신경망을 이용한 영문 대문자 활자 인식 분류방법

전장환° 이강일 이창환 이강우
동국대학교 정보통신공학과
{b4crazy°, leeki816, chlee, klee,}@dongguk.ac.kr

English Capital Letter Classification Method using Neural Network

Jang-Hwan Jun°, Kang-Il Lee, Chang-Hwan Lee, Kang-Woo Lee
Dept. of Information and Communication Engineering, Dongguk Univ.

요 약

본 논문에서는 알파벳 대문자 영상에 다양한 특징을 추출하고 이를 신경망을 이용하여 영문 알파벳을 구분하는 방법을 제안하고자 한다. 제안된 방식은 비교적 간단한 연산을 통해 입력된 영상에 대한 정보를 추출하여 신경망에 대입을 함으로서, 빠른 결과를 얻는 것과 동시에 알파벳 이미지가 아닌 알파벳 내에 들어있는 패턴들을 이용하여 알파벳을 구분함으로서 노이즈에 강한 장점을 나타내고 있다. 다양한 필체를 이용하여 실험을 수행하였고, 현재 사용 중인 상용 프로그램과 본 논문에서 제안한 방법의 적중률을 비교하였다.

1. 서 론

패턴인식은 기계학습 분야에서 어떠한 사물에 특징을 찾아내어서 자동으로 분류를 할 수 있게 해주는 것을 의미한다. 그 중에서 문자 인식은 광학 신호를 입력으로 받아들여 그 문자의 이름을 판독하는 것을 의미한다. 이러한 문자 인식을 수행하기 위해서는 주어진 문자를 판별할 수 있는 유용한 패턴을 제공해야 한다. 하지만 패턴을 만드는 것은 주어진 문제에 따라 그 답이 다르다. 따라서 어떠한 공통적인 해답을 제시할 수 있는 패턴을 제공하기는 매우 어렵다. 또한, 만들어진 패턴의 수가 너무 많은 경우, 향상되는 분류 정확도에 비해 연산 시간이 오래 걸리는 단점이 존재한다.

본 논문에서 가장 중시하는 것은 적절한 적중률을 지닌 채로 패턴을 추출하는데 연산이 적게 소모되는 패턴들을 이용하여서 영어 활자체 대문자를 인식하는 것이다. 또한 이렇게 추출된 패턴들의 결과 값을 신경망을 이용하여 학습하였다. 활자체는 글꼴 모양에 따라 같은 데이터라고 하더라도 많은 노이즈가 포함되어 있기 때문에, 제안한 패턴을 이용하더라도 발생하는 노이즈를 가장 잘 극복할 수 있는 신경망을 사용하였다. 본 논문에서는 신경망의 일반적인 형태인 Back-Propagation 알고리즘을 이용하여 학습하였다[1]. 일반적으로 문자인식에는 신경망이나 퍼지, 유전자 알고리즘등이 많이 사용되며 이들은 노이즈에 대체로 강하다는 특징을 지니고 있다. 본 논문에서 제시한 방법과 현재 사용되는 상용 활자 인식 프로그램과의 적중률을 비교해보았다.

2. 관련연구

문자 인식 기술은 스캐너나 사진등을 이용하여 입력된 입력한 문자 영상으로부터 각종 문자정보를 인식하는 수단을 제공함으로서, 기존에 수작업에 의존하던 작업방식을 자동화 시킬 수 있다. 한글, 한자, 영어 등의 다양한 문자에 대한 인식에 관한 많은 연구가 있었으며, 신문이나 보고서등의 복잡한 양식의 문서를 인식하기 위한 도구도 계발되고 있다. 지금까지 나온 대표적인 방식으로는 저장된 문자 영상과 정합 정도를 인식하는 원형정합법, 문자 영상의 특징 벡터의 유클리디안 기법으로 표현하는 통계적방법, 그리고 문자를 구성하는 요소들의 구조적 연관성을 비교하여 인식하는 구조적방법등이 있다[2].

원형정합법의 경우에는 비교적 구현이 쉽지만 모든 활자체에 대한 정보가 필요하며, 이는 굉장한 메모리 크기를 요구하게 된다. 구조적 방법의 경우에는 각각의 요소를 정확하게 추려낼 수 있는 반면, 그러한 요소를 추려내는데 엄청난 계산양이 요구될 수밖에 없다. 따라서 본 논문에서는 통계적인 방법을 이용하여, 아주 간단한 계산만으로 뽑아낼 수 있는 패턴을 제시하였다. 이러한 패턴에서 발생한 수치를 하나의 문자를 대표할 수 있는 데이터로서 그 값의 범위 또한 매우 작아서 학습시간이 매우 적게 걸리며, 또한 인식률 역시 상용프로그램과 비교해서 손색이 없는 것이 장점이다.

3. 문자 특징 추출

본 논문에서는 다음과 같은 패턴들을 각각의 알파벳 문자에 적용하여 수치화 한 결과를 신경망내의 입력으로 사용하였다. 다음 표에서 본 논문에서 사용한 패턴의 정의를 간단히 나타내었다.

[표 1] 신경망 학습을 위해 사용된 패턴

속성 이름	속성 정의
Ob_No	닫힌 영역의 개수
Hor_No	수평선의 개수
Ver_No	수직선의 개수
Cross_Hor_No	중심을 지나는 수평선과 알파벳과의 교차하는 객체의 수
Cross_Ver_No	중심을 지나는 수직선과 알파벳과의 교차하는 객체의 수
Rate_Of_R_N_L	알파벳 좌/우 비율
Rate_Of_U_N_D	알파벳 상/하 비율
Resampling_No	resampling 기법

3.1 닫힌 영역의 개수

[그림 1] 닫힌 영역의 정의

닫힌 영역은 연결된 선으로 인해서 같은 배경이 2개로 나누어 진 경우를 이야기 하며, [그림 1] 에서 알파벳 'A'에 대한 입력 영상에 대해서, 배경의 닫힌 공간을 보면 1과 2 두개로 나누어진다. 그러므로 값은 2 가 나오게 된다.

3.2 수평선의 개수

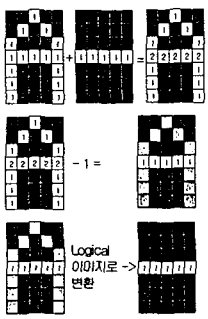
수평선의 개수를 세기 위하여, 본 논문에서는 알파벳 영상 내에서

수직선과 그 외의 pixel을 지우고 다시 원래의 알파벳과의 차집합을 취해, 수평선만을 남긴 후 객체의 수를 센다.

3.3 수직선의 개수

수평선의 개수를 구하는 것과 동일한 방식으로 진행하며 빠른 연산을 위해 수평선의 개수 구하는 작업을 하면서 구해진 수평선과 원래 객체와의 차집합을 한 후 수직선 외의 픽셀을 지워서 나온 객체의 수를 센다.

3.4 중심을 지나는 수평선과 알파벳과의 교차하는 객체의 수



[그림 2] Cross_Ver_No 구하는 법

이미지의 가운데를 통과하는 가상의 수평선을 만들고 그것과 이미지와의 교차점의 개수를 구한다. 우선 알파벳에 대한 logical 이미지를 구하는데, logical 이미지는 영상을 1, 그 이외의 0으로 정의하는 이미지를 나타낸다. logical 이미지로 변환된 알파벳에다가 중심점을 통과하는 수평선을 표현한 logical 이미지를 더하게 되면 값은 0,1,2가 존재하게 되고, 여기서 다시 전체 이미지에 대해서 -1을 더함으로서 값은 -1,0,1이 남게 된다. 이것을 다시 logical 이미지로 변환하게 되면 -1은 0으로 변하게 됨으로서 다시 0과 1인 값만 남게 된다. 1값을 가지고 있는 객체의 수를 셈으로서 교차점과 객체의 수를 알 수 있다. [그림 2]는 위의 과정을 간단히 그림으로 표현하였다.

3.5 중심을 지나는 수직선과 알파벳과의 교차하는 객체의 수

중심을 지나는 수평선과의 교차점 개수를 구하는 방법과 마찬가지로, logical 이미지를 이용한 연산을 이용하여 중심을 지나는 수직선과 알파벳 이미지와의 교차하는 객체의 수를 연산하였다.

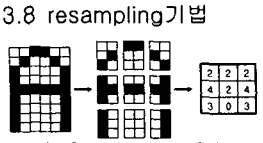
3.6 알파벳 좌/우 비율

알파벳 이미지를 반으로 자른 후 좌/우의 픽셀수를 세서 왼쪽이 많으면 -1 같으면 0 오른쪽이 많으면 1로 하여 연산하였다. 그러나 폰트 자체의 노이즈를 고려하여 좌/우 픽셀의 수가 5%이내의 차이가 나는 경우는 두 개의 값이 같은 것으로 고려하여 0값을 취하도록 하였다.

3.7 알파벳 상/하 비율

알파벳 이미지 좌/우 비율을 구하는 방법과 마찬가지로 상/하의 비율을 구한 후, 위의 비율이 크면 1 같으면 0 아래의 비율이 크면 -1로 하여 연산하였다. 이 경우도 마찬가지로 상/하의 픽셀의 수 차이가 5% 이내가 차이나는 경우는 픽셀의 수가 같은 경우로 고려하였다.

3.8 resampling기법

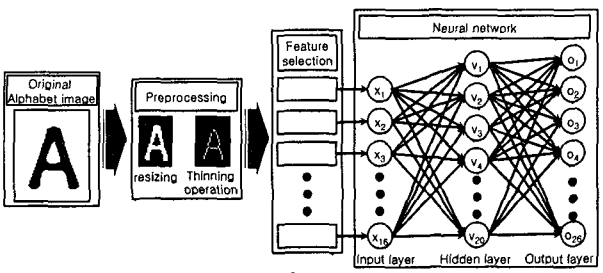

[그림 3] resampling 기법

하나의 알파벳 이미지를 n개의 독립된 객체로 나눈 후, 각각의 객체에서 알파벳을 구성하는 점의 개수를 세서 resampling속성의 값에 넣는다 [그림 3]. 이 기법은 학습 데이터의 작은 변화가 결과에 큰 영향을 미치지 않을 때 사용되는 방법이며[3], 본 논문에서는 알파벳을 9개의 독립된 객체로 나누어서 사용하였다.

4. 실 험

4.1 실험 방법

본 논문에서 위와 같이 정의된 패턴에 대해서 신경망을 이용하여 실험하였다. 여느 패턴 인식의 실험과 마찬가지로 본 실험은 크게 전처리과정, 패턴 추출, 학습의 3부분으로 나누어진다. 본 논문에서 수행한 실험에 대한 개요는 [그림 4]를 통해 나타내었다. 이 모든 작업은 MathWork사의 MATLAB를 사용하여 진행하였다.



[그림 4] 실험과정

실험을 통해 적중률을 구하는 방법은 k-fold cross-validation 방식을 사용하였다[4]. 이 방식은 전체 데이터를 k개의 동일한 크기의 부분 집합으로 나누어서 그 중의 한 개를 실험 데이터로 사용하고 나머지는 학습 데이터로 사용하여 검증하는 방식이다. 따라서 전체 글꼴 중에서 1 개의 글꼴은 실험 데이터로 설정하고 나머지 글꼴은 신경망의 학습 데이터로 사용하여 학습을 시킨 후, 학습된 신경망에 설정된 1개의 실험 데이터를 넣어서 적중률을 계산하였다. 본 논문에서는 전체 18개의 글꼴을 이용해 총 18번의 실험을 수행하여 평균 적중률을 계산하였다. 본 연구에서는 A부터Z까지 26개의 알파벳을 워드 프로세서를 이용해 서로 다른 18가지의 폰트로 만든 후[표 2], 이것을 jpg이미지로 변화하여 특징 추출 및 신경망의 학습에 사용하였다.

[표 2] 실험에 사용된 폰트

Nice Font	
굴림	ABCDEFGHIJKLMNOPQRSTUVWXYZ
돋움	ABCDEFGHIJKLMNOPQRSTUVWXYZ
휴먼 옛체	ABCDEFGHIJKLMNOPQRSTUVWXYZ
헤드라인 M	ABCDEFGHIJKLMNOPQRSTUVWXYZ
그래픽	ABCDEFGHIJKLMNOPQRSTUVWXYZ
복숭아	ABCDEFGHIJKLMNOPQRSTUVWXYZ
헤드라인	ABCDEFGHIJKLMNOPQRSTUVWXYZ
양재튼튼B	ABCDEFGHIJKLMNOPQRSTUVWXYZ
시스템	ABCDEFGHIJKLMNOPQRSTUVWXYZ
Bad Font	
궁서	ABCDEFGHIJKLMNOPQRSTUVWXYZ
휴먼매직	ABCDEFGHIJKLMNOPQRSTUVWXYZ
궁서B	ABCDEFGHIJKLMNOPQRSTUVWXYZ
엽서	ABCDEFGHIJKLMNOPQRSTUVWXYZ
오이	ABCDEFGHIJKLMNOPQRSTUVWXYZ
타이프	ABCDEFGHIJKLMNOPQRSTUVWXYZ
휴먼명조	ABCDEFGHIJKLMNOPQRSTUVWXYZ
명조	ABCDEFGHIJKLMNOPQRSTUVWXYZ
해서 간자	ABCDEFGHIJKLMNOPQRSTUVWXYZ

실험에 사용된 폰트들은 실험의 편의를 위하여 cice font 와 bad font의 집합으로 나누었다. nice font는 글씨체가 비교적 정자이고, 노이즈가 비교적 적은 글꼴들의 집합이고, Bad Font는 글씨체가 기울어져 있거나, 노이즈가 있는 폰트들을 뜻한다. 그리하여, Nice Font만을 가지고 학습시킨 후 나온 결과와 Bad Font를 가지고 학습시킨 후 나

온 결과, 그리고 이 두 가지 집합들을 다 합쳐서 학습한 후 나온 결과 3가지를 상용 프로그램인 아르마와 비교 하였다.

4.2 전처리 과정

본 논문에서는 글꼴에 따라 발생할 수 있는 노이즈를 최소화하기 위해서 다음과 같은 전처리 과정을 수행하였다.

• Image Resizing

알파벳의 패턴을 추출하기 위하여 제일 먼저 수행한 것은 Image Resizing 이다. 이는 각각 글꼴이 크기가 같더라도 자체 모양으로 인해 약간씩 크기에 대한 오차가 발생한다. 따라서 본 논문에서는 전처리 과정으로 각 폰트의 특징 중에서 크기에 관련된 부분을 정규화하기 위해서 Image Resizing을 수행하였다.

• Thinning Methodology

Thinning methodology[5]란 특정한 캐릭터의 패턴을 얇은 선으로 나타내는 것을 의미한다. 이는 글꼴에 무관하게 패턴을 추출하기 위한 것으로, 알파벳을 가지고 패턴을 추출하기 위한 픽셀 작업을 하다 보면 어떤 폰트는 상대적으로 두껍고, 어떤 폰트는 상대적으로 얇음으로 결과 값이 상이하게 된다. 그럼으로 초기단계에서 Thinning operation을 수행하여 최소의 픽셀만을 남긴 후, 특성 추출 작업을 수행하였다 [그림 5].

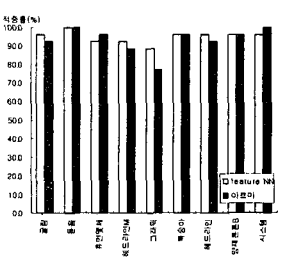
[그림 5] thinning operation 수행전과 수행후의 폰트

4.3 데이터 학습

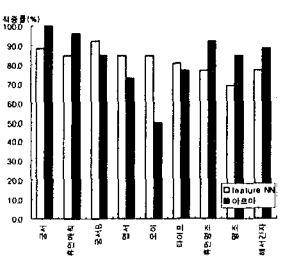
본 논문에서는 서론에 밝힌 바와 같이, 신경망을 이용하여 추출된 패턴에 대한 학습을 수행하였다. 신경망 역시 MATLAB 프로그램에서 제공하는 신경망을 사용하였다. 이 때 입력층에 들어가는 데이터의 수는 총 16개로서 resampling 기법을 통해 나온 9개의 구역의 픽셀의 수에 대해 9개 전부를 입력으로 하여서 16개가 되었다. 출력층의 노드의 수는 26개로서 이는 출력되는 노드의 값이 1인 노드의 번호가 정답으로 나오게 하였다. 은닉층의 노드의 수는 25개로서 이는 실험을 통해서 가장 적중률이 높은 노드의 수이다. 이 때, 학습 회수에 대한 제한 값은 5000회로 하였으며, 학습률은 0.04로 하였다.

4.4 실험 결과

우선 전체 글꼴을 노이즈에 따라 nice font와 bad font로 나누어서 cross-validation을 이용 실험을 하였다.[그림 6,7] 상용 프로그램과 본 논문에서 제안한 방법과의 평균 적중률을 비교한 결과는 크게 차이가 나지 않았다.

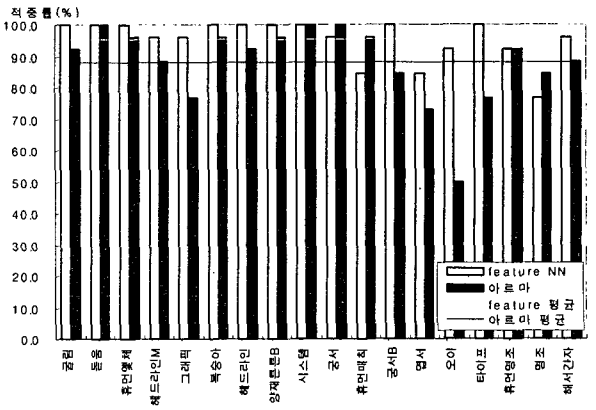


[그림 6] nice font를 이용한 cross-validation 결과



[그림 7] bad font를 이용한 cross-validation결과

전체 글꼴에 대해서 cross-validation을 이용하여 수행한 결과[그림 8]의 경우는 본 논문에서 제안한 방법의 적중률은 95.3%, 상용프로그램의 경우는 87.5%로 본 논문에서 제안한 방법이 약 7.5% 더 높은 정확도를 나타내었다.



[그림 8] 전체 글꼴에 대한 cross-validation 결과

5. 결론

본 논문에서는 알파벳 내에 들어있는 다양한 패턴들을 이용하여 각각을 구분할 수 있는 방법을 제안하였다. 제안된 패턴들은 비교적 연산이 쉽고, 학습을 신경망을 통해 수행하였기 때문에, 새로운 실험 데이터가 들어오면 빠른 시간 내에 결과를 출력할 수 있는 장점이 있다. nice font와 bad font를 따로 학습한 후 close-validation 방식을 이용해서 평균적인 적중률을 계산 하였을 때는, 두가지 글꼴집합들이 대표적인 상용프로그램과 비슷한 적중률을 가졌었다. 그러나 두가지 글꼴을 같이 학습 시켰을 때 nice폰트의 경우 비슷한 결과를 가졌지만 bad font, 즉 노이즈가 심한 글자의 경우 상용 프로그램보다 더 좋은 결과를 나타내었다. 이는 본 논문에서 제안한 방식이 연산 속도뿐만 아니라, 노이즈에 강하다는 것을 의미한다. 앞으로 좀 더 보안연구를 통해서 적중률 강화, 소문자 인식, 필기체 인식 등의 과제를 수행해 나갈 것이다.

7. 참고문헌

[1] Tom Mitcheal, *Machine Learning*, page 81-127, The McGraw-Hill Companies, Inc., 1976

[2] 손영우, *카오스 이론을 이용한 고정도 문자 인식 시스템*, 한국멀티미디어학회논문지, 4권, 6호, 2001

[3] Richard O. Duda, Peter E. Hart, David G. Stork, *Pattern Classification 2Ed*, page 471-485, Wiley Interscience, 2001

[4] M Quenouille, *Approximate tests of correlation in time series*, Journal of the Royal Statistical Society B, 11:18-84, 1949.

[5] Lousia Lam, Seong-Whan Lee, Ching Y. Suen, *Thinning Methodologies - A Comprehensive Survey*, IEEE Transactions on pattern analysis and machine intelligence, Vol 14, No 9, page 869-885, 1992

[6] Hon Keung Kwan, Yaling Cai, *A Fuzzy Neural Network and its Application to Pattern Recognition*, IEEE Transaction on fuzzy systems Vol 2, No3, page 185-193, 1994