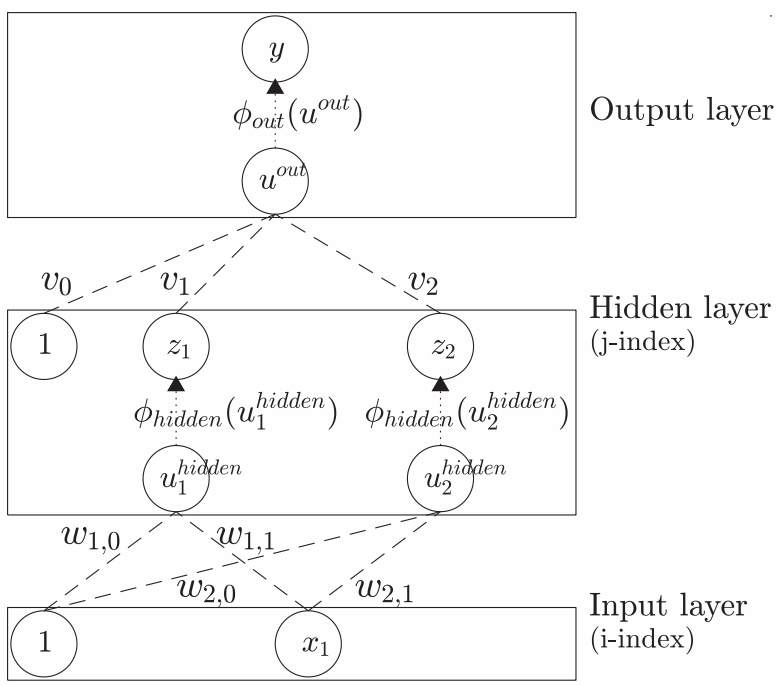

그림 2.9: 은닉층을 추가한 분류기 구조.

함수는 다음과 같다.

$$E(X,W,V) = \frac{1}{6}\sum_{m=1}^{6} E_m(x_m,W,V) = \frac{1}{6}\sum_{m=1}^{6}\frac{1}{2}||t_m - y_m||^2.$$

은닉층을 추가한 분류기 구축 과정:

은닉층이 한층 추가되고 은닉노드가 2개인 경우를 살펴보자.

과정 1. 초기설정

학습률(η)은 0.1, 초기 가중값(W,V)은 임의의 값으로 설정, 그리고 학습횟수(n)를 설정한다.

과정 2. 6개의 데이터($\mathbf{x}$)에 대해서 가중값(W,V)을 수정하는데 한 번에 한 개의 데이터에 대해서 다음 가중값을 수정하면 된다.

과정 2-1. 함숫값(y) 계산

현재 설정되어 있는 가중 값을 이용하여 출력값(y)을 계산한다.

$$\begin{aligned} z_1 &= \phi_{hidden}(u_1^{hidden}) = \phi_{hidden}(w_{10} + w_{11}x_1) \\ z_2 &= \phi_{hidden}(u_2^{hidden}) = \phi_{hidden}(w_{20} + w_{21}x_1) \\ y &= \phi_{out}(u^{out}) = \phi(v_0 + v_1z_1 + v_2z_2) \end{aligned}$$

과정 2-2. 기울기 계산

$$\begin{aligned} \frac{\partial E_m}{\partial v_j} &= \frac{\partial E_m}{\partial u^{out}}\frac{\partial u^{out}}{\partial v_j} = (y-t)\phi'(u^{out})z_j, \quad j=0,1,2, \\ \frac{\partial E_m}{\partial w_{j,i}} &= \frac{\partial E_m}{\partial u_j^{hidden}}\frac{\partial u_j^{hidden}}{\partial w_{j,i}} = \frac{\partial E_m}{\partial u^{out}}\frac{\partial u^{out}}{\partial u_j^{hidden}}x_i \\ &= (y-t)\phi'(u^{out})\phi'(u_j^{hidden})v_jx_i, \quad i=0,1, \quad j=1,2. \end{aligned}$$

```
E = inline('sum(0.5*(yy-tt).^2)/6','yy','tt');

E1 = 1000;

while Resid>=Tol && n <=MaxIter

    for m=1:length(T)

        x=[1;X(m)];

        uh=w*x;

        z=p(uh);

        z=[1;z];

        uo=v*z;

        y(m)=p(uo);

        dEdv=(y(m)-T(m))*Dp(uo)*z';

        dEdw=((y(m)-T(m))*Dp(uo))*(Dp(uh).*v(2:end)')*x';

        v=v-eta*dEdv;

        w=w-eta*dEdw;

    end

    for m=1:length(T)

        x=[1;X(m)];

        uh=w*x;

        z=p(uh);

        z=[1;z];

        uo=v*z;

        y(m)=p(uo);

    end
```

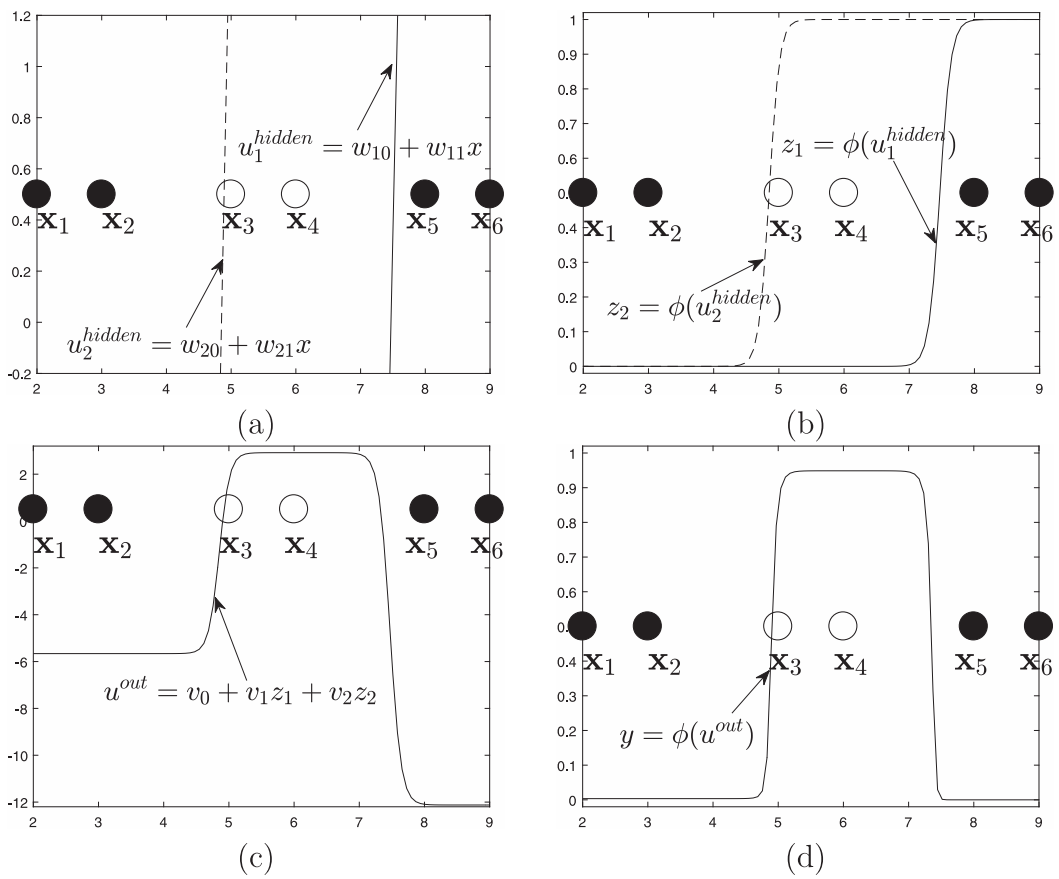

그림 2.10: 예제 8. 추정한 가중값을 이용하여 구한 (a) 선형 결정 경계, (b) 은닉층의 출력값, (c) 비선형 결정 경계, (d) 예제의 결과.

이블 데이터이다. 학습이미지 데이터는 $47040000(=60000\times 28\times 28)$차원의 벡터로 구성되어 있고 $784(=28\times 28)$개마다 1개의 이미지 데이터 의미를 가지고 있다.

$$
\begin{aligned}
X(1,1:784) &= [\ x_1, \quad x_2, \quad \cdot\ \cdot\ \cdot\ \cdot\ , x_{784}] \\
X(2,1:784) &= [\ x_{785}, \quad x_{786}, \quad \cdot\ \cdot\ \cdot\ \cdot\ , x_{1568}] \\
&\ \ \vdots \qquad\qquad\qquad\quad \vdots \\
X(60000,1:784) &= [\ x_{47039217}, \quad \cdot\ \cdot\ \cdot\ \cdot\ , x_{47040000}\ ]
\end{aligned}
$$

그림 2.12: 전처리된 이미지 데이터 행렬.

[그림 2.12]와 같이 우리는 47040000개의 정보를 784개 단위로 6만개 이미지로 분리한 뒤 학습하는데 활용할 것이다. 설명의 편의를 위해서 MNIST 데이터베이스를 읽어 오는 대신에 본 교재에서는 데이터베이스를 사용하기 쉽게 전처리하여 제공하겠다. 다음 [그림 2.13]은 학습 데이터 안에 있는 0부터 9까지의 숫자들을 출력한 결과이다. 각 그림의 격자는 28×28로 되어 있다.

제 5 절 숫자인식 머신러닝 알고리즘

28×28 크기의 숫자 이미지 데이터 $\mathbf{x}$가 입력되면 그 이미지가 어떤 숫자인지를 알려주는 $\mathbf{y}=(y_1,y_2,\cdots,y_{10})$를 출력하는 함수 f를 구축하려고 한다.

$$\mathbf{y}=f(\mathbf{x},W,V), \tag{2.9}$$

여기서 W, V는 알고리즘이 주어진 학습 데이터를 이용하여 학습을 통해서 정해야 하는 상수 행렬이다. 만약에 $y_k=\max_{1\le i\le 10} y_i$ 이면 입력 이미지 $\mathbf{x}$를 숫자 k로 인식을 할 것이다. 여기서 $k=10$인 경우는 숫자 0과 동일시한다.

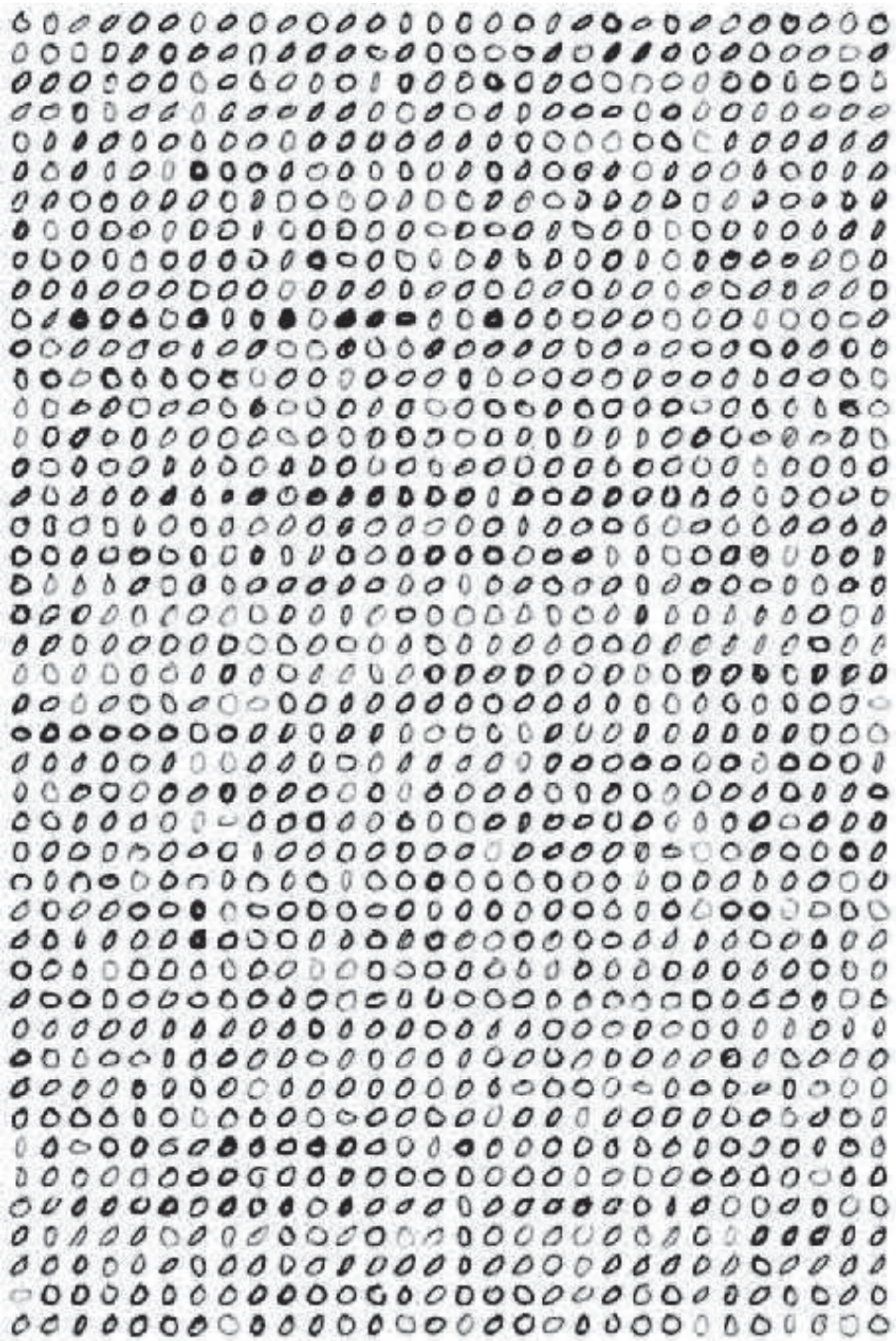

그림 2.14: MNIST 데이터베이스 숫자 0.

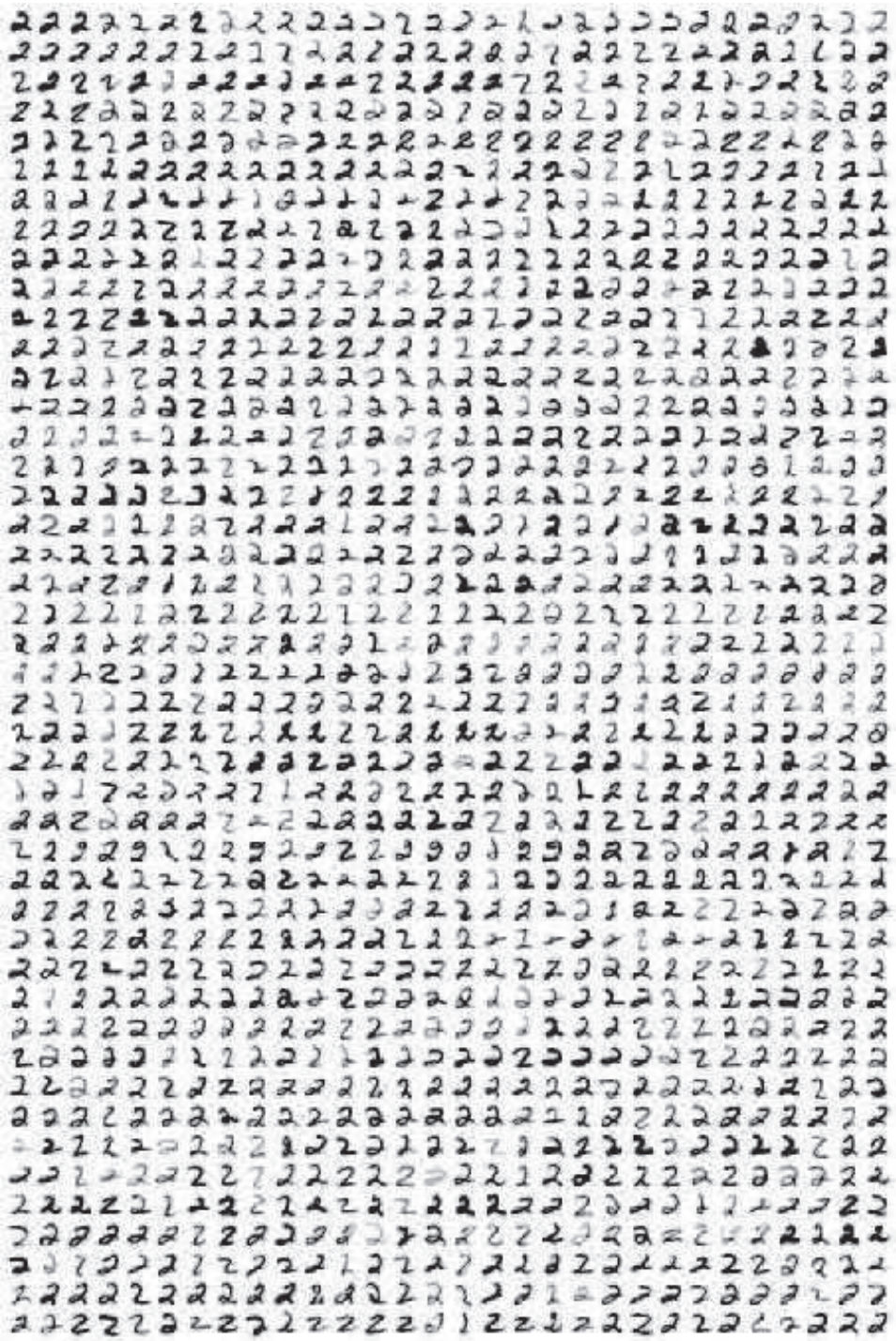

그림 2.16: MNIST 데이터베이스 숫자 2.

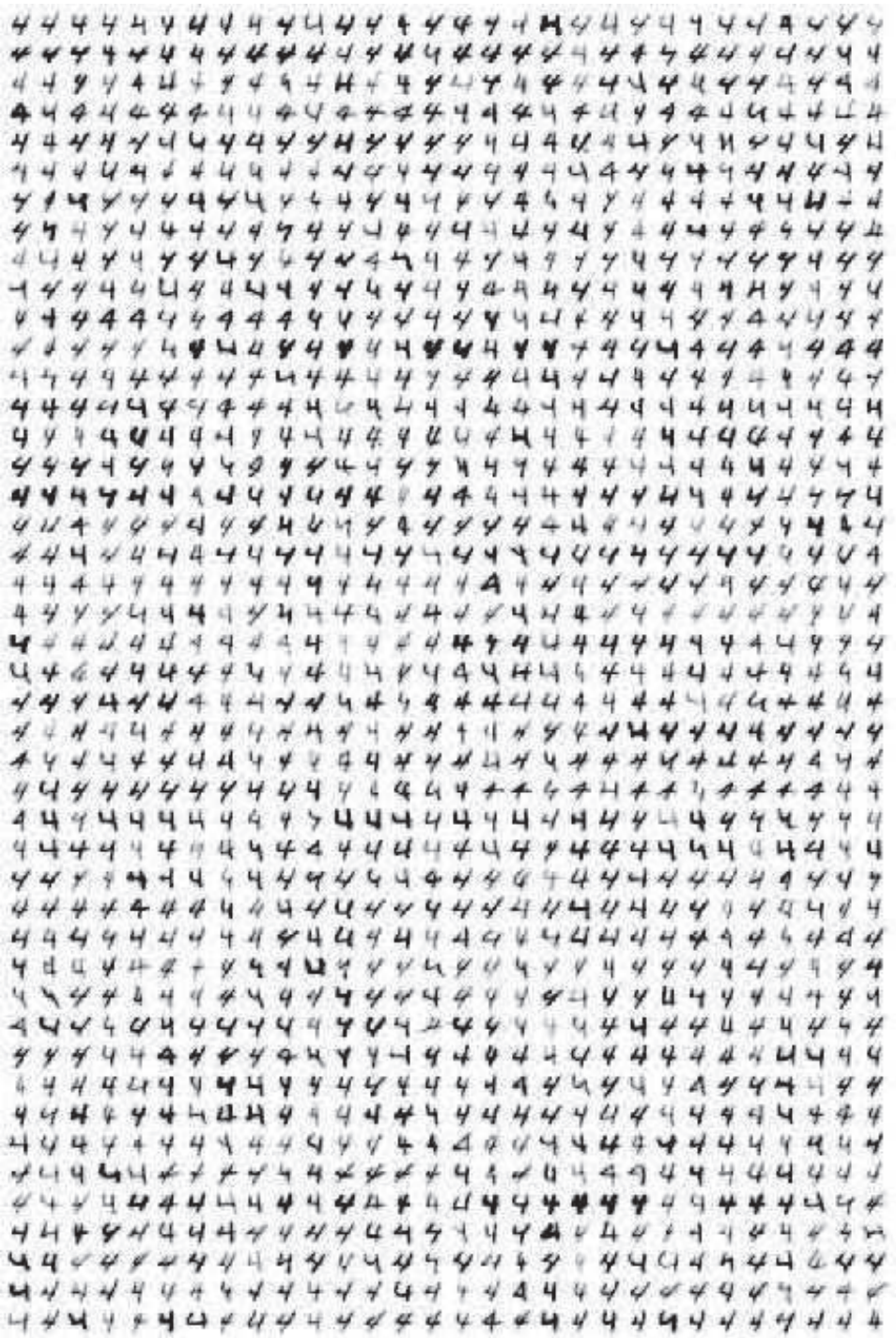

그림 2.18: MNIST 데이터베이스 숫자 4.

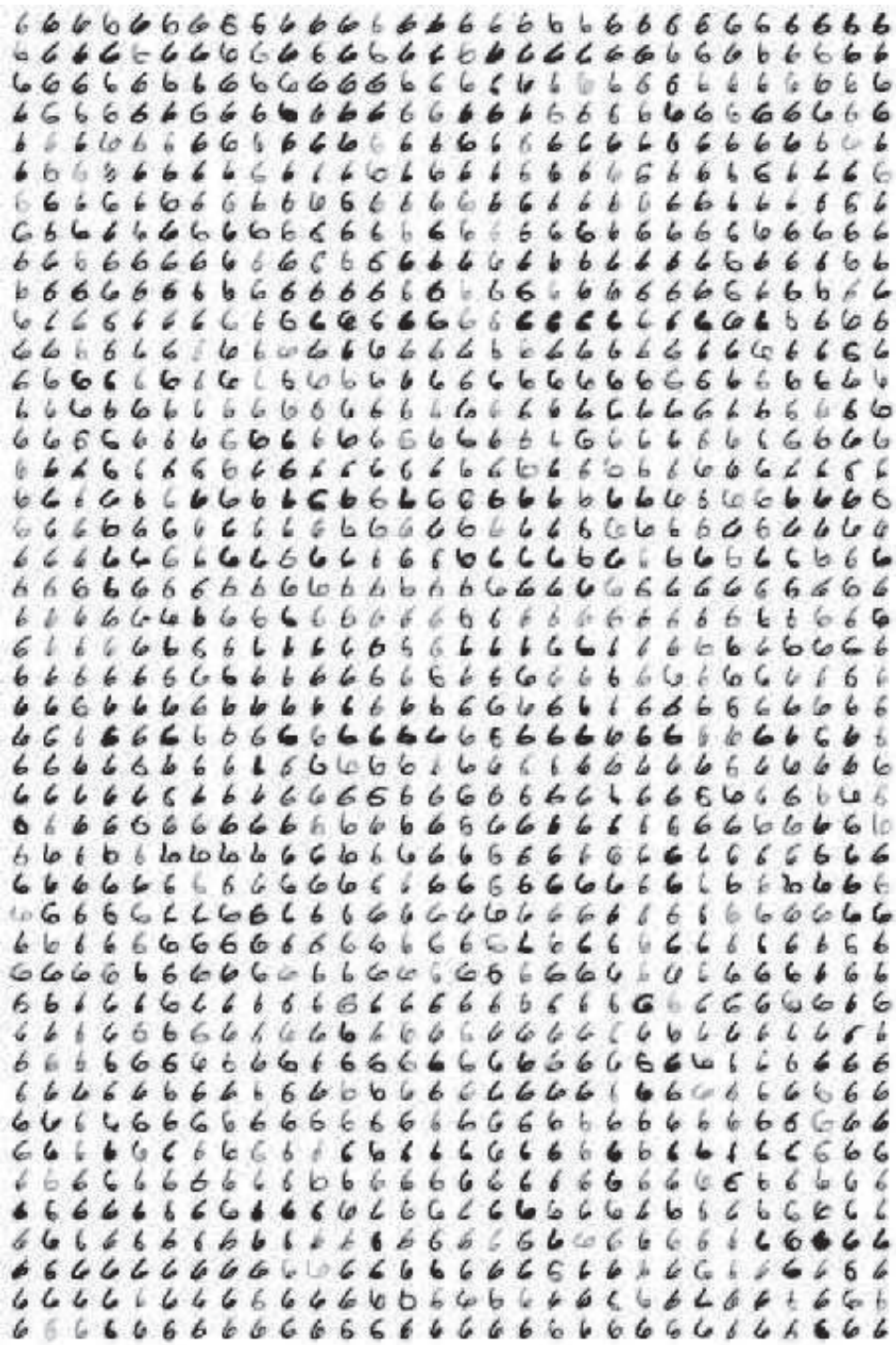

그림 2.20: MNIST 데이터베이스 숫자 6.

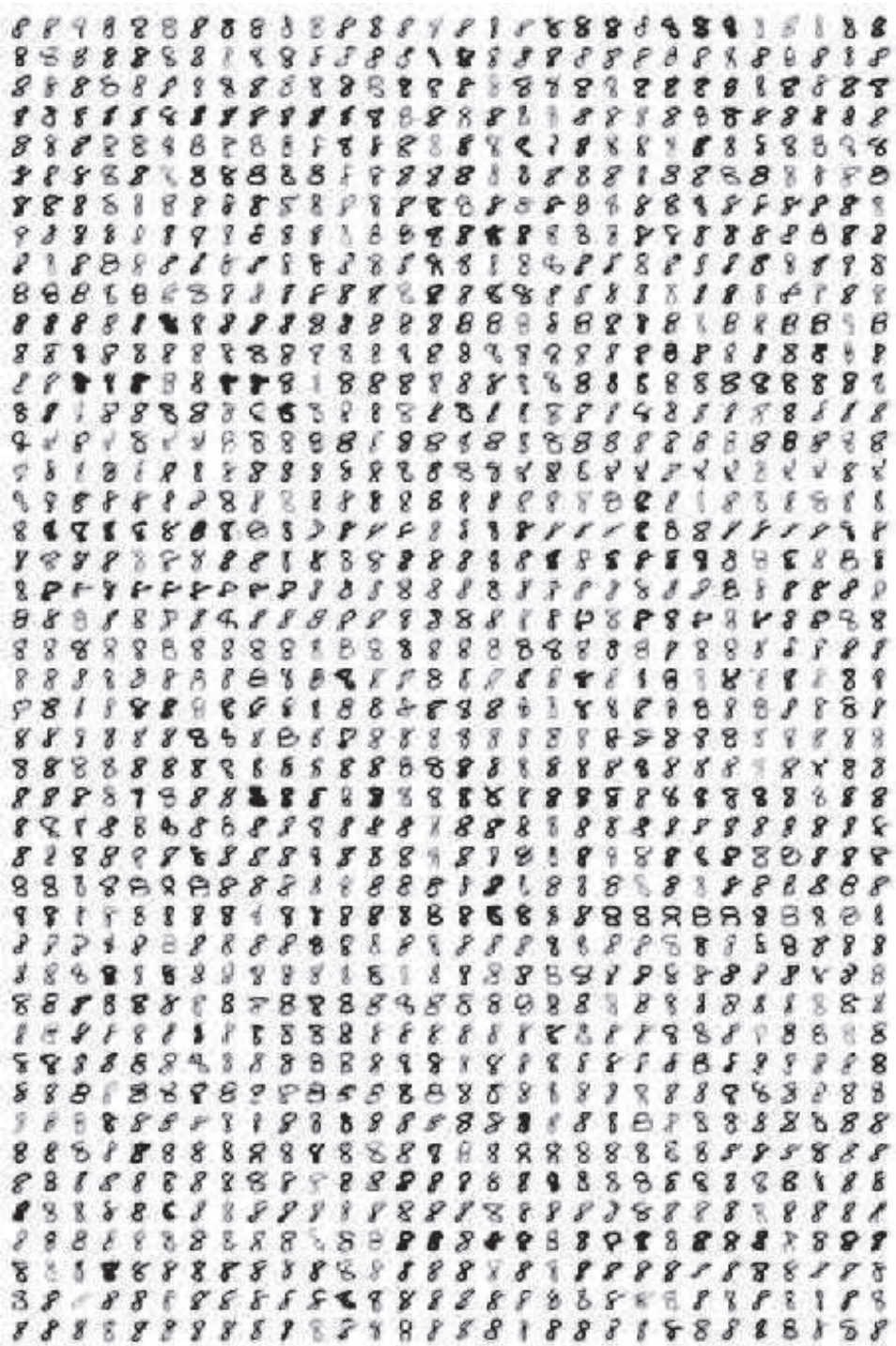

그림 2.22: MNIST 데이터베이스 숫자 8.

숫자인식 머신러닝 알고리즘의 목표는 주어진 학습 데이터로부터 W, V의 값을 잘 구해서, 새로운 숫자 이미지의 숫자를 함수 (2.9)를 이용하여 판별하는 것이다. 행렬 W, V의 값이 주어졌을 때, $\mathbf{y}$를 구하는 방법을 자세히 알아보자. 숫자 이미지 데이터 $\mathbf{x} = [x_1, x_2, \cdots, x_{784}]^T$에 $x_0 = 1$을 추가하여 $\tilde{\mathbf{x}} = [x_0, x_1, x_2, \cdots, x_{784}]^T$ 을 입력 데이터로 놓자. 총 30개의 은닉층의 노드들 각각을 계산하기 위해서 주어진 이미지 데이터 $\tilde{\mathbf{x}} = [x_0\ x_1\ x_2\ \cdots\ x_{784}]^T$에 가중값 785개 $(w_{j,0}, \cdots, w_{j,i}, \cdots, w_{j,784})$가 곱해져서 합산하게 된다. 즉, $j = 1, \cdots, 30$에 대해서 다음을 계산한다.

$$u_j^{hidden} = w_{j,0}x_0 + \cdots + w_{j,i}x_i + \cdots + w_{j,784}x_{784} \tag{2.10}$$

[그림 2.24]을 보면 위 과정은 입력층(Input layer)에서 은닉층 (Hidden layer)에 도식화 되었다.

이를 행렬과 벡터의 곱으로 나타내면 다음과 같다.

$$\mathbf{u}^{hidden} = \begin{pmatrix} u_1^{hidden} \\ \vdots \\ u_j^{hidden} \\ \vdots \\ u_{30}^{hidden} \end{pmatrix} = \begin{pmatrix} \sum_{i=0}^{784} w_{1,i}x_i \\ \vdots \\ \sum_{i=0}^{784} w_{j,i}x_i \\ \vdots \\ \sum_{i=0}^{784} w_{30,i}x_i \end{pmatrix} \tag{2.11}$$

$$
= \begin{pmatrix}
w_{1,0}x_0 + \cdots + w_{1,i}x_i + \cdots + w_{1,784}x_{784} \\
\vdots \\
w_{j,0}x_0 + \cdots + w_{j,i}x_i + \cdots + w_{j,784}x_{784} \\
\vdots \\
w_{30,0}x_0 + \cdots + w_{30,i}x_i + \cdots + w_{30,784}x_{784}
\end{pmatrix}
$$

$$
= \begin{pmatrix}
w_{1,0} & \cdots & w_{1,i} & \cdots & w_{1,784} \\
\vdots & & \vdots & & \vdots \\
w_{j,0} & \cdots & w_{j,i} & \cdots & w_{j,784} \\
\vdots & & \vdots & & \vdots \\
w_{30,0} & \cdots & w_{30,i} & \cdots & w_{30,784}
\end{pmatrix}
\begin{pmatrix}
x_0 \\ \vdots \\ x_i \\ \vdots \\ x_{784}
\end{pmatrix}
= W\tilde{\mathbf{x}}
$$

여기서 $x_0 = 1$, 그리고 $w_{j,0},\ j = 1,\ \cdots, 30$는 편향이다. 가중값 W는 30×785 크기의 행렬이다. 이렇게 구한 가중합 $\mathbf{u}^{hidden}$은 은닉층의 입력이 되고 은닉층의 활성화 함수로 시그모이드 (sigmoid) 함수 $\phi_{hidden}(u) = \dfrac{1}{1+e^{-u}}$를 사용하여

$$\mathbf{u}^{out} = \begin{pmatrix} u_1^{out} \\ \vdots \\ u_k^{out} \\ \vdots \\ u_{10}^{out} \end{pmatrix} = \begin{pmatrix} \sum_{j=0}^{30} v_{1,j} z_j \\ \vdots \\ \sum_{j=0}^{30} v_{k,j} z_j \\ \vdots \\ \sum_{j=0}^{30} v_{10,j} z_j \end{pmatrix} = \begin{pmatrix} v_{1,0} z_0 + \cdots + v_{1,j} z_j + \cdots + v_{1,30} z_{30} \\ \vdots \\ v_{k,0} z_0 + \cdots + v_{k,j} z_j + \cdots + v_{k,30} z_{30} \\ \vdots \\ v_{10,0} z_0 + \cdots + v_{10,j} z_j + \cdots + v_{10,30} z_{30} \end{pmatrix}$$

$$= \begin{pmatrix} v_{1,0} & \cdots & v_{1,j} & \cdots & v_{1,30} \\ \vdots & & \vdots & & \vdots \\ v_{k,0} & \cdots & v_{k,j} & \cdots & v_{k,j} \\ \vdots & & \vdots & & \vdots \\ v_{10,0} & \cdots & v_{10,j} & \cdots & v_{10,30} \end{pmatrix} \begin{pmatrix} z_0 \\ \vdots \\ z_j \\ \vdots \\ z_{30} \end{pmatrix} = V\tilde{\mathbf{z}} \tag{2.14}$$

여기서 $z_0 = 1$, 그리고 $v_{k,0},\ j = 1, \cdots, 10$는 편향이다. 가중값 V는 10×31 크기의 행렬이다. 이렇게 구한 가중합 $\mathbf{u}^{out}$은 출력층의 입력이 되고 출력층의 활성화 함수로 시그모이드 함수 $\phi_{out}(u) = \dfrac{1}{1+e^{-u}}$를 사용하여 출력층의 출력을 다음과 같이 계산한다.

$$E_l(\mathbf{x}_l, \mathbf{t}_l, W, V) = \frac{1}{2}\|\mathbf{y}_l - \mathbf{t}_l\|^2, \quad \|\mathbf{t}\|^2 = \sum_{k=1}^{10} t_k^2. \tag{2.18}$$

여기서 $\mathbf{t}_l = [t_1\ t_2\ \cdots\ t_{10}]^T$는 l번째 이미지의 숫자를 나타내는 값이다. 즉, 다음과 같이 각 숫자를 나타내는 벡터이다.

$$\begin{aligned}
&\text{숫자 } 1\text{번 이미지}: \quad \mathbf{t} = [1,0,0,0,0,0,0,0,0,0]\\
&\text{숫자 } 2\text{번 이미지}: \quad \mathbf{t} = [0,1,0,0,0,0,0,0,0,0]\\
&\text{숫자 } 3\text{번 이미지}: \quad \mathbf{t} = [0,0,1,0,0,0,0,0,0,0]\\
&\text{숫자 } 4\text{번 이미지}: \quad \mathbf{t} = [0,0,0,1,0,0,0,0,0,0]\\
&\text{숫자 } 5\text{번 이미지}: \quad \mathbf{t} = [0,0,0,0,1,0,0,0,0,0]\\
&\text{숫자 } 6\text{번 이미지}: \quad \mathbf{t} = [0,0,0,0,0,1,0,0,0,0]\\
&\text{숫자 } 7\text{번 이미지}: \quad \mathbf{t} = [0,0,0,0,0,0,1,0,0,0]\\
&\text{숫자 } 8\text{번 이미지}: \quad \mathbf{t} = [0,0,0,0,0,0,0,1,0,0]\\
&\text{숫자 } 9\text{번 이미지}: \quad \mathbf{t} = [0,0,0,0,0,0,0,0,1,0]\\
&\text{숫자 } 0\text{번 이미지}: \quad \mathbf{t} = [0,0,0,0,0,0,0,0,0,1]
\end{aligned}$$

편의상 $\mathbf{x} = \mathbf{x}_l$, $\mathbf{t} = \mathbf{t}_l$라 하자. 즉,

$$\begin{aligned}
\mathbf{y} &= (y_1, y_2, \cdots, y_{10}),\\
\mathbf{t} &= (t_1, t_2, \cdots, t_{10}).
\end{aligned}$$

여기서 $\phi_{out}(u)$의 미분은 다음과 같다.

$$\phi'_{out}(u) = \left(\frac{1}{1+e^{-u}}\right)' = \frac{e^{-u}}{(1+e^{-u})^2}. \tag{2.23}$$

다음으로 식 (2.20)에 있는 $\frac{\partial E_l^{(n)}}{\partial w_{j,i}}$를 유도하기 위해 식 (2.19)를 좀 더 풀어서 다음과 같이 $w_{j,i}$의 함수로 나타내자.

$$\begin{aligned}
E_l(\mathbf{x},\mathbf{t},W,V) = \|\mathbf{y}-\mathbf{t}\|^2 &= \frac{1}{2}\sum_{k=1}^{10}(\phi_{out}(u_k^{out})-t_k)^2 \\
&= \frac{1}{2}\big[(\phi_{out}(u_1^{out})-t_1)^2 + (\phi_{out}(u_2^{out})-t_2)^2 + \cdots \\
&+ (\phi_{out}(u_k^{out})-t_k)^2 + \cdots + (\phi_{out}(u_{10}^{out})-t_{10})^2\big] \\
&= \frac{1}{2}\left(\left[\phi_{out}\left(\sum_{j=0}^{30} v_{1,j}z_j\right)-t_1\right]^2 + \left[\phi_{out}\left(\sum_{j=0}^{30} v_{2,j}z_j\right)-t_2\right]^2 + \cdots \right. \\
&\left. + \left[\phi_{out}\left(\sum_{j=0}^{30} v_{k,j}z_j\right)-t_k\right]^2 + \cdots + \left[\phi_{out}\left(\sum_{j=0}^{30} v_{10,j}z_j\right)-t_{10}\right]^2\right) \\
&= \frac{1}{2}\left(\left[\phi_{out}\left(v_{1,0}+\sum_{j=1}^{30} v_{1,j}\phi_{hidden}\left(\sum_{i=0}^{784} w_{j,i}x_i\right)\right)-t_1\right]^2 \right. \\
&+ \left[\phi_{out}\left(v_{2,0}+\sum_{j=1}^{30} v_{2,j}\phi_{hidden}\left(\sum_{i=0}^{784} w_{j,i}x_i\right)\right)-t_2\right]^2 + \cdots \\
&+ \left[\phi_{out}\left(v_{k,0}+\sum_{j=1}^{30} v_{k,j}\phi_{hidden}\left(\sum_{i=0}^{784} w_{j,i}x_i\right)\right)-t_k\right]^2 + \cdots \\
&\left. + \left[\phi_{out}\left(v_{10,0}+\sum_{j=1}^{30} v_{10,j}\phi_{hidden}\left(\sum_{i=0}^{784} w_{j,i}x_i\right)\right)-t_{10}\right]^2\right).
\end{aligned} \tag{2.24}$$

$$= \phi'_{hidden}\left(u_j^{hidden}\right) \sum_{k=1}^{10} \left[(y_k - t_k)\phi'_{out}\left(u_k^{out}\right) v_{k,j}\right] x_i.$$

여기서 $\phi_{out}(u)$의 미분은 다음과 같다.

$$\phi'_{hidden}(u) = \left(\frac{1}{1+e^{-u}}\right)' = \frac{e^{-u}}{(1+e^{-u})^2}. \tag{2.26}$$

알고리즘 요약

$$\mathbf{E}(X,T,W,V) = \frac{1}{60000}\sum_{m=1}^{60000}\frac{1}{2}\|\mathbf{t}_m - f(\mathbf{x}_m,W,V)\|^2$$

1. 초기 가중값 $W^{(1)}, V^{(1)}$를 각각 크기가 30×785, 10×31 크기의 랜덤 값으로 초기화한다. 학습률 $\eta = 0.05$와 오차의 최솟값(tol), 최대반복횟수를 설정한다.

2. 6만개의 학습 이미지 데이터에 대해서 가중값 (W와 V)을 수정하는데 한 번에 한 개의 이미지 데이터에 대해서 가중 값을 수정한다.

현재 갖고 있는 가중값 $W^{(n)}, V^{(n)}$을 이용하여 다음 순서대로 구한다.

방정식 (2.11) 이용하여 $\mathbf{u}^{hidden} = W^{(n)}\mathbf{x}$,

방정식 (2.12) 이용하여 $\mathbf{z} = \phi_{hidden}(\mathbf{u}^{hidden})$,

방정식 (2.14) 이용하여 $\mathbf{u}^{out} = V^{(n)}\mathbf{z}$,

방정식 (2.15) 이용하여 $\mathbf{y} = \phi_{out}(\mathbf{u}^{out})$.

숫자 인식 테스트

다음 홈페이지에 있는 데이터와 MATLAB 코드를 이용하여 숫자인식 테스트를 수행해보자. http://math.korea.ac.kr/∼cfdkim/ML에서 OneDayML_code.zip을 다운받아서 적당한 폴더에 저장한 후에 그 폴더 안에 압축을 푼다. 압축을 푼 폴더 안에 있는 learning.m 파일을 더블클릭 한 후에 F5로 실행시키면 경로 변경창이 뜨면 경로 변경을 해주면 된다. 이렇게 실행된 코드는 MNIST의 학습 데이터를 불러오고 학습을 해서 W와 V의 값을 구한다.

```
%%% learning_book.m %%%
clear;
fileID = fopen('trainimages.bin');
X = fread(fileID,[60000 784])/255;
fclose(fileID);
fileID = fopen('trainlabels.bin');
T = fread(fileID);
fclose(fileID);
T(T==0)=10;
input_node  = 28*28;
hidden_node = 30;
output_node = 10;
rand('seed',10);
w = 0.1*(2*rand(hidden_node, input_node+1)-1);
v = 0.1*(2*rand(output_node, hidden_node+1)-1);
eta=0.05; MaxIter=50;Iter=1;
```

```
        t(T(m))=1;
        del_k=d_sigmoid(uo).*(y-t);
        dEdv=del_k*z';
        del_j=d_sigmoid(uh).*(v(:,2:end)'*del_k);
        dEdw=del_j*xm';
        v=v-eta*dEdv;
        w=w-eta*dEdw;
    end
    E2=0;
    for m=1:length(T)
        xl=[1 X(m,:)]';
        uh=w*xl;
        z=sigmoid(uh);
        z=[1;z];
        uo=v*z;
        y=sigmoid(uo);
        t(1:output_node,1)=0;
        t(T(m))=1;
        E2=E2+sum((y-t).^2);
    end
    E2=E2/length(T);
    Resid = abs(E2-E1);
    E1 = E2;
```

```
end
fprintf('New image is  : %d \n', predict_value);
```

다음은 저자가 예측해 본 결과이다.

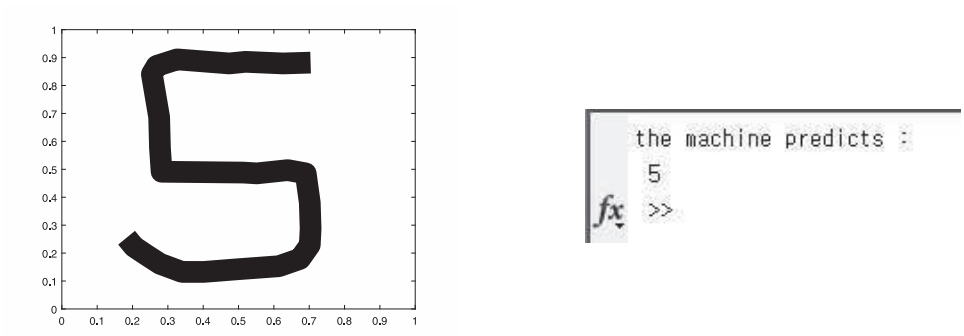

그림 2.25: 숫자 5를 예측한 결과.

위의 학습이 성공적으로 이루어 졌는지 확인하기 위해서 다음의 테스트를 진행하도록 하자. 테스트는 윈도우의 그림판을 이용하여 숫자를 그려보고, 파일로 저장한 후, 그 파일의 숫자를 성공적으로 인식하는 지 확인하는 것이다.

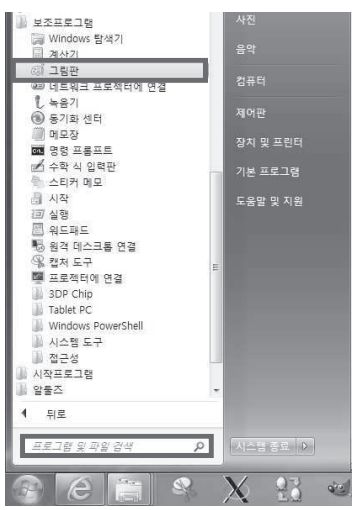

그림 2.26: 그림판 실행

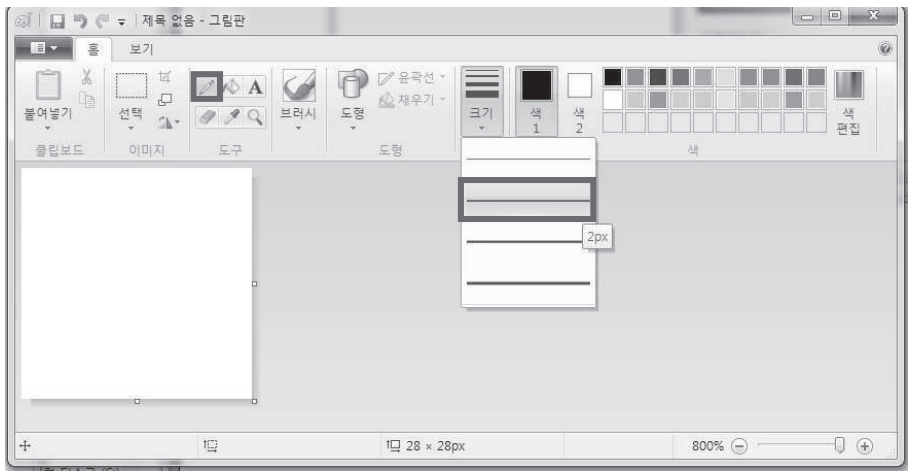

그림 2.28: 연필 두께 설정

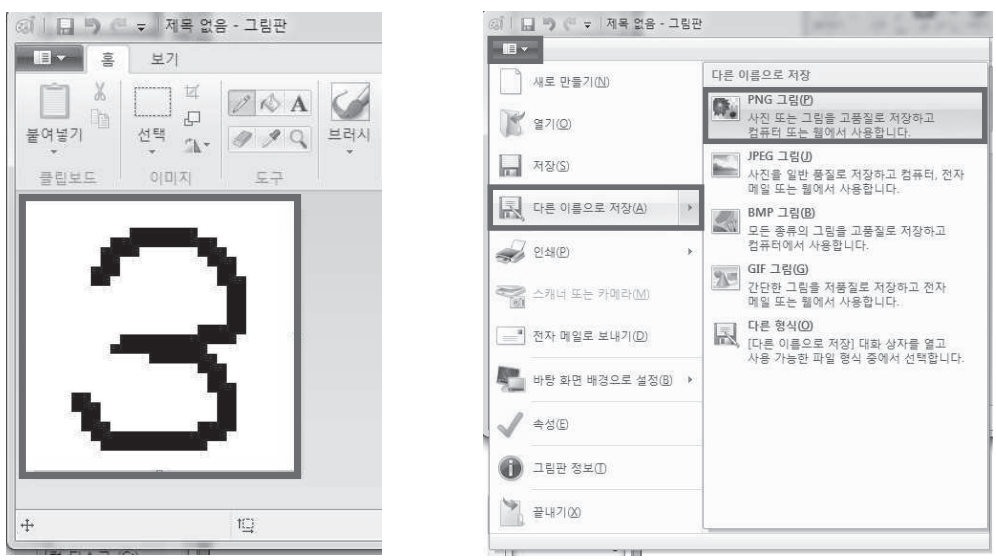

그림 2.29: 원하는 숫자 작성 및 test.png 파일 저장

장 3

메모리 부족 오류 해결

MATLAB을 사용하여 시뮬레이션을 진행하다가 보면 가끔 다음 그림과 같이 out of memory 오류가 발생하는 경우가 존재한다.

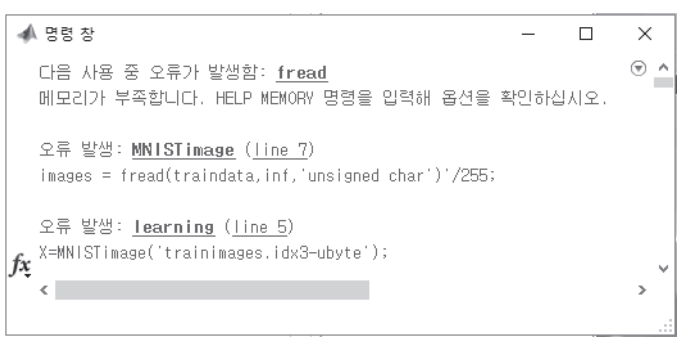

그림 3.1: MATLAB out of memory 문제 발생

이럴 경우 MATLAB 기본 설정의 Java 힙 메모리를 높여주는 방법과 시스템 상에서 프로그램에 할당하는 메모리를 높여주는 방법이 있다. 윈도우7과 윈도우 10 버전에서 테스트해본 결과 모두 성공적으로 문제를 해결할 수 있었다. 우선 Java 힙 메모리를 높여주는 방법은 다음과 같다.

제 2 절 시스템상에서 할당하는 메모리 수정

우선 윈도우10 기준으로 왼쪽 아래 부분의 돋보기 모양 아이콘을 클릭한 후, cmd를 검색하고 이를 마우스 오른쪽 버튼으로 클릭하여 관리자 권한으로 실행하기를 눌러준다.

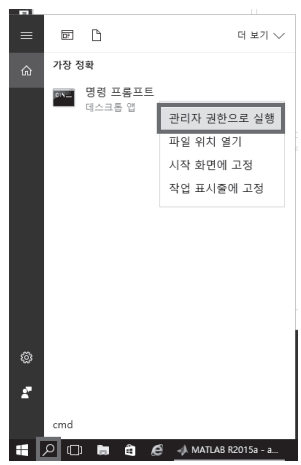

그림 3.4: cmd 관리자 권한으로 실행하기

cmd 창에 다음의 명령어(BCDEdit /set increaseuserva 3072)를 입력하면 프로그램 시작시 메모리를 3GB로 할당하게 된다.

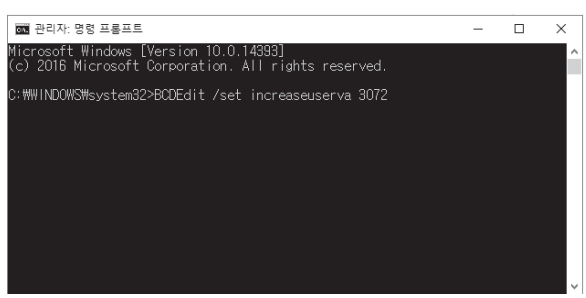

그림 3.5: 메모리 설정 명령어 입력

맨 아래 부분의 총 페이징 파일 크기를 참고하여 사용자 지정 크기를 설정하면 된다.

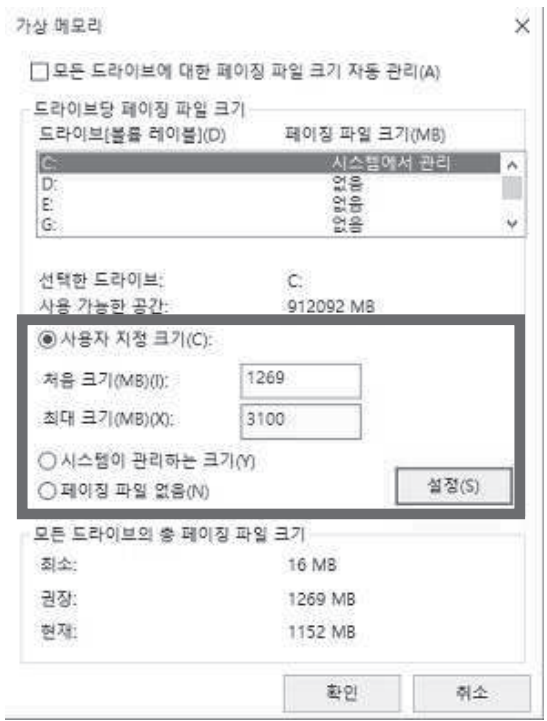

그림 3.8: 가상 메모리 사용자 지정 크기로 설정

설정을 누르고 컴퓨터를 재부팅 하면 설정값이 적용되고, out of memory 문제가 해결된다.

참고 문헌

[1] 기초부터 활용까지 패턴인식과 기계학습, 박혜영, 이관용, 이한출판사, 2011.

[2] 패턴인식, 오일석, 교보문고, 2008.

[3] 밑바닥부터 시작하는 딥러닝, 사이토 고키 저, 개앞맵시 역, 한빛미디어, 2017

[4] 딥러닝 제대로 시작하기, 오카타니 타카유키 저, 심효섭 역, 제이펍, 2016

[5] 신경망 첫걸음, 타리크 라시드 저, 송교석 역, 한빛미디어, 2017

[6] https://en.wikipedia.org/wiki/MNIST_database

[7] http://yann.lecun.com/exdb/mnist/

[8] https://codeonweb.com/entry/12045839-0aa9-4bad-8c7e-336b89401e10

[9] http://solarisailab.com/archives/303

[10] http://ufldl.stanford.edu/wiki/resources/mnistHelper.zip