論文97-34C-2-8

거리 근사를 이용하는 고속 최근 이웃 탐색 분류기에 관한 연구

(Study on The Fast Nearest-Neighbor Searching Classifier Using Distance Approximation)

李 日 完 *, 蔡 洙 翊 *

(Eel-Wan Lee and Soo-Ik Chae)

요 약

본 논문에서는 탐색 과정에서의 연산량을 줄인 최근 이웃 탐색(nearest neighbor searching) 방법의 분류기를 제시한다. 제안한 분류기에서 클래스는 참조 클래스와 비참조 클래스로 나뉜다. 제안한 분류기는 입력과 클래스간의 거리를 클래스간의 거리를 이용하여 거리를 근사하는 방법을 통하여 연산량을 감소한다. 제안한 분류기는 단지 참조 클래스와 입력간의 거리만을 계산한다. 주어진 분류기가 있을 때에 이에 해당하는 RCC (Reduced Computational Complexity) 분류기는 참조 클래스의 집합을 선정함으로써 설계할 수 있다. 참조 클래스 집합의 선정은 가능한 한 적은 오분류율의 증가와 함께 연산량을 최대로 감소할 수 있도록 이루어져야 한다. NIST database에서의 필기체 숫자 인식기용 분류기를 RCC 분류기로 설계하였다. 실험 결과 RCC분류기는 기존의 최근 이웃 탐색 분류기에 비해 0.5%의 추가 오분류율을 보였으며 연산량은 60%를 감소할 수 있었다.

Abstract

In this paper, we propose a new nearest-neighbor classifier with reduced computational complexity in search process. In the proposed classifier, the classes are divided into two sets: reference and non-reference sets. It reduces computational requirement by approximating the distance between the input and a class with the information of distances among the classes. It calculates only the distance between the input and the reference classes. We convert a given classifier into RCC(reduced computational complexity) classifier by selecting a proper set of the reference classes. In this procedure, it is needed to find a good set of reference classes which can provide large reduction in computational complexity but small increase in misclassification probability of its corresponding RCC classifier. We designed RCC classifiers for the recognition of digits from the NIST database. We obtained an RCC classifier with 60% reduction in the computational complexity with the cost of 0.5% increase in misclassification probability.

I. 서 론

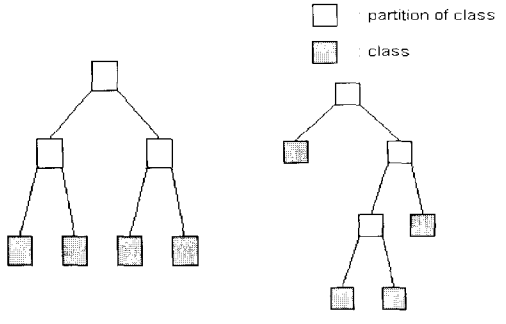
최근 이웃 탐색 분류기(nearest neighbor searching classifier)는 그 구조와 학습 방법이 다른 분류기에 비해 단순하여 일찍부터 여러 분류 문제에 사용되었던 분류기 중의 하나이다[1]. 이러한, 최근 이웃탐색 분류기의 분류 과정은 최근에 영상/음성 압축등에 많이 사용되고 있는 벡터 양자화의 부호화(encoding) 과정과 많은 점에서 유사하다. 이 분류기에서의 분류과정과 벡터 양자화기의 부호화 과정의 공통적인 문제점은 클래스의 갯수나 코드북의 크기에 비례하는 연산량에 있다. 이러한 문제는 하드웨어 구현시 병렬 처리를 매우 효율적으로 이용하여 해결할 수 있으나, 이를 단일 범용 프로세서로 구현할 때에 보다 효율적으로 최

* 正會員, 서울大學校 工學大學 電氣工學部
(School of Electrical Engineering, Engineering College, Seoul National Univ.)

接受日字:1995年7月3日, 수정완료일:1997年1月27日

근 이웃 탐색 분류기를 구현하는 것이 필요하다.

최근의 고속 벡터 양자화 기법에 관한 연구보고 [2] [3] [4] [5]에 의하면, 각 코드간에는 어느 정도의 상호 연관성이 있기 때문에 주어진 코드를 모두 탐색하지 않고도 그 거리가 가장 작은 코우드를 찾을 수 있는 고속 탐색 방법들이 가능함을 알려준다. 이와 같은 고속 탐색 방법은 그 과정이 비슷한 최근 이웃 탐색 분류기에도 그대로 적용할 수 있음을 알 수 있다. 그러나, 대부분의 이와 같은 고속 탐색 방법들은 그 탐색 시간이 일정하지 않다는 문제를 가지고 있다. 현재까지의 연구로는 항등 탐색 시간을 가지면서 고속 탐색을 하기 위해서는 어느 정도의 인식률 감소를 감수해야만 한다. 대표적인 항등 탐색 시간을 가지는 구조로 계층적 분류 방법(hierarchical classification)이 가장 많이 사용된다[5]. 그러나, 계층적 분류 방법도 항등 탐색시간을 갖기 위해서는 그림 1에서처럼 같은 계층의 모든 벡터 분할 영역(partition)이 같은 수의 원소를 가져야 한다는 제약조건이 있다[1].



(a) Constant search time guaranteed (b) Search time is variable

그림 1. 계층적 분류기에서의 탐색시간
Fig. 1. Search time of the hierarchical classifier.

본 논문에서는 이와 같은 계층적 분류 방법이외에 [2] [3] [4]등의 연구 결과에서 밝혀진 것처럼 클래스들간의 거리에서 추가의 잉여 정보를 이용할 수 있는 점에 착안한 또 다른 항등 탐색 시간을 가지는 고속 탐색 방법을 2장에서 제시한다. 제안한 방법이 효율적이기 위해서는 이미 설계된 분류기의 탐색 시간을 줄일 때에 가능한 그 인식율의 저하를 최소화할 수 있는 방법론을 제시할 수 있어야 한다. 이를 위해 3장에서는 기존의 분류기가 주어졌을 때에 이 제안된 분류기를 설계하는 알고리듬과 이의 타당성에 대해 설명한다. 4장에서는 제안한 고속 탐색 방법을 간단한 숫자와 문자에 대한 인식기에 적용하여 모의 실험을 통해 여러 가지 특성을 파악한다. 보다 실용적인 응용 예로는 NIST 데이터베이스 HWDB-1에서 추출한 숫자를 분류하는 인식기에 적용한다. 제안한 고속 탐색 방법은 5장에서 살펴보듯이 여러 분야에서 적용될 수 있다.

Ⅱ. 제안하는 고속 탐색 구조

기존의 최근 이웃 탐색 분류기 구조는 그림 1에서처럼 각 클래스로부터의 거리를 구하는 거리 함수를 구현한 부분과 각 거리 함수로부터 계산된 거리 중 가장 작은 값을 가지는 클래스를 출력하는 minNET 부분으로 구성된다. 이와 같은 최근 이웃 탐색 분류기에서 각 클래스로부터의 거리는 각 클래스마다, 하나 또는 그 이상의 대표 벡터(representation vector), c_i, 로부터의 거리로 계산하는 것이 일반적이다. 입력 x에 대해 한 클래스의 대표 벡터 c_i로부터 구한 거리를 $d(x, c_i)$라고 하면, 이 $d(x, c_i)$로부터 얻어진 거리에는 클래스 c_i로부터의 거리 이외에도 다른 클래스로부터의 거리에 관한 정보가 포함되어 있을 수 있다. 그러나, 그림 2에서와 같은 구조에서는 이와 같은 잉여 정보를 사용하는 것이 매우 어렵다.

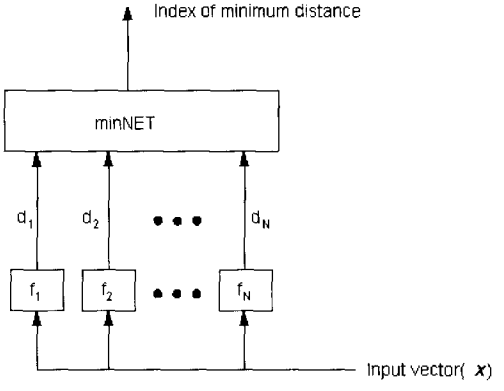


그림 2. 최근 이웃 탐색 분류기의 기존 구조
Fig. 2. NN-Classifier with conventional architecture.

본 논문에서는 한 클래스로부터의 거리에 들어있는 정보를 분류에 최대한 활용할 수 있도록 하기 위해 그림 3에서와 같이 고속 탐색 부분을 추가한 구조의 분류기로 RCC(Reduced Computational Complexity)

분류기를 제안한다. RCC 분류기에서는 모든 클래스로부터의 거리를 구하지 않는다. 클래스는 입력벡터가 인가되었을 때에 입력과의 거리를 계산하는 참조 클래스와 참조 클래스로부터의 거리로부터 해당하는 거리를 근사하는 비참조 클래스로 나뉘어 서로 다른 방법으로 처리된다. 이 분류기는 우선 주어진 클래스 개수(N)보다 적은 개수(M)의 참조 클래스(reference class), $S=\{r_1, r_2, \cdots, r_M\}$, 로부터의 거리만을 구하고 그 이외(N-M) 비참조 클래스로부터의 거리는 이미 구한 M개의 참조 클래스로부터의 거리를 이용하여 근사한다. M개의 참조 클래스들은 정확히 입력벡터와 자신간의 거리를 알고 있으므로 minNET$_1$은 이들 거리를 이용하여 최소 거리를 가지는 클래스인 r_w를 구한다. 마찬가지로 minNET$_2$는 근사된 거리로 (N-M)개의 비참조 클래스 중에서 최소 거리를 가지는 클래스인 c_w를 선정한다. 최종 minNET$_0$은 이 두개의 클래스 $\{r_w, c_w\}$에 대해 거리를 계산한 후에 그 중 작은 거리를 가지는 클래스를 최종 출력으로 출력한다.

여기에서 우리는 최종 출력을 구할 때에 정확히 계산한 거리나 거리의 근사값 중에서 하나를 선택하여 사용할 수 있다. 이와 같은 구조의 분류기를 사용하였을 때의 분류기의 연산량과 필요로 하는 메모리에 대해 각각 2.1절과 2.2절에서 설명한다. 단, 입력으로는 실수나 또는 이진수로 K차원의 벡터를 가정한다. 또한 참조 클래스의 갯수는 M이고 전체 클래스의 갯수는 N이다.

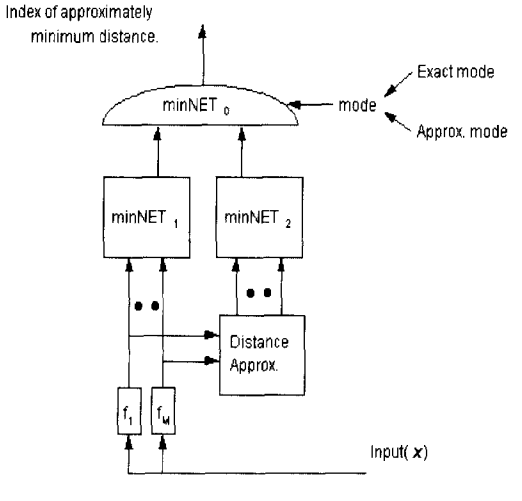


그림 3. 제안하는 구조의 분류기
Fig. 3. Classifier with the proposed architecture.

1. 필요로 하는 연산량

앞의 가정을 따르면 한 클래스로부터의 거리를 구하는 함수 $f_i(x)$의 연산량은 대부분 K에 비례한다. 이를 $\alpha_1 \cdot K$라 하자. 이 경우에 기존 구조의 분류기에서 총 필요한 연산량은 $\alpha_1 \cdot N \cdot K$가 된다. 새로운 구조의 분류기에서 필요로 하는 연산량을 살펴보면 우선 참조 클래스로부터 거리를 구하는 데에는 한 클래스당 $\alpha_1 \cdot K$의 연산량이 필요하므로 $\alpha_1 \cdot M \cdot K$의 연산량이 필요하다. 또한 나머지 클래스들로부터의 거리를 정하는 데에 필요한 연산량은 3. 1절에서 있는 식 (3)에서 볼 수 있듯이 대략 한 클래스 당 $\alpha_2 \cdot M$으로 참조 클래스의 개수에 비례한다고 볼 수 있다. 나머지 모든 비참조 클래스로부터의 거리를 구하는 데에는 $\alpha_2 \cdot M \cdot (N-M)$의 연산량이 필요하다. 만약 최종 분류 단계에서 c_w에 대해 정확한 거리 계산을 할 경우에는 추가로 $\alpha_1 \cdot K$만큼의 연산량이 필요하다. 이 세 항의 합이 제안한 구조의 분류기가 요구하는 총 연산량으로 $\alpha_1 \cdot (M+1) \cdot K + \alpha_2 \cdot M(N-M)$ 이다. 결국 제안한 분류기의 구조를 사용하였을 경우의 속도 증가는

$$\frac{1}{\frac{\alpha_1 \cdot (M+1) \cdot K + \alpha_2 \cdot M(N-M)}{\alpha_1 \cdot N \cdot K}} \approx \frac{1/M}{\frac{1}{N} + \frac{\alpha_2}{\alpha_1} \cdot \frac{1-M/N}{K}} \tag{1}$$

로 주어진다.

이와 같은 새로운 구조의 분류기를 적용하기에 특히 유리한 응용 조건들을 살펴보면 다음과 같다. 우선 클래스의 개수가 매우 많으나 실제로는 이들 사이에 매우 높은 관련이 있어 고유 공간 차원(intrinsic dimension)[6]이 낮아 적은 개수의 참조 클래스로도 주어진 인식률 저하 범위 이내에 들어올 수 있는 경우이다. 또 다른 경우로는 하나의 클래스로부터의 거리를 구하는 연산량 $\alpha_1 \cdot K$가 거리 근사에 필요한 $\alpha_2 \cdot M$에 비해 매우 큰 경우이다. 이는 문자인식의 경우 raw data를 입력으로 사용할 때에 흔히 발생한다.

2. 필요로 하는 메모리

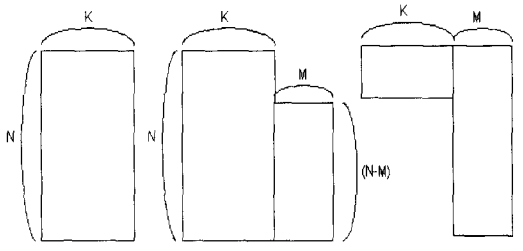
본 논문에서 제안한 방법과 같이 클래스간의 거리의 성질을 이용하여 이들을 미리 저장하는 고속 탐색 방식들의 단점으로 지적되는 것은 기존의 분류기나 코우드 북(Code Book)이외에 클래스간의 거리를 저장하기 위해 추가의 메모리가 필요하다는 것이다. 그러나, 이는 4장에서 실험결과로 보이겠지만, 최종 분류 단계에

서 정확한 거리 대신 거리 근사값을 이용하여 이와 같은 메모리 증가의 문제는 최소화할 수 있다. 그러나, 이 경우 어느 정도 오분류율의 증가를 감수해야 한다.

최종 분류 단계에서 비참조 클래스에 대한 거리 근사값만을 사용하면 N-M개의 클래스는 결국 기존의 분류기에서 필요로 했던 대표 벡터를 가지지 않아도 되므로 우선적으로 $K\cdot(N-M)$만큼의 메모리를 절약하게 된다. 여기에 앞에서 클래스간의 거리를 저장함으로써 필요한 메모리, $M\cdot N$을 제하면,

$$K\cdot(N-M)-N\cdot M \tag{2}$$

만큼의 메모리를 연산량과 함께 감소할 수 있게 된다.



(a) conventional (b) proposed method (c) proposed method
with exact mode with approx. mode

그림 4. 세 가지 분류 방법의 메모리 요구 사항
Fig. 4. Memory requirements in three classification methods.

III. 제안한 구조의 분류기 설계 방법

앞에서 제시한 고속 탐색 방법을 적용한 분류기는 서론에서도 밝혔듯이 항등 고속 탐색 알고리듬으로 일정 정도의 인식률의 감소를 감수해야한다. 즉, M이 적으면 적을 수록 탐색시간은 줄어드나, 이는 결과적으로 인식률을 지나치게 저하시킬 소지가 있다. 이에 대해서는 3.1절에서 보이는 비교적 간단하지만 구체적인 예로 그 경향을 파악한다. 이 예에서 또한 총 N개의 클래스 중에서 M(=1, ···, N) 개의 클래스를 선정함에 있어서 2^N개의 조합이 있을 수 있으며, 어떤 참조 클래스가 선정되느냐에 따라 거리 근사(distance approximate)의 정확도가 달라진다는 것도 알 수 있다. 거리 근사의 정확도는 결국 인식률에 영향을 미친다. 따라서, 제안한 고속 탐색 분류 방법의 설계(최적화) 문제는 다음과 같이 정의될 수 있다.

제안한 항등 고속 분류기의 최적화 문제의 정의

"허용할 수 있는 추가 오분류율의 최대치 ε이 주어졌을 때에 이를 만족하는 원소의 수, M,을 최소화하는 참조 클래스의 부분 집합 $S=\{r_1, r_2, \cdots, r_M\}$을 전체 클래스 대표 벡터 집합 $U=\{c_1, c_2, \cdots, c_N\}$으로 부터 구하라."

제시한 문제를 풀기 위해서는 우선적으로 참조 클래스의 집합, S, 가 주어졌을 때에 이에 해당하는 추가 오분류율을 추정하는 것이 필요하다. 이와 같은 추가 오분류율은 앞에서 언급한 것과 같이 거리 근사의 정확도에 영향을 받으므로 제안하는 거리 근사식의 특성을 파악해야 한다. 따라서, 3.1절에서는 본 논문에서 전체의 탐색 시간에서 적은 비율만을 차지할 수 있도록 거리 근사식을 제안하고 이의 특성을 파악한다. 제안된 거리 근사식에 의해 설계된 고속 탐색 방법의 성능을 우리는 3.2절에서 인식률 저하를 척도로 해서 분석적으로 살펴본다. 그러나, 추가 오분류율의 추정치 ΔP_E는 각 참조 클래스간의 상호연관 관계에 의해 총 2^N개의 해 공간에 대해 모두 계산을 해야만 최적값을 얻을 수 있다. 이를 해결하기 위해서 3.3절에서는 greedy algorithm에 기반한 참조 클래스 선정 방법을 제안하여 적은 시간 내에 효율적으로 준최적해를 구할 수 있는 방법을 제안한다.

1. 클래스로부터의 거리 추정 방법

본 논문에서 참조 클래스로부터의 거리를 이용하여 나머지 거리를 추정하는 방법으로 쓴 일반식은 (3)과 같다.

$$\hat{d}(c_i, x)=\sqrt{\frac{\sum_{j=1}^{M}(d(r_j, x)-d(r_j, c_i))^2}{M}} \tag{3}$$

단, S는 참조 클래스의 집합이고 $M=n(S)$이다.

이 식은 주어진 참조 클래스로부터 나머지 클래스까지의 거리 $d(r_i, c_j)$를 모두 저장한 후에 참조 클래스와 입력간의 거리 $d(c_i, x)$와의 차이를 각 클래스로부터의 거리로 추정한다. 이렇게 정한 거리 추정식은 원래의 정확한 클래스의 중심에 대해서는 0에 가까운 값을 나타냄을 알 수 있다. 이 성질은 식 (3)이 거리 추정식으로 쓰이기 위해 필요한 기본적인 성질이다. 그러나, 그 이외의 다른 성질들은 식 (3)의 비선형성 때문에 분석이 그리 용이하지 않다. 따라서, 본 논문에서는 (3)이 비교적 작은 입력공간에서의 잡음에 대해서는 올바른

거리 추정을 할 수 있는 특성이 있음을 아래와 같이 구체적인 예를 들어 각 클래스의 영역의 변화를 살펴봄으로써 확인하였다.

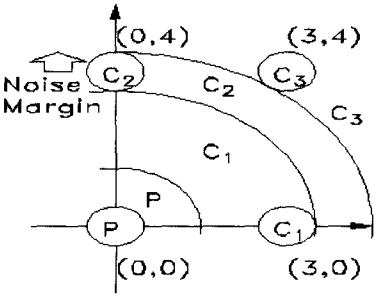


그림 5. 참조 클래스가 P일 경우의 분류 영역
Fig. 5. Classification region for the reference class P.

그림 5는 4개의 클래스가 각각 {(0,0), (3,0), (0,4), (3,4)}에 중심을 갖고 일정 반경을 갖는 클래스들이 있을 때, 이를 클래스 P로부터의 거리만을 이용해 거리 추정을 하고 그 추정된 거리로부터 구한 각 클래스의 영역이다. 각 클래스 영역은 클래스 P를 중심으로 동심원을 그리고 있다. 이때에 분류기 전체에서의 노이즈 마진은 그림 5에 나타낸 것처럼 클래스 3에 의해 결정된다. 또한 클래스 P 하나로부터의 거리만으로 거리를 추정하여 얻은 분류기의 노이즈 마진은 원래의 분류기에 비해서 많이 줄었음을 알 수 있다.

그림 6은 두개의 참조 클래스를 이용하여 거리 추정을 하였을 경우에 생길 수 있는 경우를 나타낸 것이다. 그림 6-(a),(b),(c) 각각은 참조 클래스로 {P, C_1} {P, C_2} {P, C_3}을 선택하였을 때의 클래스 영역을 나타낸다. 어느 것이든 간에 하나의 클래스로부터 추정한 경우인 그림 6에 비해서 노이즈 마진이 늘었음을 알 수 있다. 특히, 참조 클래스로 {P, C_1}이나 {P, C_2}를 선정한 경우에는 원래의 클래스 영역과 매우 닮았음을 알 수 있다.

이상에서 식 (3)을 이용하면 비교적 그 크기가 적은 입력 잡음에 대해서는 올바른 거리 추정이 가능함을 알았다. 또한 참조 클래스의 개수가 늘어날수록 제안된 구조의 분류기는 보다 나은 성능을 보일 수 있을 것을 예측할 수 있다. 그리고, 같은 개수의 참조 클래스간에도 이 중 어느 클래스를 참조 클래스로 선정했는가에 따라 거리 추정 블록의 성능이 많은 영향을 받음을 알 수 있다.

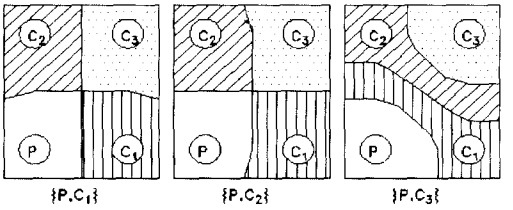


그림 6. 참조 클래스가 두개일 경우의 분류 영역
Fig. 6. Classification region when the number of reference class is 2.

2. 비용 함수의 결정

분류기 설계에서 일반적으로 사용되는 비용함수인 오분류 확률은 식 (4)과 같이 주어진다.

$$P_E = \sum_{i,j} P_j \cdot q_{i,j} = \sum_{i \neq j} P_j \cdot q_{i,j} \tag{4}$$

단, P_j는 클래스 j의 사전 확률,

$q_{i,j}$는 클래스 j를 클래스 i로 인식할 확률,

분류 문제에서 사전(事前) 확률은 특별한 가정이 없는 한, $P_j = 1/N$으로 가정하는 것이 타당하다. 새로이 정의된 거리 근사를 이용하였을 경우 클래스 c_j로부터 클래스 c_i로의 오인식률 $q_{i,j}$는 이들이 거리 근사에 의해 얻어진 새로운 공간, R^M, 에서도 분산이 σ^2인 정규분포를 가진다고 가정을 하면 식 (5)와 같은 값을 얻을 수 있게된다.

$$q_{i,j} = \int_{\frac{\hat{d}_{i,j}}{2}}^{+\infty} \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-x^2}{2\sigma^2}} dx \tag{5}$$

Chernoff's bound[7]에 의해 우리는 $q_{i,j}$가 식 (6)과 같이 상한을 가짐을 알 수 있다.

$$q_{i,j} \leq \exp\left(\frac{-\hat{d}_{i,j}^2}{8\sigma^2}\right) \tag{6}$$

그러나, 오분류율의 합 $\sum_{i=1}^{N} q_{i,j}$은 항상 1보다 작아야 하는 조건을 만족해야하므로, 우리는 $q_{i,j}$를 식 (7)과 같이 정규화하여 근사한다.

$$q_{i,j} \approx 1/N \cdot e^{-\frac{\hat{d}_{i,j}^2}{8\sigma^2}} \tag{7}$$

또한, 식 (7)에서의 거리 근사는 이를 위해 선정한

참조 클래스의 집합, $S=\{r_1, r_2, \cdots, r_M\}$에 의해 주어지므로 오분류 확률을 식 (8)에서와 같이 S의 함수로 표현한다.

$$P_E(S)=\frac{1}{N^2}\cdot\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N}\exp\left(-\frac{\hat{d}_{i,j}^{\,2}}{8\sigma^2}\right) \quad (8)$$

식 (8)에서 주어진 분류기의 오분류율인 $P_E(U)$를 제거하면 참조 클래스의 집합 S에 원소가 하나씩 첨가될 때마다 추가 오분류율의 추정치 $\Delta P_E(S)$를 식 (9)와 같이 구할 수 있다. 이와 같은 전개는 minNET_1과 minNET_0에서의 입력이 모두 정확히 계산한 거리이기 때문에 가능하다.

$$\Delta P_E(S)=\frac{1}{N^2}\cdot\sum_{i=M+1}^{N}\sum_{j=M+1}^{N}\exp\left(-\frac{\hat{d}_{i,j}^{\,2}}{8\eta\cdot\sigma^2}\right) \quad (9)$$

앞의 문제 정의에서 언급한 바대로 이 추가 오분류 확률 근사식이 실제의 오분류 확률을 보다 잘 표현할 수 있도록, 실험적으로 결정하는 파라미터 $\eta\geq 1$를 도입하였다. 본 논문에서의 실험은 대부분 이 파라미터 η를 1.0 근처로 하였을 때에 설계시 사용되는 추가 오분류 확률 추정치와 실제 실험결과에서의 추가된 오분류 확률이 비교적 잘 일치하는 것을 확인하였다.

3. 참조 클래스의 선정 알고리듬

본 논문에서 제시하는 참조 클래스 집합 선정 알고리듬은 greedy algorithm에 기초한다. 한 참조 클래스를 추가할 때마다 $\Delta P_E(S\cup\{c_i\})$를 최소화하는 참조 클래스 c_i를 하나씩 S에 추가한다. 이렇게 참조 클래스 집합 S를 하나씩 추가하다가 문제에서 주어진 조건, $\Delta P_E<\varepsilon$, 을 만족하면 참조 클래스 선정을 중단한다. 이 수렴 조건은 최소한 M = N-1인 경우에 ΔP_E가 0으로 붕괴되므로, 적어도 $M=1,2,\cdots,N-1$의 범위에서 이 알고리듬은 항상 종료 조건을 만족한다. 이 과정을 정리하면 다음과 같다.

a. 초기화.
- 이미 참조 클래스로 선정된 클래스의 집합 S를 공집합으로 초기화한다.
  비참조 클래스 집합 T를 U로 초기화한다.

b. 참조 클래스의 선정.
  T의 원소 중에서 $\Delta P_E(S\cup\{c_i\})$가 가장 적은 클래스 c_i를 새로운 참조 클래스 r_k로 선정한다.
- $S\leftarrow S\cup\{c_i\}$, $T\leftarrow T-\{c_i\}$

c. 반복 및 종료.
- 갱신된 S로 절차 b를 반복한다. $\Delta P_E(S)<\varepsilon$을 만족하면 절차 d를 수행한다.

d. 잡음에 강한 참조 클래스 집합의 선정.
- c에 의해 M이 선정되면, η를 일정 비율 증가시켜 절차 a, b, c를 반복한다.
- 이것은 새로 구한 $n(S_{\eta(k+1)})=M+1$이 될 때까지 반복을 한다.
- $S_{\eta(k)}$를 최종적으로 출력 후에 종료한다.

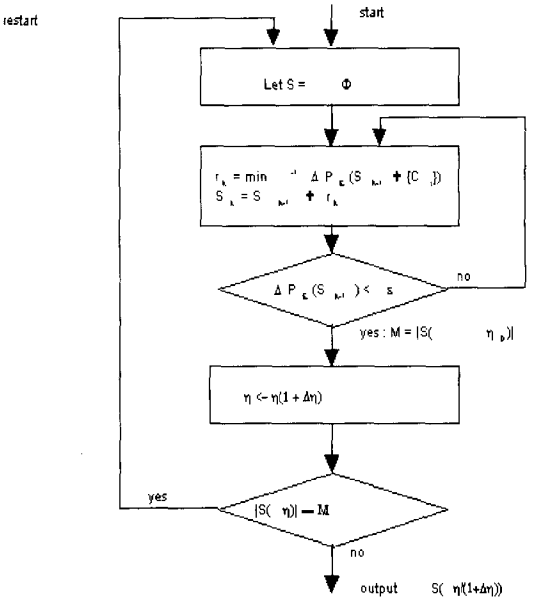


그림 7. 제안된 참조 클래스 선정 알고리듬의 흐름도
Fig. 7. Flow chart of the proposed algorithm for reference class selection.

특히, 절차 d는 같은 갯수의 참조 클래스 집합에서 가능한 한 잡음에 강하도록 하기 위해 주어진 분산보다 더 큰 분산일 경우를 고려하는 절차이다. 이와 같은 알고리듬을 적용하여 M개의 클래스를 참조 클래스로 선정한다면 결국 이 M개의 클래스를 선정하는 데에 걸리는 시간은 $O(N^M)$을 따르게 된다. 따라서 주어진 문제의 고유 공간 차원이 작으면 그만큼 설계하는 데에도 적은 시간이 걸리게 된다.

Ⅳ. 제안된 분류기의 응용 예

1. 예제 1: 숫자인식에의 응용

8*8크기의 숫자를 인식하는 분류기를 RCC 분류기

로 재설계하였다. 각 클래스마다 하나의 대표 벡터만을 가지는 경우를 가정하였다. 입력 데이터는 이 대표 벡터를 인위적으로 변형시켜서 사용하였다. 데이터를 채취할 때에 생길 수 있는 변형으로는 1비트의 잡음과 글자 위치의 부정확함을 가정하였다. 크기나 회전에 의한 변형은 고려하지 않았다. 이와 같은 경우에 클래스와 입력간의 거리는 shifted hamming distance(쉬프트를 허용하는 해밍거리)를 이용하면 글자 위치의 부정확함으로 인한 오분류의 문제는 없앨 수 있다. 쉬프트를 허용하는 해밍거리의 정의는 (10)과 같다.

$$D_{HS}(x, y) = \min D_H(x(t), y(t+\Delta t)) \qquad (10)$$

여기에서, $y(t+\Delta t)$는 $y(t)$를 벡터 Δt만큼 쉬프트한 것을 나타낸다.

이 쉬프트를 허용하는 해밍거리는 허용하는 쉬프트의 정도에 비례해 연산량이 늘어난다. 본 논문에서는 글자의 1/4정도가 상, 하, 좌, 우로 움직일 수 있다고 가정을 하였으므로 하나의 쉬프트를 허용하는 해밍거리는 원래의 해밍거리 계산 식에 비해 가로로 5화소, 세로로 5화소를 움직여 거리를 계산해야 하므로 25배 정도 많은 연산량을 요구한다.

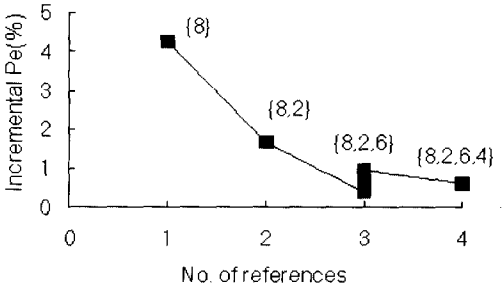


그림 8. 참조 클래스 선정 과정시의 ΔP_E의 변화
Fig. 8. Change of ΔP_E during the reference class selection procedure.

클래스간 거리의 분포를 알려주는 표준편차는 0.82정도로 입력 공간에서 1비트의 잡음이 거의 그대로 반영되었다. $\varepsilon = 1\%$, $\eta_0 = 1.0$으로 하여 참조 클래스 선정 알고리듬을 수행하여 얻은 참조 클래스 집합 S로 {2,3,8}을 얻을 수 있었다. 참조 클래스의 개수가 3이라는 것을 결정한 후에는 참조 클래스의 개수가 3개에서 4개로 넘어가기까지 η를 가능한 증가시켜 예상했던 경우보다도 더 큰 잡음이 들어왔을 경우에도 비교적 좋은 성능을 보이도록 하는 참조 클래스를 선정하였다. 최종적으로 { 2, 6, 8 }을 참조 클래스의 집합으로 선정하였다. 참조 클래스 선정 과정에서 각 숫자를 참조 클래스로 취할 경우에 대한 비용 함수가 변하는 과정과 참조 클래스를 선택함으로써 줄어드는 추가 오분류율을 그림 8에 나타내었다.

최종 minNET$_0$를 exact mode와 approximate mode로 동작시킨 경우의 실험결과 및 분류기의 특성은 다음과 같다. 분류기에서 K_1은 1비트 잡음을 고려하여 설계한 분류기를 뜻하며 K_2는 2비트 잡음을 고려하여 설계한 분류기이다.

표 1. 예제 1의 모의 실험 결과
Table 1. Simulation results of the example.

분류기		1 비트 잡음시(0%)		2 비트 잡음시(0%)	
		ΔP_E(%)	상대 연산량	ΔP_E(%)	상대 연산량
K_1	Exact	1.2%	40%	6.6%	40%
	Approx	1.2%	30%	6.8%	30%
K_2	Exact	0%	50%	0%	50%
	Approx	0%	40%	1.4%	40%

제안한 알고리듬으로 설계한 분류기의 성질을 알아보기 위해 2, 3, 4개의 클래스로 선정할 수 있는 모든 경우(각각 $_{10}C_2$, $_{10}C_3$, $_{10}C_4$)의 오분류 확률에 대한 분포를 그림 9에 나타내었다. 이 그림에서 알 수 있듯이 제안한 참조 클래스 알고리듬으로 비록 최적해를 얻을 수는 없었지만 전체의 해 중에서는 비교적 좋은 성능을 나타냄을 알 수 있다.

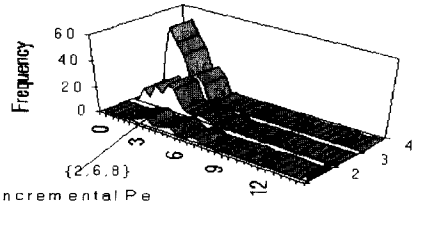


그림 9. 1비트 노이즈일 경우 예제 1-분류기의 성능 분포
Fig. 9. Distribution of the classifier-ex1 performance in 1-bit noise.

2. 예제 2: 영문자 및 숫자 인식에의 응용

두번째의 예제로는 K=64, N=62인 경우로 $\alpha_1 \cdot K$가

앞의 예제에 비해 N보다 상대적으로 작아 전체적인 탐색량이 많은 경우에 대해 실험하였다. 앞의 응용 예를 보다 확장하여 인식의 대상으로 삼은 문자는 앞에서 예제로 썼던 10개의 숫자에 26개의 영문자 대문자와 26개의 영문자 소문자를 추가하여 총 62개의 글자를 인식의 대상으로 삼았다. 그러나, 거리 함수로는 hamming distance를 써서 글자를 채취할 때 위치의 부정확함에 대한 고려를 하지는 않고 다만 잡음만을 글자 변형의 원인으로 생각하였다. 허용 추가 오분류율은 1%이고 $\eta_0=1.0$으로 실험하였다. 마찬가지로 1비트 잡음을 가정하여 본문에서 설명한 알고리듬으로 거리 추정 블록에서 쓰일 참조 글자를 선정한 결과 62개의 문자 중에서 {'9','E','I','Q','e','h'} 이렇게 6개의 글자를 선정하였다. 2비트 잡음일 경우도 가정하여 코드 생성을 하였다. 이 경우에는 8개의 참조 글자로 {'9','E','I','M','P','Q','e','t','x'}를 선정하였다. 각각의 설계된 분류기에 대해 1비트 잡음과 2비트 잡음이 있는 입력 글자들을 가지고 성능을 측정한 결과를 마찬가지로 표 2에 실었다. 여기에서도 K_1과 K_2는 각각 1비트, 2비트 잡음을 고려하여 설계한 분류기이다. 이 결과에서도 볼 수 있듯이 분류기의 연산량은 극복해야 하는 잡음의 크기와 연동될 수밖에 없으며, 본 논문에서 제시한 고속 탐색 방법은 참조 클래스의 개수를 바꾸어 이에 효과적으로 대처할 수 있음을 확인하였다.

표 2. 예제 2의 모의 실험 결과
Table 2. Simulation results of the example 2.

분류기		1 비트 잡음시(0%)		2 비트 잡음시(0.4%)	
		ΔP_E(%)	상대 연산량	ΔP_E(%)	상대 연산량
K_1	Exact	0.6%	11%	2.1%	11%
	Approx	0.8%	10%	3.0%	10%
K_2	Exact	0.3%	16%	1.3%	16%
	Approx	0.5%	14%	1.8%	14%

3. NIST special database HW-1의 숫자 인식에의 응용

4.1절과 4.2절에서의 예들은 저장해야하는 샘플의 수가 각 글자당 하나씩만 존재하는 비교적 작은 크기의 샘플사이즈를 가지는 가상의 데이터로 실험을 한 경우이다. 본 절에서는 NIST special database HWDB-1[8]에서 **box3**에서 **box5**안에 들어 있는 숫자에 대한 분류기에 대해 본 논문의 방법을 적용하였다. 그림 10은 이들 중의 한 box에 있는 10개의 글자를 connected component analysis를 이용하여 영역 분할한 후에 24x24의 크기로 정규화한 영상이다. 이와 같은 처리를 통하여 100명(1,000글자)분에 대한 학습군(training vector) 한 벌과 실험군(test vector) 한 벌을 얻는다. 분류기의 입력으로는 24x24크기의 raw data를 그대로 사용하였다. 최근 이웃 탐색 분류기는 1,000개의 학습군의 표본을 모두 저장한 후에 이중에서 가장 가까운 표본이 속하는 숫자를 출력한다[11].

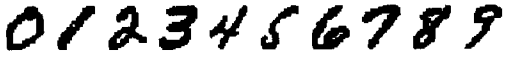


그림 10. NIST special database HW-1의 f_0014의 box4에서 영역분할 후 정규화한 숫자의 예
Fig. 10. Example of the numerals normalized and segmented from box4 of f_0014 of NIST special database HW-1.

이와 같은 방법으로 분류기를 구현하였을 경우에 실험군에 대하여 3.6%의 오분류율을 얻을 수 있었다. NIST database에서 추출한 글자들은 변형 및 잡음이 심하기 때문에 이 전체를 그림 11에서와 같이 100개의 그림 11에서처럼 서브 네트워크로 분할한 후에 각 서브 네크워크를 예제 1에서와 같이 모두 참조 클래스의 집합으로 {2, 6 ,8}을 가지도록 하였고 exact mode만을 사용하여 최종 출력을 하도록 구성하였다. 이와 같이 설계하였을 경우에 연산량은 60%를 감소할 수 있었고, 메모리는 각 비참조 숫자마다 3*1word(=12 byte)씩 늘어 매 글자마다 24x24를 72byte에 저장한 것에 비해 평균적으로 글자당 8.4byte가 증가하여 12%정도의 추가 메모리를 요구하였다.

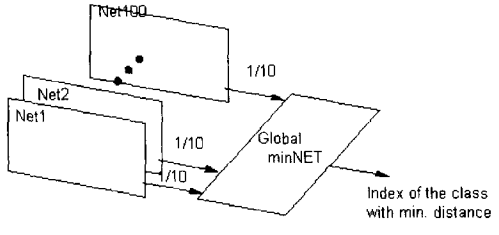


그림 11. NIST 필기체 숫자 분류기의 계층 구조
Fig. 11. Hierarchical architecture of the NIST hand-written numeral classifier.

위의 방법으로 재설계한 고속 탐색 분류기는 실험군에 대하여 4.1%의 오분류율을 보여 원래의 분류기에 비해서 0.5%의 오분류율의 증가를 보였다. 이는 예제 1에서 1비트의 오율이 있을 경우에 1%미만의 추가 오분류율을 가지도록 하는 참조 클래스를 각 서브 네트워크에 사용하였기에 가능한 결과라 볼 수 있다.

표 3. NIST 숫자에 대한 분류기의 성능
Table 3. Performance of the classifier for the numerals in NIST special database HW-1.

분류기의 구분	오분류율(%)	상대적인 탐색시간 (%)	사용 메모리(%)
Given classifier	3.6	100	100
Proposed method with exact mode	4.1	42	112

V. 결론 및 향후 연구 과제

본 논문에서는 일정 수준의 오분류율을 허용하는 분류기를 설계할 때에 기존의 최근 이웃 탐색 분류기에서 잉여 계산량을 줄일 수 있는 구조를 제안하였다. 또한 실제 숫자 분류기와 문자 인식의 예제를 통해 이와 같은 분류기를 설계하는 방법을 제시하였다. 아울러 제안한 분류기의 설계방법은 비록 최적해를 구하지는 못하지만 전체 해 공간에서 상당히 좋은 품질의 해를 구함을 확인하였다. NIST 데이터 베이스를 이용한 실험을 통해서 제안한 고속 탐색 방법 및 설계 방법이 여러 경우에 있어서 기존의 최근 이웃 탐색 방법보다 훨씬 효율적인 대안이 될 수 있음을 확인하였다.

제안한 고속 탐색 방법의 유효한 사용 영역을 알기 위해서는, 주어진 분류 문제에 대해 추가 오분류율을 줄일 수 있는 한계에 대한 정량적인 연구가 더 필요하다. 또한, 제안한 구조를 기존의 다른 계층적인 분류기와 비교하는 연구도 향후 연구에서 다루어져야 할 것이다.

감사의 글

본 논문은 정보 통신 연구 관리단(ITA)의 1996년도 대학기초 연구지원 사업(과제번호 u96-34)의 지원에 의해 이루어진 연구 결과입니다.

참 고 문 헌

[1] Duda and Hart, "Pattern Classification and scene analysis," Wiely, New York, 1973.

[2] W. Li and E. Salari, "A Fast Vector Quantization Encoding Method for Image compression," IEEE trans. CAS for video tech., vol. 5, no. 2, pp. 119-123, April 1995.

[3] G. Poggi, "Fast algorithm for full-search VQ encoding," Electron. Lett., vol. 29, pp 1141-1142, June 1993.

[4] C.-M. Huang, Q. Bi, G.S. Stiles, and R.W. Harris, "Fast full search equivalent encoding algorithm for image compression using vector quantization," IEEE Trans. Image Process., vol. 1, no. 3, pp. 413-416, July 1992.

[5] S.-W. Ra, and J.-K. Kim, "A Fast mean-distance-ordered partial codebook search algorithm for image vector quantization," IEEE Trans. CAS, vol. 40, no. 9, pp. 576-579, Sept. 1993.

[6] Peter J. Verveer and Robert P.W. Duin, "An Evaluation of Intrinsic Dimensionality Estimators," IEEE Trans. PAMI, vol. 17, no. 1, pp. 81-86, Jan. 1995.

[7] A. Papouplis, Probability, Random variables, and Stochastic processes, McGraw-Hill, 1991.

[8] The NIST database can be obtained by writing to : Standard Reference Data, National Institute of Standards and Technology, 221/A323 Gaithersburg, MD 20899 USA and asking for NIST special database 1(HWDB).

저 자 소 개

李 日 完(正會員) 第 33卷 B編 第 5號 參照
서울대학교 전기공학부 박사과정 재학중

蔡 洙 翊(正會員) 第 33卷 B編 第 5號 參照
서울대학교 전기공학부 부교수