인공지능 레포트

-숫자인식 프로젝트-

지도 교수님: 공용해 교수님
제출일: 2019.05.21.

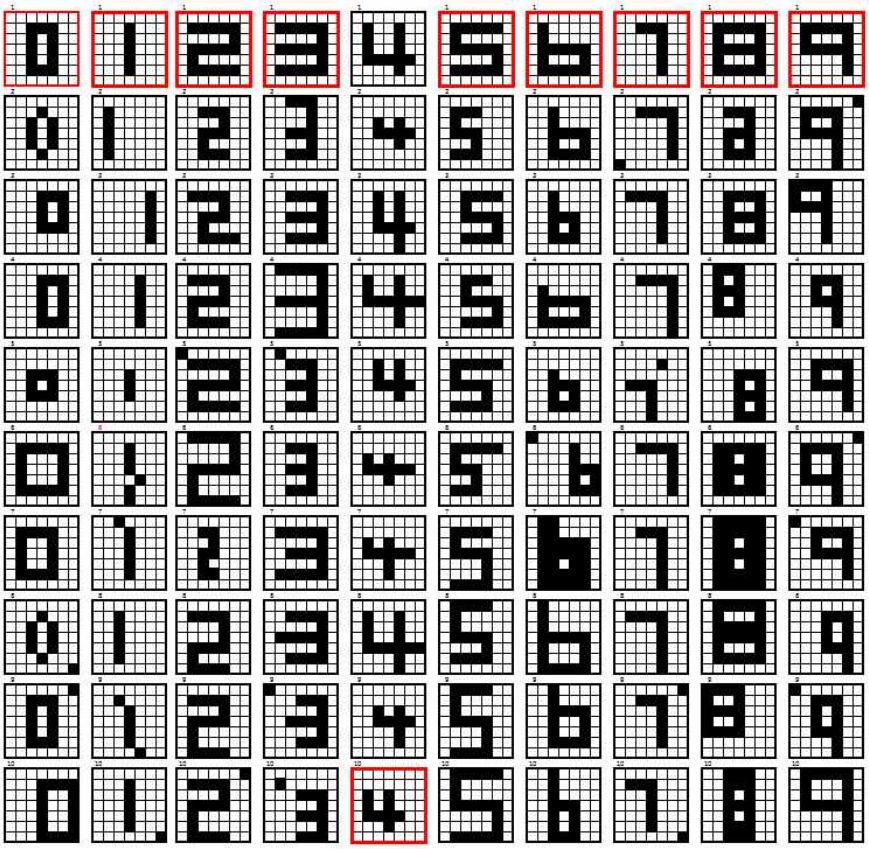
소속: 순천향대학교 의료IT공학과
성명: 20165215 위예진

-목차-

1. 서론

(a) 숫자샘플 정하기

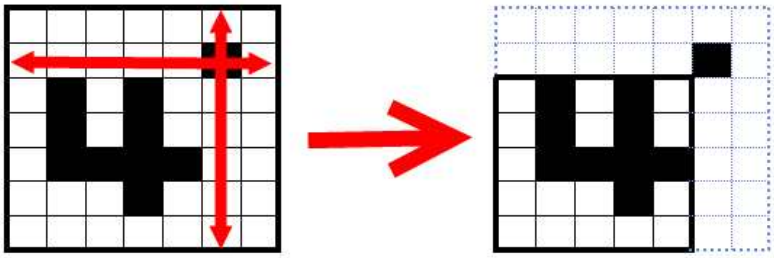
교수님의 지시에 따라 숫자샘플들은 큰 변형을 주지 않았다. 숫자 끝의 꺾임이나, 곡선 등의 특징들은 고려하지 않고 숫자 샘플을 정하였다. 그러나 두께, 위치이동, 크기에 대해서는 고려하였다. 다음 그림은 정한 숫자샘플 100개이다. 빨간색 테두리로 된 샘플들은 프로토 타입을 의미한다.



[그림 1. 정한 숫자샘플 100개]

(b) 기본이 되는 특징 구하기 & 잡음 제거

한가지 특징벡터를 제외하고, 특징벡터들을 나누는 기본이 되는 특징값은 프로젝션 방식을 통해 각 행, 열의 1의 개수를 구하였다. 의미 있는 데이터들과 인접하지 않는 잡음을 제거하기 위해 하나의 행에서의 1의 개수가 1개였을 때, 그 열의 1의 개수 또한 1이라면 잡음이라고 여겨서 특징벡터를 고려할 영역을 줄였다. 아래는 잡음을 제거하고 크기를 줄인 예시이다.



[그림2. 잡음 제거를 통한 크기 축소 예시]

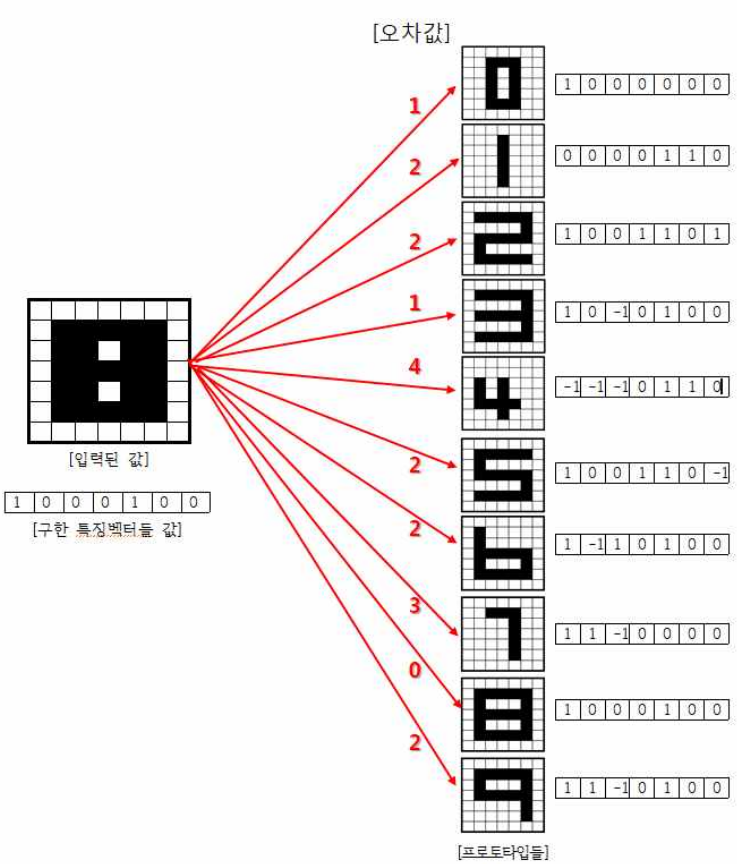
(c) 특징벡터 정하기 (7개)

프로젝션하여 얻은 열, 행의 1의 개수를 바탕으로 6가지의 특징벡터와 1의 개수로는 구분이 어려운 2, 5를 위한 1가지의 특징벡터, 즉 총 7가지의 특징벡터를 정하였다.

특징 1. 가장자리 행과 가운데 행들과의 1의 평균개수 비교
특징 2. 행을 반으로 나눠 1의 개수 비교
특징 3. 열을 반으로 나눠 1의 개수 비교
특징 4. 100% 차 있는 열이 없는지 (2, 5에 의미 있음)
특징 5. 행의 중간이 100% 차 있는지 (0, 7에 의미 있음)
특징 6. 열의 중간이 100% 차 있는지 (1에 의미 있음)
특징 7. 처음 행과 마지막 행에서 처음 0을 찾은 열 순서 비교 (2,5를 비교하기 위한 특징 벡터)

(d) 인식 방법 (프로토 타입과의 비교)

프로토타입의 특징벡터값들을 구한 후, 새로 들어온 입력값의 특징벡터들과 비교한다. 새로 들어온 입력값은 숫자 0~9까지 1개씩 총 10개의 프로토타입들과 각각 7개의 특징벡터 값을 비교를 하게 되고, 그 중에서 특징벡터값의 오차율이 가장 적은 숫자로 인식값을 내놓게 된다.

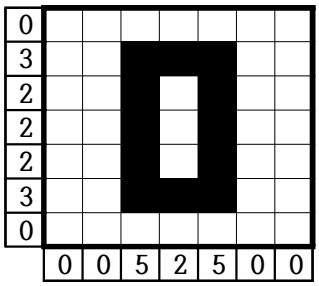


[그림 3. 프로토타입과의 특징벡터 값 비교의 예]

2. 본론

(a) 기본이 되는 특징값 : 프로젝션 방법

7개의 특징벡터 중 개의 특징벡터를 구하는 바탕이 되는 특징값이다. 프로젝션 방법으로 행, 열 각각에서의 1의 개수를 구한다. 이 값은 잡음을 제거하고 크기를 줄이는 데에도 쓰인다.

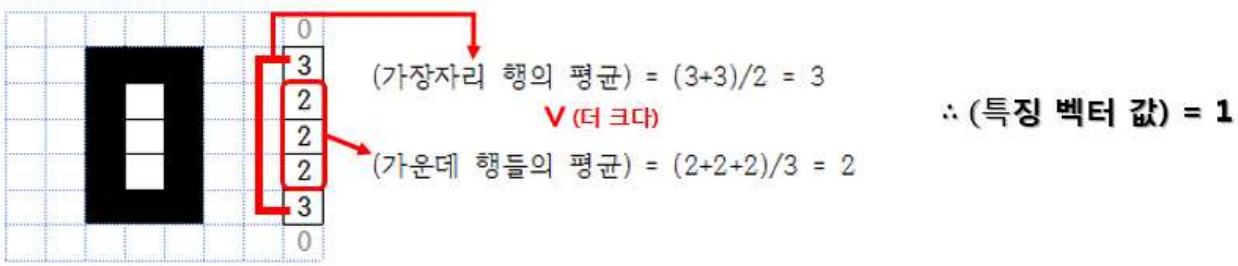


[표 1. 프로젝션 방식으로 기본 특징값을 구한 예시]

(b) 첫 번째 특징 : 가장자리 행들과 가운데 행들과의 1의 평균개수 비교

- 값 기준
  (가장자리 행의 평균) > (가운데 행들의 평균) : 1
  (가장자리 행의 평균) = (가운데 행들의 평균) : 0
  (가장자리 행의 평균) < (가운데 행들의 평균) : -1

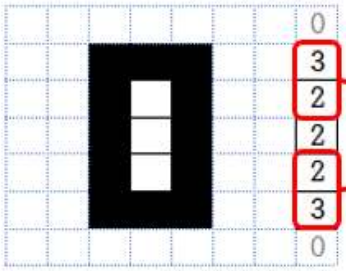
- 예시 (아래의 샘플은 잡음제거와 크기축소를 거친 상태이다.)



(c) 두 번째 특징 : 행을 반으로 나눠 1의 개수 비교

- 값 기준
  (위쪽 행들의 1의 개수) > (아래쪽 행들의 1의 개수) : 1
  (위쪽 행들의 1의 개수) = (아래쪽 행들의 1의 개수) : 0
  (위쪽 행들의 1의 개수) < (아래쪽 행들의 1의 개수) : -1

- 예시 (아래의 샘플은 잡음제거와 크기축소를 거친 상태이다.)



(위쪽 행들의 1의 개수) = 3+2 = 5

II (같다)

(아래쪽 행들의 1의 개수) = 2+3 = 5

∴ (특징 벡터 값) = 0

(d) 세 번째 특징 : 열을 반으로 나눠 1의 개수 비교

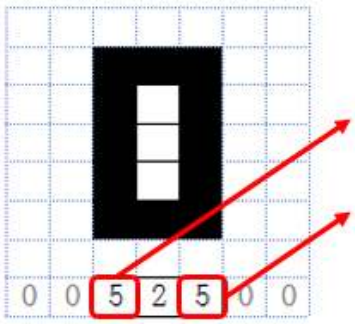
- 값 기준

  (왼쪽 열들의 1의 개수) > (오른쪽 열들의 1의 개수) : 1

  (왼쪽 열들의 1의 개수) = (오른쪽 열들의 1의 개수) : 0

  (왼쪽 열들의 1의 개수) < (오른쪽 열들의 1의 개수) : -1

- 예시 (아래의 샘플은 잡음제거와 크기축소를 거친 상태이다.)



(왼쪽 열들의 1의 개수) = 5

II (같다)

(오른쪽 열들의 1의 개수) = 5

∴ (특징 벡터 값) = 0

(e) 네 번째 특징 : 100% 차 있는 열이 없는지

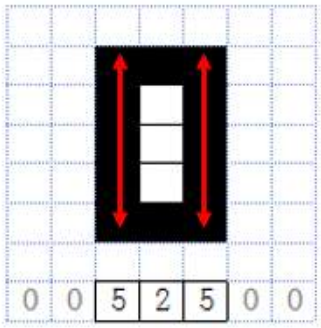
2와 5는 꽉 차 있는 열이 없으므로 둘을 인식하는 데에 도움을 줄 수 있다.

- 값 기준

  100% 차 있는 열이 없음 : 1

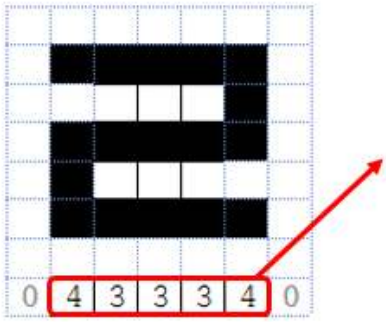
  100% 차 있는 열이 있음 : 0

- 예시 (아래의 샘플은 잡음제거와 크기축소를 거친 상태이다.)



꽉 차있는 열이 존재함 (2개 있음)

∴ (특징 벡터 값) = 0



꽉 차있는 열이 존재하지 않음
(5가 없음)

∴ (특징 벡터 값) = 0

(f) 다섯 번째 특징 : 행의 중간이 100% 차 있는지

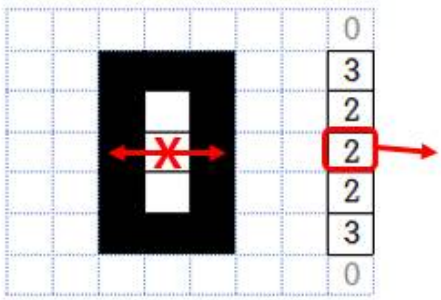
0과 7은 행의 중간이 가득 차 있지 않으므로 둘을 분류하기 위해 사용한다.

- 값 기준

  행의 중간이 100% 차 있음 : 1

  행의 중간이 100% 차 있지 않음 : 0

- 예시 (아래의 샘플은 잡음제거와 크기축소를 거친 상태이다.)



행의 중간이 꽉 차지 않음
(3이 아님)

∴ (특징 벡터 값) = 0

(g) 여섯 번째 특징 : 열의 중간이 100% 차 있는지

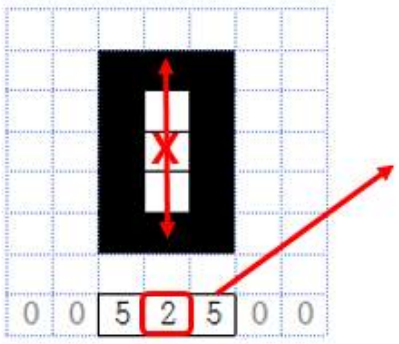
1은 열이 모두 차 있으므로 1을 분류하기 위해 사용한다.

- 값 기준

  열의 중간이 100% 차 있음 : 1

  열의 중간이 100% 차 있지 않음 : 0

- 예시 (아래의 샘플은 잡음제거와 크기축소를 거친 상태이다.)



열의 중간이 꽉 차지 않음
(5가 아님)

∴ (특징 벡터 값) = 0

(h) 일곱 번째 특징 : 처음 행과 마지막 행에서 처음 0을 찾은 열 순서 비교 (2, 5를 구분하기 위한 특징)

처음 행에서 위에서부터 아래로 0이 있는지 탐색하고, 0인 곳의 열 인덱스 확인한다. 마찬가지로 마지막 행에서 아래에서부터 위로 0이 있는지 탐색하고, 0인 곳의 열 인덱스를 확인한다. 처음 행과 마지막 행의 개수를 비교한다.

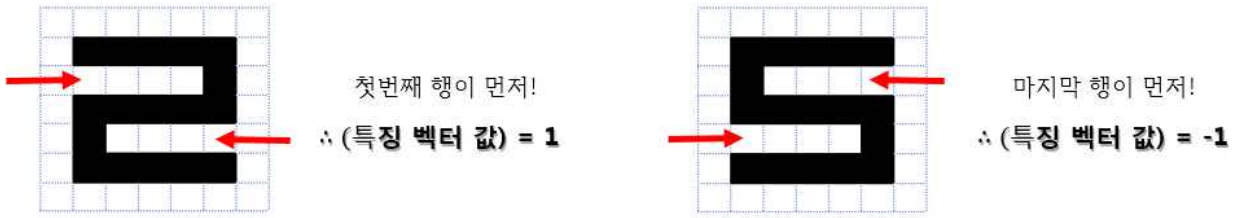
- 값 기준

(처음 행의 열 인덱스) < (마지막 행의 열 인덱스) : 1 (숫자 2)

(처음 행의 열 인덱스) > (마지막 행의 열 인덱스) : -1 (숫자 5)

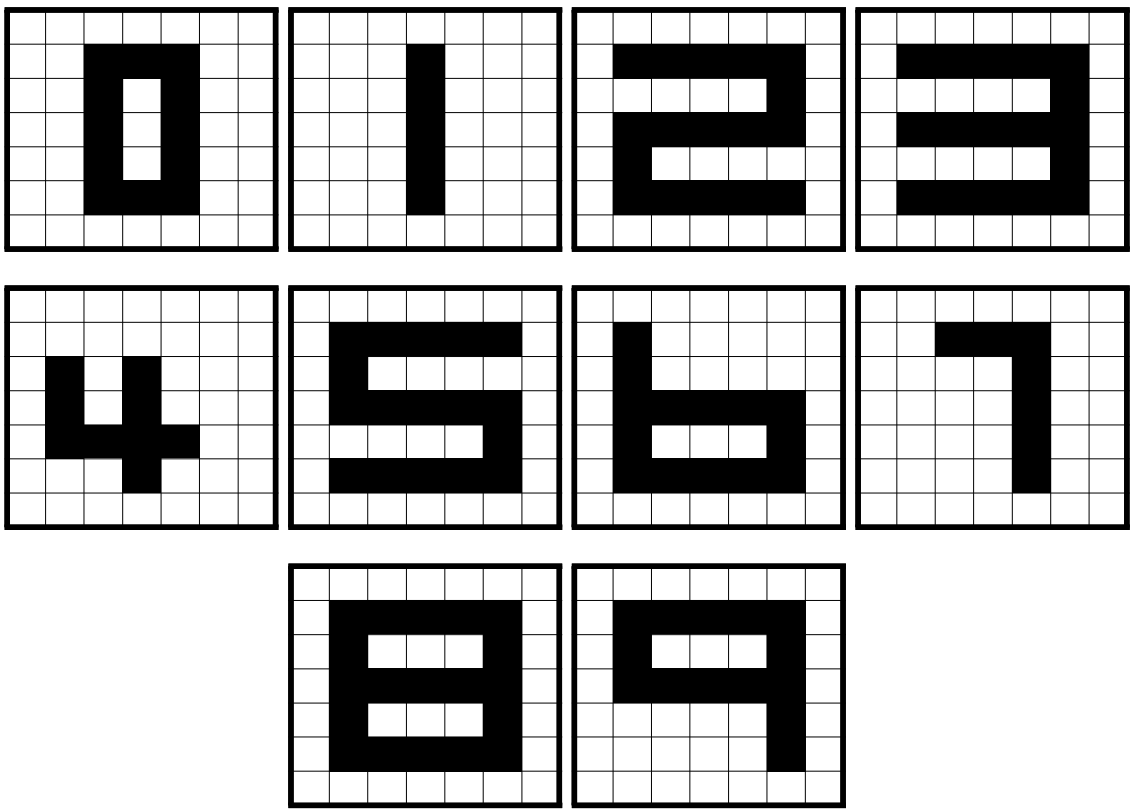
처음 행/마지막 행에서 하나라도 0을 찾지 못하거나, 둘 다 찾았어도 열의 인덱스 값이 같음 : 0

- 예시 (아래의 샘플은 잡음제거와 크기축소를 거친 상태이다.)



(i) 대표샘플 구성방법

명확하고 특징이 확실하게 해당 숫자로 여겨질 수 있는 것들을 프로토타입으로 정했다. 아래 샘플들은 숫자마다 하나씩 뽑은 프로토타입 샘플이다.



[그림 4. 숫자 0~9의 각각의 프로토타입]

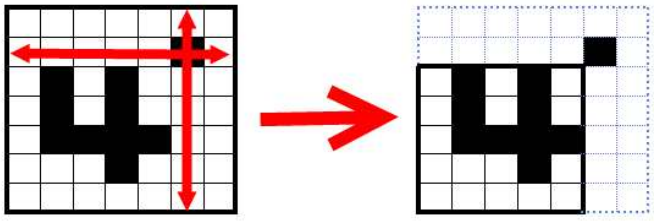
특징벡터 함수의 이름	0	1	2	3	4	5	6	7	8	9
avgCom (위아래행평균이더크면1, 작으면-1)	> (1)	=	> (1)	> (1)	< (-1)	> (1)	∇ (>)	> (1)	> (1)	∇ (>)
half (R) (위쪽이더크면1,작으면-1)	=	=	=	=	∇ (<)	=	< (-1)	> (1)	=	> (1)
half (C) (왼쪽이더크면1,작으면-1)	=	=	=	<	∇ (<)	=	> (1)	< (-1)	=	< (-1)
noFullCol (없으면1)	O (0)	O (0)	X (1)	O (0)	O (0)	X (1)	O (0)	O (0)	O (0)	O (0)
midFull (R) (차있으면1)	X (0)	O (1)	O (1)	O (1)	∇ (O)	O (1)	∇ (O)	X (0)	O (1)	∇ (O)
midFull (C) (차있으면1)	X (0)	O (1)	X (0)	X (0)	∇ (O)	X (0)	X (0)	X (0)	X (0)	X (0)
first0(2,5) (왼쪽에서찾은게더먼저임2 그러면1)	X (0)	X (0)	O (1)	X (0)	X (0)	O (-1)	X (0)	X (0)	X (0)	X (0)

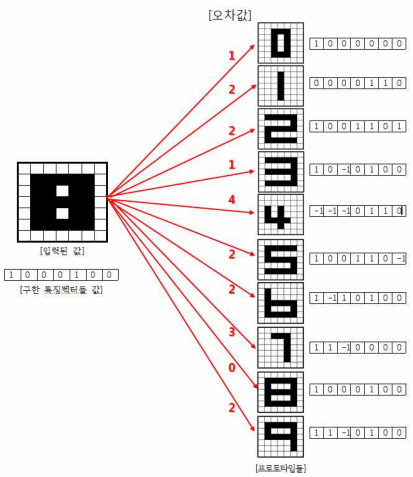
[표. 숫자별 특징벡터 값 비교]

(i) 인식방법 : 의미 있는 데이터만 확인하기 위해 잡음을 제거한 후, 크기를 줄임

먼저 정해놓은 프로토타입의 특징벡터들을 구한다. 입력된 샘플의 특징벡터들을 구한 후, 프로토타입의 특징벡터들과 비교해본다. 그 중에서 가장 오차율이 작은, 즉 가장 일치율이 높은 프로토타입의 숫자로 인식이 된다.

특징벡터들은 기본적으로 프로젝션 방식을 통해 각각의 행, 열들의 1의 개수를 기본적인 특징으로 분류하게 된다. 또한, 잡음을 제거하고 위치이동을 했을 때의 인식률을 높이기 위해 크기를 줄인다.





3. 소스코드

(a) 필요한 함수와 변수 등을 선언하는 헤더파일 (help.h)

```
#pragma once
/*
* 숫자 인식하는 곳 -> main
* 특징벡터 추출하는 곳 -> searcher
* 숫자데이터 저장하는 곳 -> numbers
*/

//필요한 매크로들
#define SORT 10          //숫자의 종류(0~9, 10개)
#define ORDER 10         //각 숫자의 샘플 개수
#define ROWS 7           //2차원 행렬의 행 개수
#define COLS 7           //2차원 행렬의 열 개수
#define FEATURE 7        //특징벡터의 수
#define PROTO 1          //프로토타입의 수

// 행/열을 반으로 나눌 때 사용하는 매크로
#define MID(x, y) (((y) - (x) + 1 ) / 2)
#define REST(x, y) (((y) - (x) + 1 ) % 2)

//정규화를 위한 함수들
void counting(int N[SORT][ORDER][ROWS][COLS], int sort, int order);              // 행/열 마다의 1개수 셈
int cross_r(int N[SORT][ORDER][ROWS][COLS], int sort, int order, int row);  // 행의 1개수의 합이 1일 때 그 1이
int cross_c(int N[SORT][ORDER][ROWS][COLS], int sort, int order, int row);  // 있는 열의 개수도 1인지 확인
void downsize(int N[SORT][ORDER][ROWS][COLS], int sort, int order);

//특징벡터들
int half(int R[ROWS], int start, int end);
int noFullCol(int C[COLS], int start, int end, int r_s, int r_e);
int midFull(int D[ROWS], int start, int end, int s, int e);
int avgCom(int R[ROWS], int row_s, int row_e);
int first0(int D[SORT][ORDER][ROWS][COLS], int i, int j, int row_s, int row_e, int col_s, int col_e);

//필요한 배열들
int inData[SORT][ORDER][ROWS][COLS];      // [숫자종류n][숫자n10개][행][열]
int protoData[SORT][ORDER][ROWS][COLS];                          // 정한 프로토타입들
int count_r[ROWS];                  // 1의 개수를 행마다 더한 행렬
int count_c[COLS];                  // 1의 개수를 열마다 더한 행렬
int featureIn[FEATURE];   // 입력된 숫자의 특징벡터의 값을 갖고 있는 행렬
int featureNum[SORT][FEATURE];   // 프로토타입들의 특징벡터의 값을 갖고 있는 행렬
int rate[SORT];                     // 프로토타입과의 일치율

// 필요한 값들
int row_s, col_s;           // 합이 0이 아닌 시작
int row_e, col_e;           // 합이 0이 아닌 끝
int max;          //일치율이 가장 높은 숫자
int numRate, totalRate;                     //숫자인식한 결과 일치율 숫자당, 전체
```

(b) 지정한 프로토타입과 들어온 숫자 넣는 소스파일 (numbers.c)

```
//* 숫자데이터 저장하는 곳 -> numbers

#include "help.h"

//숫자별 프로토타입 데이터
//SORT : 숫자종류
//ORDER : 숫자가 갖고 있는 프로토타입의 개수
//행의 값 : 숫자 순서대로
//열의 값 : avgCom, half(R), half(C), noFullCol, midFull(R),midFull(C), first0 순서
int protoData[SORT][ORDER][ROWS][COLS]={(생략)};

// [숫자종류n][숫자n10개][행][열]
int inData[SORT][ORDER][ROWS][COLS]=={(생략)};
```

(c) 특징벡터를 추출하는 함수들을 정의하는 소스파일 (searcher.c)

```
//* 특징벡터 추출하는 곳 -> searcher

#include "help.h"

void counting(int N[SORT][ORDER][ROWS][COLS], int sort, int order) {

	// 개수 초기화
	for (int i = 0; i < ROWS; i++) {
		count_r[i] = 0;
		count_c[i] = 0;
	}

	// 행마다 1의 개수를 셈
	for (int i = 0; i < ROWS; i++) {
		for (int j = 0; j < COLS; j++) {
			if (N[sort][order][i][j] == 1)
				count_r[i] += 1;
			else
				;
		}
	}

	// 열마다 1의 개수를 셈
	for (int j = 0; j < COLS; j++) {
		for (int i = 0; i < ROWS; i++) {
			if (N[sort][order][i][j] == 1)
				count_c[j] += 1;
			else
				;
		}
	}

}

// 가로, 세로의 각각의 합이 1인 행, 열도 제외(십자 중에 합이 1이면 1)
int cross_r(int N[SORT][ORDER][ROWS][COLS], int sort, int order, int row) {
	int col;

	if (count_r[row] == 1) {
		//1이 있는 열의 위치 찾음
		for (int m = 0; m < COLS; m++) {
			if (N[sort][order][row][m] == 1) {
				col = m;
				break;
			}
			else ;
		}
		//열의 1개수가 1인지
		if (count_c[col] == 1)
			return 1;
		else
			return 0;
	}
	else
		return 0;
}
int cross_c(int N[SORT][ORDER][ROWS][COLS], int sort, int order, int col) {
	int row;

	if (count_c[col] == 1) {
		//1이 있는 행의 위치 찾음
		for (int m = 0; m < ROWS; m++) {
			if (N[sort][order][m][col] == 1) {
```

```
                                row = m;
                                break;
                        }
                        else;
                }
                //행의 1개수가 1인지
                if (count_r[row] == 1)
                        return 1;
                else
                        return 0;
        }
        else
                return 0;
}

void downsize(int N[SORT][ORDER][ROWS][COLS], int sort, int order) {

        // 1의 개수합이 0인 행 인식계산에서 제외하기
        for (int i = 0; i < ROWS; i++) {
                if (count_r[i] != 0 && cross_r(N, sort, order, i) == 0) {
                        row_s = i;
                        for (int j = i; j < ROWS; j++) {
                                if (cross_r(N, sort, order, j) == 1) {
                                        row_e = j - 1;
                                        break;
                                }
                                else if (count_r[j] == 0 && cross_r(N, sort, order, j) ==
0) {
                                        row_e = j - 1;
                                        break;
                                }
                                else if (count_r[6] != 0 && cross_r(N, sort, order,
6)==0) {
                                        row_e = 6;
                                        break;
                                }
                                else
                                        ;
                        }
                        break;
                }

        }

        // 1의 개수합이 0인 열 인식계산에서 제외하기
        for (int i = 0; i < COLS; i++) {
                if (count_c[i] != 0 && cross_c(N, sort, order, i) == 0) {
                        col_s = i;
                        for (int j = i; j < COLS; j++) {
                                if (cross_c(N, sort, order, j) == 1) {
                                        col_e = j - 1;
                                        break;
                                }
                                else if (count_c[j] == 0) {
                                        col_e = j - 1;
                                        break;
                                }
                                else if (count_c[6] != 0 && cross_c(N, sort, order, 6) == 0) {
                                        col_e = 6;
                                        break;
                                }
                                else
                                        ;
                        }
                        break;
                }
```

```
	}


}

//열 개수가 양옆 대칭 인지
/*
int symmetry(int C_c[COLS], int col_s, int col_e) {
	int mid = MID(col_s, col_e) + col_s;
	int count = 0;
	//개수가 짝수이면
	if (MID(col_s, col_e) == 0) {
		for (int i = 0; i < MID(col_s, col_e); i++) {
			if (C_c[col_s + i] == C_c[col_e - i])
				count += 1;
			else ;
		}
		if (count == MID(col_s, col_e))
			return 1;
		else
			return 0;
	}

	//개수가 홀수이면
	else {

	}
}
*/

//맨위아래행과 가운데행들의 평균 1의 개수비교(위아래행 평균이 더 크면 1)
int avgCom(int R[ROWS], int start, int end) {
	double count_edge = (R[start] + R[end]) / 2.0;
	double count_center = 0.0;
	int center_num = end - start - 1;

	for (int i = 0; i < center_num; i++) {
		count_center += R[start + 1 + i];
	}
	count_center /= center_num;

	if (count_edge > count_center)
		return 1;
	else if (count_edge < count_center)
		return -1;
	else
		return 0;
}

//행,열을 반으로 나눠 1의 개수 비교(위쪽이 더 큰가?)
int half(int R[ROWS], int start, int end) {
	int count_l = 0;
	int count_r = 0;

	for (int i = 0; i < MID(start, end); i++) {
		count_l += R[start + i];
		count_r += R[end - i];
	}
	if (count_l == count_r)
		return 0;
	else if (count_l > count_r)
		return 1;
	else
		return -1;
}
```

```
//100%로 차있는 열이 없음(없으면 1)
int noFullCol(int C[COLS], int start, int end, int r_s, int r_e) {
	int count = 0;
	for (int i = 0; i < end - start + 1; i++) {
		if (C[start + i] == r_e - r_s + 1)
			count += 1;
		else
			;
	}
	if (count > 0)
		return 0;
	else
		return 1;
}

//행, 열의 중간이 100% 차있음(차있으면 1)
int midFull(int D[ROWS], int start, int end, int s, int e) {
	int mid = MID(start, end) + start;
	if (D[mid] == e - s + 1)
		return 1;
	else
		return 0;
}

//숫자 2,5 구분을 위한 앞, 뒤 뚫려있는 거 찾기(둘의 행위치 달라야 함)
int first0(int D[SORT][ORDER][ROWS][COLS], int r, int c, int row_s, int row_e, int col_s, int
col_e) {
	int left = 10;
	int right = 10;

	//왼쪽 위부터
	for (int i = 0; i < row_e - row_s-1; i++) {
		if (D[r][c][row_s + i + 1][col_s] == 0) {
			left = row_s + i + 1;
			break;
		}
	}
	//오른쪽 아래부터
	for (int i = 0; i < row_e - row_s - 1; i++) {
		if (D[r][c][row_e - i - 1][col_e] == 0) {
			right = row_e - i - 1;
			break;
		}
	}
	if (left == 10 || right == 10)
		return 0;
	else if (left < right && left != 10 && right != 10)
		return 1;
	else if (right < left && left != 10 && right != 10)
		return -1;
	else
		return 0;
}
```

(d) 프로토타입과 입력된 숫자의 특징벡터 값들을 구하고, 서로 비교함, 일치율 계산 (main.c)

```
//* 숫자 인식하는 곳 -> main

#include "help.h"
#include <stdio.h>

void main() {

	//각 숫자들의 프로토타입의 특징벡터 추출
	//열의 값
	//avgCom, half(R), half(C), noFullCol, midFull(R),midFull(C) 순서
	/*
	int avgCom(int R[ROWS], int row_s, int row_e);
	int half(int R[ROWS], int start, int end);
	int noFullCol(int C[COLS], int start, int end, int r_s, int r_e);
	int midFull(int D[ROWS], int start, int end, int s, int e);
	*/
	printf("--프로토타입의 특징벡터 배열--\n");
	for (int i = 0; i < SORT; i++) {
		counting(protoData, i, 0);
		downsize(protoData, i, 0);

		featureNum[i][0] = avgCom(count_r, row_s, row_e);
		featureNum[i][1] = half(count_r, row_s, row_e);
		featureNum[i][2] = half(count_c, col_s, col_e);
		featureNum[i][3] = noFullCol(count_c, col_s, col_e, row_s, row_e);
		featureNum[i][4] = midFull(count_r, row_s, row_e, col_s, col_e);
		featureNum[i][5] = midFull(count_c, col_s, col_e, row_s, row_e);
		featureNum[i][6] = first0(protoData, i, 0, row_s, row_e, col_s, col_e);

		printf("숫자 %d : (%d, %d, %d, %d, %d, %d, %d) \n", i, featureNum[i][0], featureNum[i][1], featureNum[i][2], featureNum[i][3], featureNum[i][4], featureNum[i][5], featureNum[i][6]);
		rate[i] = 0;
	}
	printf("\n");

	//전체 일치율 초기화
	totalRate = 0;

	//100개의 숫자들 입력
	for (int i = 0; i < SORT; i++) {
		for (int j = 0; j < ORDER; j++) {

			counting(inData, i, j);
			downsize(inData, i, j);

			featureIn[0] = avgCom(count_r, row_s, row_e);
			featureIn[1] = half(count_r, row_s, row_e);
			featureIn[2] = half(count_c, col_s, col_e);
			featureIn[3] = noFullCol(count_c, col_s, col_e, row_s, row_e);
			featureIn[4] = midFull(count_r, row_s, row_e, col_s, col_e);
			featureIn[5] = midFull(count_c, col_s, col_e, row_s, row_e);
			featureIn[6] = first0(inData, i, j, row_s, row_e, col_s, col_e);

			for (int m = 0; m < SORT; m++) {
				for (int n = 0; n < FEATURE; n++) {
					if (featureNum[m][n] == featureIn[n])
						rate[m] += 1;
				}
			}
```

```
                        for (int m = 0; m < SORT; m++) {
                                if (rate[m] > rate[max])
                                        max = m;
                        }

                        if (max == i)
                                numRate += 1;
                        else
                                ;

                        printf("%d", max);


                        //프로토타입과의 일치율, 일치숫자 초기화
                        for (int m = 0; m < SORT; m++)
                                rate[m] = 0;
                        max = 0;

                        //printf("\n");
                        printf("   ");
                }
                totalRate += numRate;
                printf("숫자 %d의 일치율 : %d\n", i, numRate*10);

                //숫자일치율 초기화
                numRate = 0;
        }
        printf("전체 숫자인식률 : %d \n\n\n\n\n", totalRate);
}
```

4. 결과

(a) 학생별 인식 후 화면 캡쳐

수업을 듣는 학생 총 33명의 학생들 중 숫자 샘플을 제출하지 않은 1명을 제외한 32명의 숫자 샘플들로 숫자 인식률을 확인해보았다. 다음 그림들은 학생 한명씩 돌려본 결과의 콘솔 캡쳐화면이다.

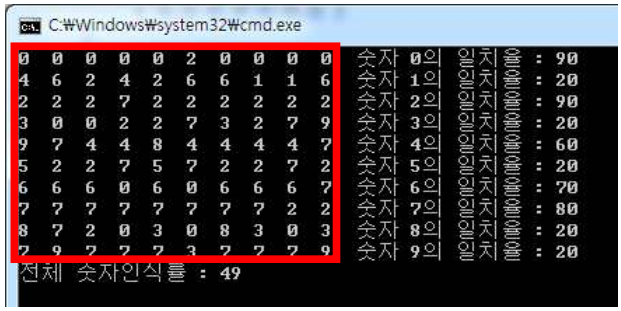
왼쪽 빨간 박스 안의 숫자들은 숫자 당 10개의 샘플들의 숫자들을 인식한 결과를 각각 나타내어 준다. [학생 1]을 예로 들면,

	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th
숫자0	0	0	0	0	0	2	0	0	0	0
숫자1	4	6	2	4	2	6	6	1	1	6
숫자2	2	2	2	7	2	2	2	2	2	2
숫자3	3	0	0	2	2	7	3	2	7	9
숫자4	9	7	4	4	4	8	4	4	4	7
숫자5	5	2	2	7	5	7	2	2	7	2
숫자6	6	6	6	0	6	0	6	6	6	7
숫자7	7	7	7	7	7	7	7	7	2	2
숫자8	8	7	2	0	3	0	8	3	0	3
숫자9	7	9	7	7	7	3	7	7	7	9

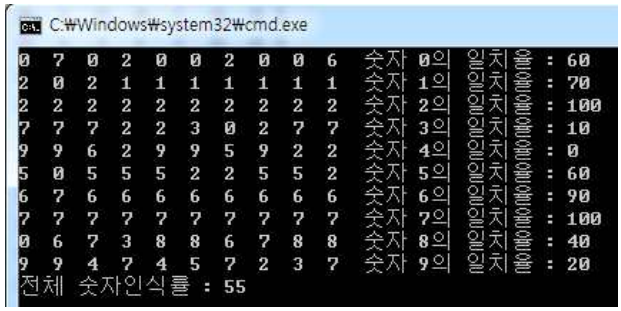
숫자 0을 10개 입력했을 때에는 6번째 데이터만 0이 아닌 2로 인식하였고, 숫자 9를 입력했을 때에는 두 번째와 10번째 데이터만 9로 인식함을 알 수 있다. (이렇게 표를 통해 어떠한 데이터가 다른 데이터로 오인식 되는지 확인할 수 있고, 이러한 오인식을 고칠 방안을 비교적 쉽게 찾을 수 있다.)

빨간 박스의 오른쪽에 각 숫자들의 인식률이 써 있고, 박스의 아래쪽에는 전체 숫자 샘플 100개를 입력하여 얻어낸 인식률을 나타낸다.

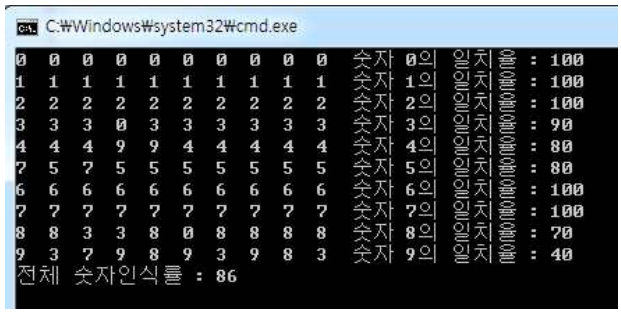
[학생 1]



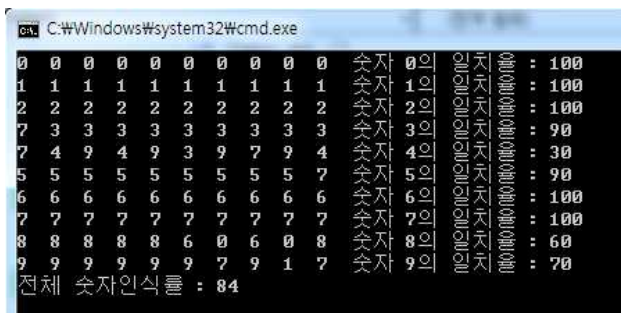
[학생 2]



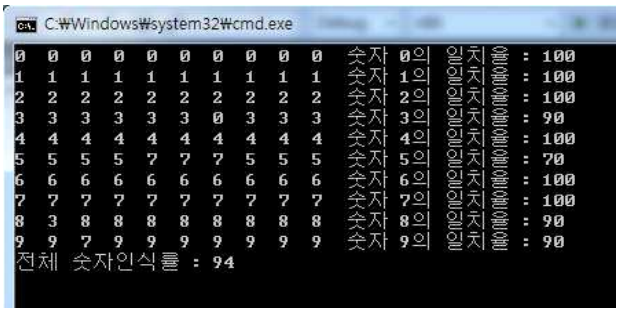
[학생 3]



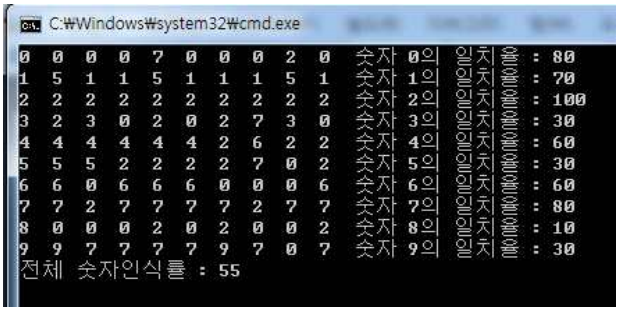
[학생 4]



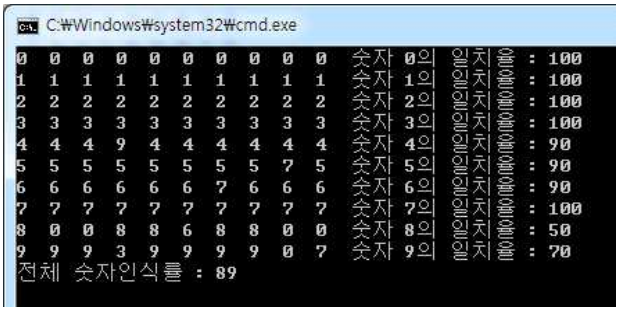
[학생 5]



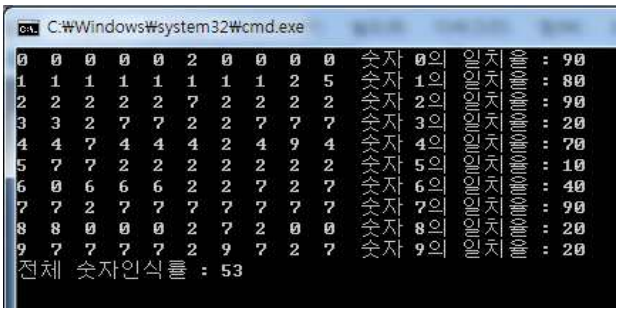
[학생 6]



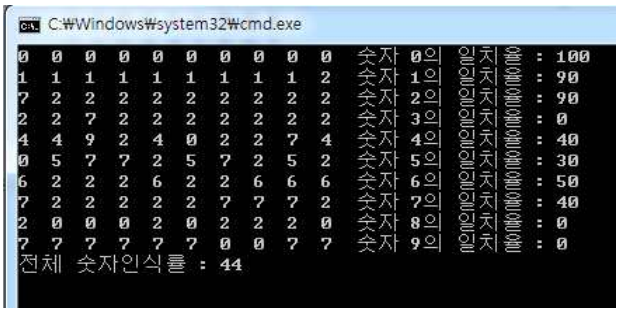
[학생 7]



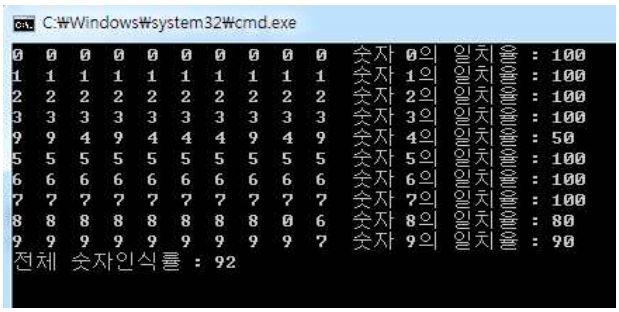
[학생 8]



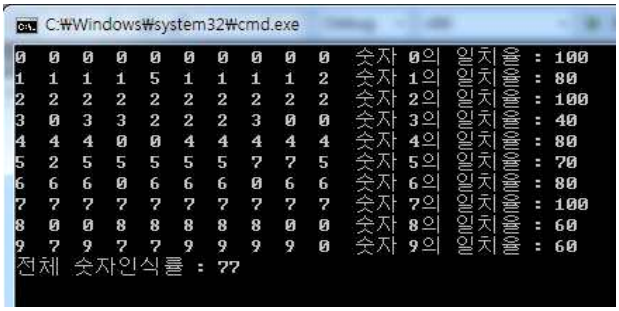
[학생 9]



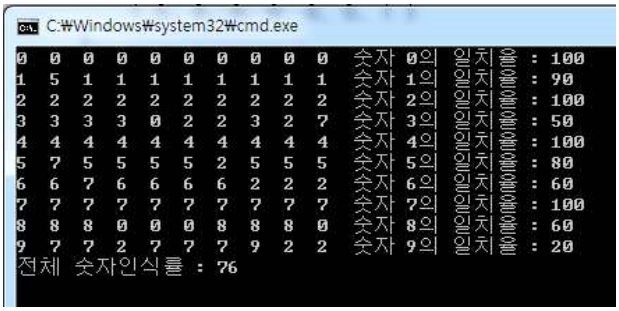
[학생 10]



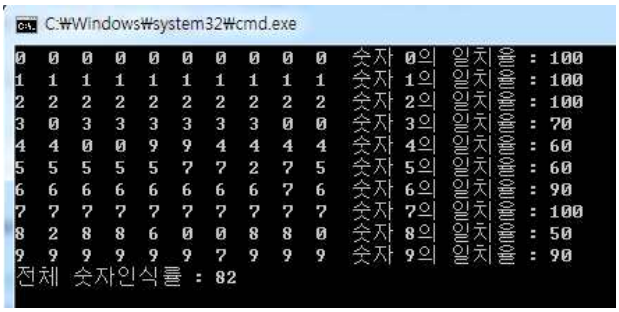
[학생 11]



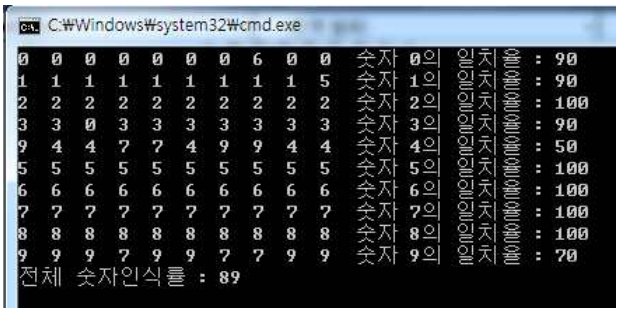
[학생 12]



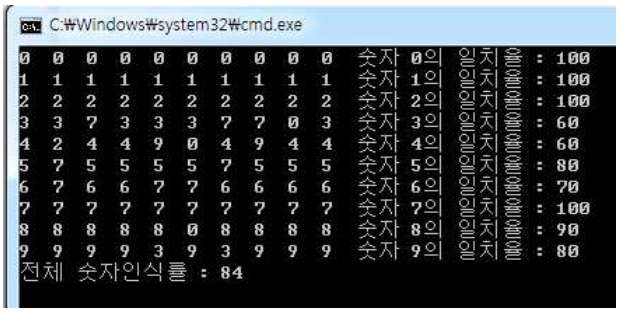
[학생 13]



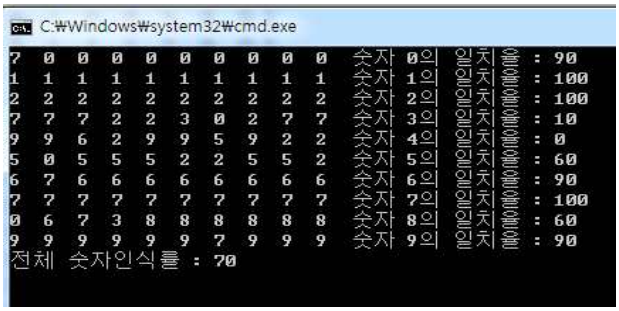
[학생 14]



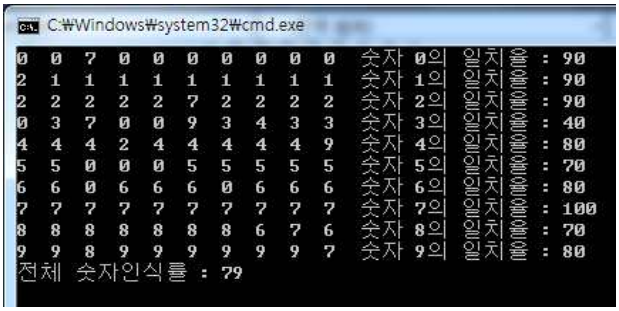
[학생 15]



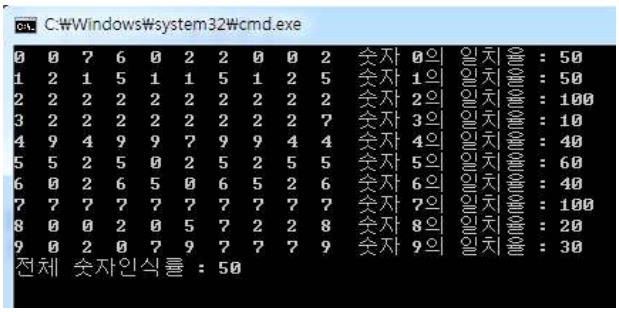
[학생 16]



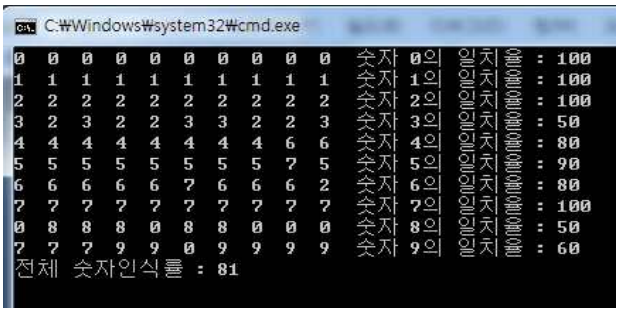
[학생 17]



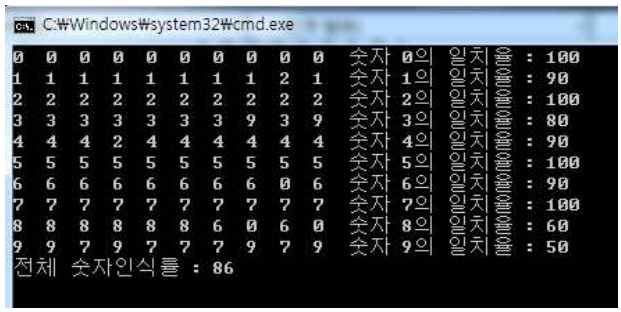
[학생 18]



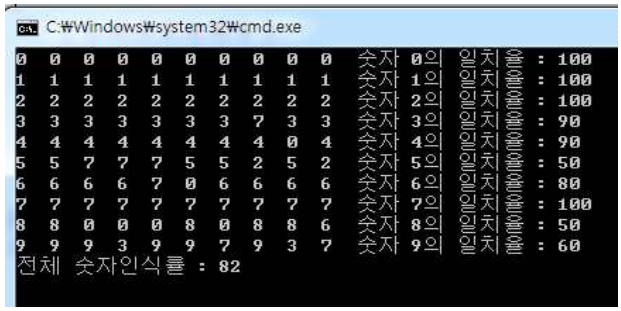
[학생 19]



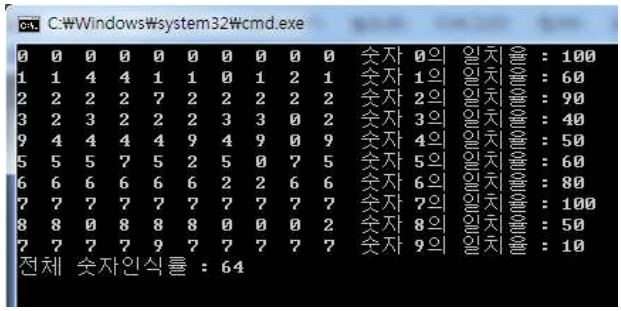
[학생 20]



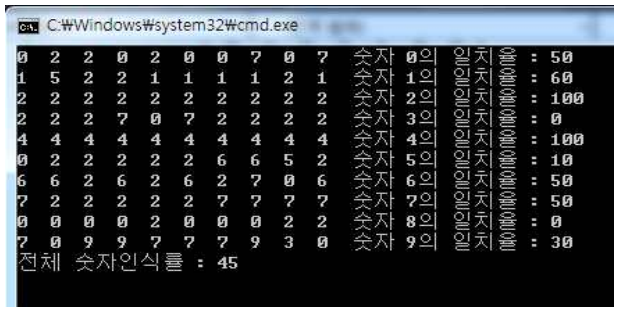
[학생 21]



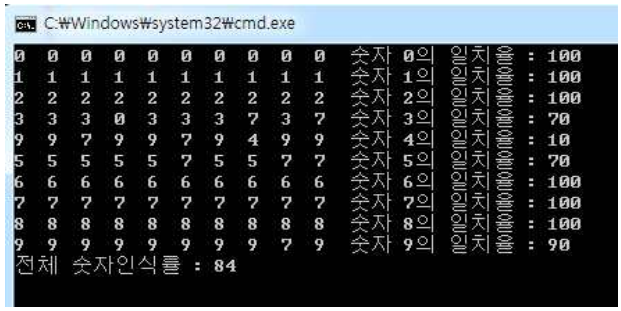
[학생 22]



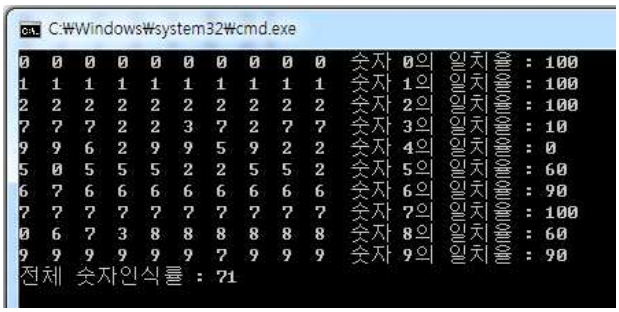
[학생 23]



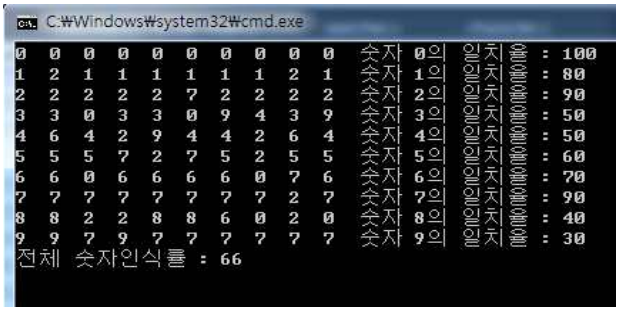
[학생 24]



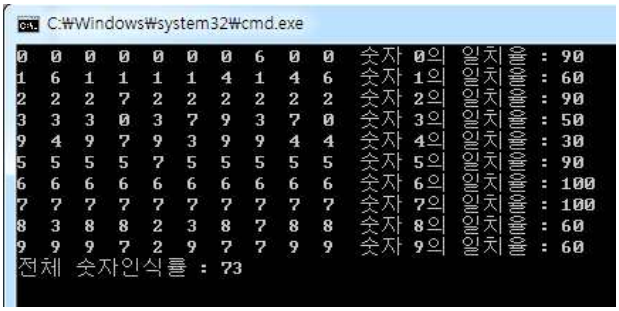
[학생 25]



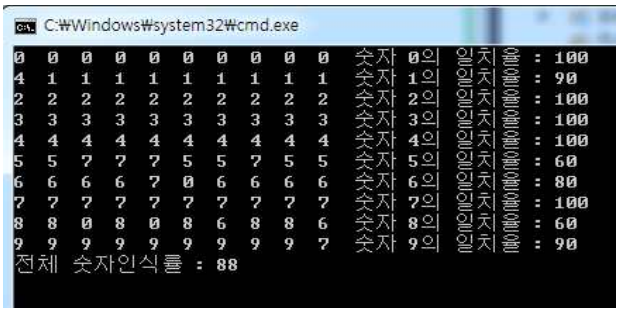
[학생 26]



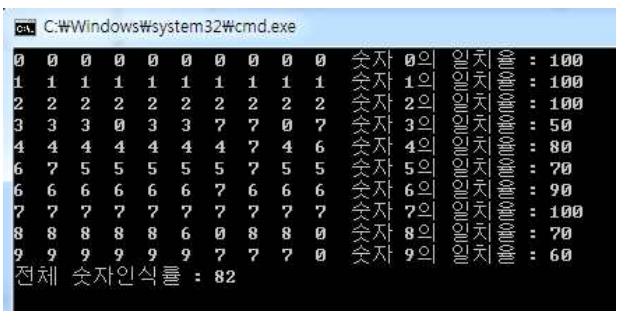
[학생 27]



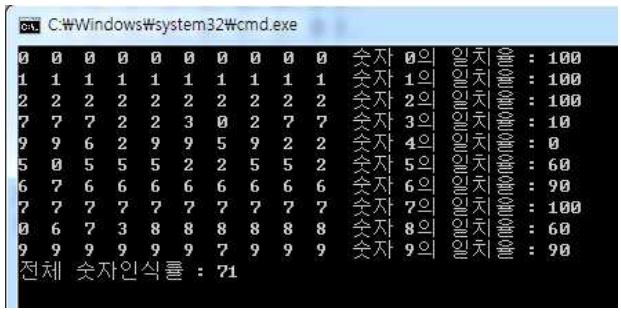
[학생 28]



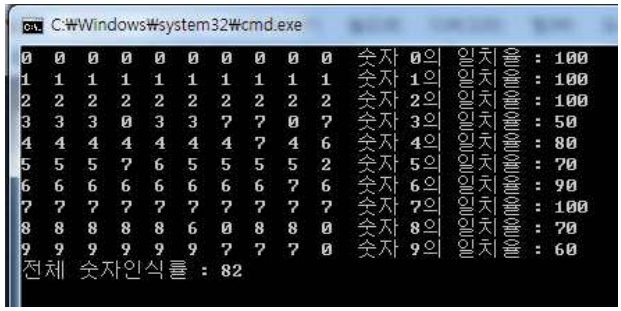
[학생 29]



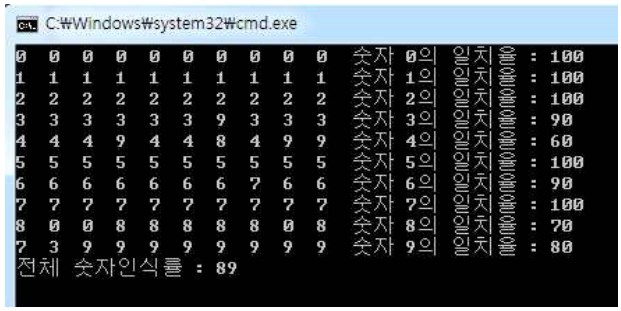
[학생 30]



[학생 31]



[학생 32]



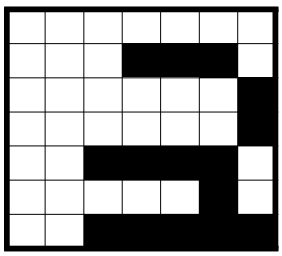
(b) 인식률 결과표 및 그래프

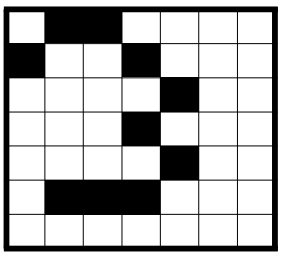
다음은 학생별 숫자 샘플의 숫자인식 인식률을 정리한 표이다. 학생 1, 5, 6, 8, 9, 18, 22, 23, 26 의 데이터에서 인식률이 현저하게 떨어짐을 알 수 있었다. 이 학생들의 데이터를 제외하고 나머지 학생들의 데이터로 총 인식률을 구한 결과, 인식률이 82.2%로 나쁘지 않은 편이었다.

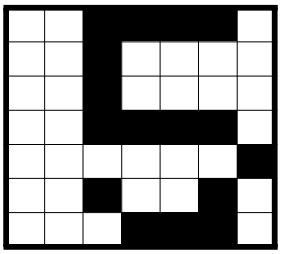
[표1. 학생별 인식율]

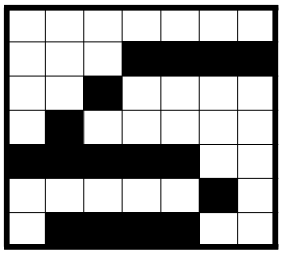
학생	0	1	2	3	4	5	6	7	8	9	총합
1	60	70	100	10	0	60	90	100	40	20	55
2	100	100	100	90	80	80	100	100	70	40	86
3	100	100	100	90	30	90	100	100	60	70	84
4	100	100	100	90	100	70	100	100	90	90	94
5	80	70	100	30	60	30	60	80	10	30	55
6	80	70	100	30	60	30	60	80	10	30	55
7	100	100	100	100	90	90	90	100	50	70	89
8	90	80	90	20	70	10	40	90	20	20	53
9	100	90	90	0	40	30	50	40	0	0	44
10	100	100	100	100	50	100	100	100	80	90	92
11	100	80	100	40	80	70	80	100	60	60	77
12	100	90	100	50	100	80	60	100	60	20	76
13	100	100	100	70	60	60	90	100	50	90	82
14	90	90	100	90	50	100	100	100	100	70	89
15	100	100	100	60	60	80	70	100	90	80	84
16	90	100	100	10	0	60	90	100	60	90	70
17	90	90	90	40	80	70	80	100	70	80	79
18	50	50	100	10	40	60	40	100	20	30	50
19	100	100	100	50	80	90	80	100	50	60	81
20	100	90	100	80	90	100	90	100	60	50	86
21	100	100	100	90	90	50	80	100	50	60	82
22	100	60	90	40	50	60	80	100	50	10	64
23	50	60	100	0	100	10	50	50	0	30	45
24	100	100	100	70	10	70	100	100	100	90	84
25	100	100	100	10	0	60	90	100	60	90	71
26	100	80	90	50	50	60	70	90	40	30	66
27	90	60	90	50	30	90	100	100	60	60	73
28	100	90	100	100	100	60	80	100	60	90	88
29	100	100	100	50	80	70	90	100	70	60	82
30	100	100	100	10	0	60	90	100	60	90	71
31	100	100	100	50	80	70	90	100	70	60	82
32	100	100	100	90	60	100	90	100	70	80	89

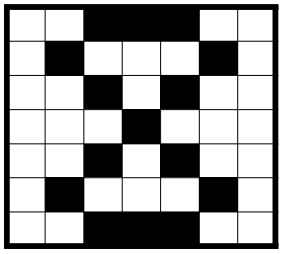
인식률이 현저하게 낮았던 학생들의 데이터를 모아서 인식률이 낮은 이유를 확인해보았다. 다음 데이터들은 인식률이 20% 이하였던 숫자의 데이터들 중 일부이다.

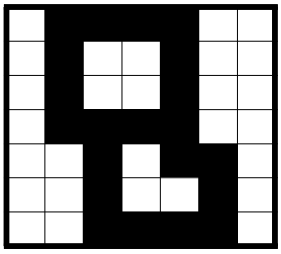

[학생 1의 숫자 3의 5번째]

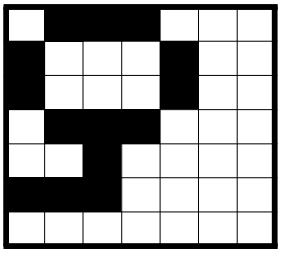

[학생 9의 숫자 3의 9번째]

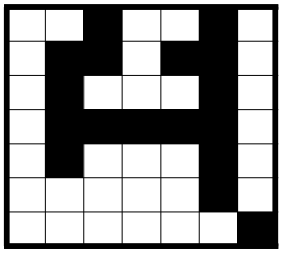

[학생 23의 숫자 5의 10번째]


[학생 8의 숫자 5의 4번째]


[학생 18의 숫자 8의 8번째]


[학생 18의 숫자 8의 6번째]


[학생 8의 숫자 9의 9번째]


[학생 1의 숫자 9의 9번째]

곡선이나 옆으로 치우쳐짐에 대한 특징들을 고려하지 않았는데, 위의 예시들과 같이 많은 변형을 준 학생들이 있어서 인식률이 떨어졌음을 알 수 있었다.

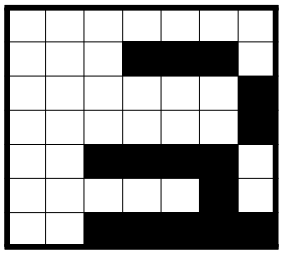
다음은 전체 숫자 인식의 인식율을 정리한 표와 그래프이다. 숫자 3, 4, 5, 8, 9의 인식률이 80% 이하로 떨어져 인식성능이 많이 좋지 않은 것으로 보인다. 나머지 숫자인 0, 1, 2, 6, 7은 인식률이 80% 이상으로 인식성능이 대체적으로 괜찮음을 알 수 있었다.

[표2. 숫자별 인식률]

구분	0	1	2	3	4	5	6	7	8	9	총합
인식률	92.8	88.1	98.1	52.2	58.4	66.3	80.6	94.7	54.3	57.5	74.31

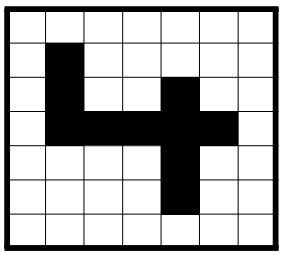
[그래프1. 숫자별 인식률]

인식률이 현저하게 낮은 3, 4, 8, 9의 인식률이 낮은 이유에 대해 알아보았다.



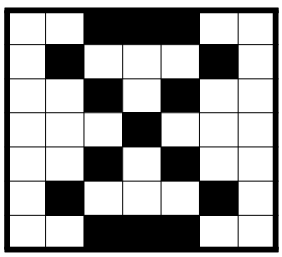
[학생 1의 숫자 3_5데이터]

숫자 3은 위아래가 대칭이고, 오른쪽이 일직선으로 쭉 이어지는 모양으로 생각했었는데 그렇지 않은 샘플들이 많았다. 열의 가장자리의 행들과 나머지들의 행들 개수는 숫자 3에게 의미 있는 특징이 아닌 것 같다.



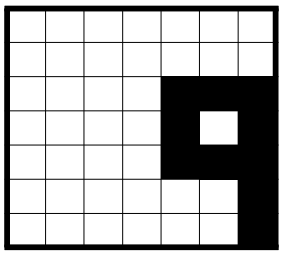
[학생 1의 숫자 4_1데이터]

숫자 4는 정한 7개의 특징 벡터들의 값 중, 4가지가 명확하지 않다. 이 문제는 프로토타입을 여러 개 둠으로써, 인식률을 높일 수 있을 것으로 보이지만, 현재는 프로토타입을 1개만 정하여서 인식률이 낮게 나온 것으로 보인다.



[학생 18의 숫자 8_8데이터]

숫자 8은 큰 변이를 고려하지 않는 오류들을 제외하더라도 네 번째 특징(꽉 차 있는 열이 없는지)와 다섯 번째 특징(행의 중간이 꽉 차 있는지) 때문에 인식에 오류가 많았다. 숫자 8은 위아래 대칭, 좌우가 대칭이지만 행의 중간이 100% 차 있지 않거나 100% 차 있는 열이 없는 경우가 많았다.



[학생 20의 숫자 9_9데이터]

숫자 9는 숫자 7로 오인식되는 경우가 많았다. 그 이유는 아무래도 숫자 9와 숫자 7에서 특징벡터의 값이 같게 나오는 특징들이 많아서 그랬던 것 같다. 본 코딩에서는 특징벡터의 값이 프로토타입과 일치하는 개수가 많은 것으로 숫자를 인식하는데, 만약 개수가 같다면 더 작은 숫자로 인식이 되는 점에서 문제가 생기는 것 같다. 숫자 2와 5를 분류하기 위해 특징벡터를 만든 것처럼 숫자 9와 7을 구분하기 위한 특징벡터를 하나 따로 만들었어야 하는 것 같다.

5. 결론

아날로그 숫자에서 큰 변형이 없는 것들로 숫자 샘플을 만들었지만. 수집된 다른 학생들의 데이터들에 곡선, 꺾임 등의 큰 변형들이 들어가 있어 인식률이 매우 낮아졌다. 설정한 특징 벡터들도 생각보다 의미 있는 특징들이 아니어서 숫자들을 분류하는 것이 쉽지 않았다.

또한, 숫자 7, 9처럼 서로 포함하게 되는 숫자들도 있기 때문에 특징벡터가 같게 나오는 경우가 꽤 있었다. 이를 방지하기 위해 입력한 숫자가 무엇인지 알아볼 때에 8, 6, 9, 0, 3, 2, 4, 5, 7, 1의 프로토타입 순서로 비교를 해서 확인을 해야할 것 같다.

이번에는 시간관계상 프로토타입을 하나밖에 지정하지 못했지만, 추후에 프로토타입을 여러 개 두어 인식률을 높일 수 있을 것이다.