Journal of Digital Contents Society Vol. 13 No. 3 Sep. 2012(pp. 309-315)
http://dx.doi.org/10.9728/dcs.2012.13.3.309

삼각 부등식을 이용한 빠른 벡터 양자화 코드북 생성

이현진*

요 약

액티브 데이터는 벡터 양자화 코드북이 생성될 때 소속된 군집이 변경되는 입력 데이터이다. 벡터 양자화 코드북 생성 알고리즘의 수행 과정을 살펴보면, 전체 입력 데이터 중 실제 액티브 데이터는 알고리즘이 반복될 수록 감소된다. 따라서 액티브 데이터를 정확히 추정하여, 추정된 액티브 데이터에 대해서 코드북 생성을 수행하면, 전체 코드북 생성 시간을 크게 단축할 수 있다. 본 논문에서는 삼각 부등식을 이용하여 액티브 데이터를 선택하는 방법을 제안한다. 실험결과 액티브 데이터들을 빠른 시간에 추정할 할 수 있었고, 이를 통해 전체 벡터 양자화 코드북 생성 시간 측면에서 우수한 성능을 보였다.

An Efficient Vector Quantization Codebook generation using a Triangle Inequality

Hyunjin Lee*

Abstract

Active data are the input data which are changed its membership as Vector Quantization codebook generation algorithm is processed. In the process of VQ codebook generation algorithm performed, the actual active data out of the entire input data will be less presented as the process is performed. Therefore, if we can accurately find the active data and only if we are going to do VQ codebook generation on the active data, then we can significantly reduce the overall generation time. In this paper, we presented the triangle inequality based algorithm to select the active data. Experimental results show that our algorithm is superior to other methods in terms of the VQ codebook generation time.

Keywords : Vector Quantization, VQ Codebook, Triangle Inequality

1. 서 론

벡터 양자화(Vector Quantization, VQ)는 데이터를 요약하기 위한 효율적이고 간단한 방법이다[1]. 벡터 양자화는 간단하고 적용하기 쉽기 때문에 패턴 인식, 이미지 압축, web/data mining, 음성 인식, 얼굴 인식 등 다양한 분야에서 사용되고 있다[2-7]. 다양한 VQ 코드북 생성 알고리즘들이 있으며, VQ 코드북 생성 알고리즘이 가지고 있는 특징은 군집에 속한 데이터들의 유사도는 증가시키고, 군집에 속하지 않은 데이터들의 유사도는 감소시키는 것이다.

VQ 코드북을 이용한 데이터 분석을 수행할 때, 데이터 구조나 분류와 같은 데이터와 관련된 정보들은 주어지지 않는다. 그래서, VQ는 데이터 탐색이나 발굴의 과정이라고 한다. 대규모 데이터를 비슷한 특징을 가지는 데이터의 집합으로 묶기 때문에 데이터 집중이나 단순화라고 하기도 한다. 특히, VQ는 데이터에 대한 구조적인 특징에 대한 정보를 생성하기 때문에 가설을 생성하는 과정과 연관되어 있다[8].

Generalized Lloyd Algorithm(GLA)는 가장 많이 사용되는 VQ 코드북 생성 알고리즘의 하나이다[9,10]. 이 알고리즘은 d-차원 공간에 위치한 점들을 입력 데이터로 가진다. 이 알고리즘의 목적은 k개의 군집으로 입력 데이터들을 매핑

※ 제일저자(First Author): 이현진
접수일:2012년 08월 01일, 수정일:2012년 08월 17일
완료일:2012년 08월 20일
* 숭실사이버대학교 컴퓨터정보통신학과
hjlee@mail.kcu.ac

시키는 것이다. GLA는 사용하기 쉽고, 효율적이라는 장점을 가진다. 반면에 군집의 개수를 미리 선정해야 하고, 이상치를 처리하는데 어려움이 있고, 대규모 데이터를 처리하는데 비효율적이라는 단점이 있다.

GLA의 계산 시간을 줄이고자 하는 연구는 많이 수행되고 있다. Lai와 Lue는 중심점을 생성하는 과정의 속도를 향상시킨 빠른 코드북 생성 알고리즘(fast codebook generation algorithm, FCGA)을 제안했다[11]. FCGA는 Huang등이 제안한 빠른 탐색 기법[12]을 수정하고, 코드워드(code word)를 생성시 삼각 부등식을 활용했다. Pelleg와 Moore는 K-means 군집화를 위해 kd 트리에서의 정렬 기법을 사용했다[13]. Kanungo 등은 새로운 코드워드를 빠르게 발견하기 위하여 kd 트리를 이용한 필터링 알고리즘을 개발했다[14]. 이러한 방법들은 코드북 생성 알고리즘의 분할 단계에서의 속도 향상을 위하여 기하학적인 정보를 사용했다. 이 방법들은 한 반복 단계에서의 정보를 다음 단계에서 사용하지 않는다.

김성재 등은 벡터 양자화를 빠르게 수행하기 위하여 삼각 부등식을 이용한 빠른 VQ 코드북 탐색법을 제안했다[15]. 김성재 등은 코드북을 탐색할 때 삼각 부등식을 사용하고, 본 논문에서는 코드북을 생성할 때 삼각 부등식을 사용한다는 점이 차이가 있다. 또한, 김성재 등이 제안하는 방법은 고정된 키팩터를 이용하지만, 제안하는 방법은 기준이 되는 값이 계속 변화한다는 차이가 있다.

본 논문은 각 단계에서의 거리 계산 시간을 줄여서 최적화된 GLA를 제공하는 것이다. 최적화 알고리즘은 GLA에서 대부분의 거리 계산은 여분의 것이라는데 기반을 둔다. 군집의 중심에 가까운 데이터들은 해당 군집에 반드시 소속될 것이기 때문에 정확한 거리를 계산할 필요가 없다. 또한, 군집의 중심에서 먼 데이터들은 해당 군집에 소속되지 않을 것이기 때문에 마찬가지로 정확한 거리를 계산할 필요가 없다.

제안하는 방법은 일반적인 GLA에 기반을 두기 때문에 다음 특징을 만족시켜야 한다[16]. 먼저, 어떠한 초기 중심으로부터 시작해도 상관이 없고, 현존하는 모든 초기화 기법을 사용할 수 있어야 한다. 두 번째, 똑같은 초기 중심에서 시작하면, 최종 결과는 일반적인 GLA와 동일한 결과를 보여야 한다. 마지막으로, 거리를 계산하는 방식은 유클리디언 거리(Euclidean distance)나 코사인 상관계수(Cosine Coefficeint)등 특정 거리에 한정하지 않고, 모든 거리 계산 방식을 지원해야한다.

제안하는 방법은 위 세 가지를 모두 만족하며, 특히 두 번째 특징에서 최종 결과뿐만 아니라 각 단계에서도 일반적인 GLA와 동일한 결과를 보이고 있다.

본 논문의 구성은 다음과 같다. 2장에서는 삼각 부등식을 이용하여 액티브 데이터를 구하는 방법을 설명하고, 3장에서는 새로운 알고리즘에 대해서 설명한다. 4장에서는 실험 결과를 분석하고 5장에서 결론을 맺는다.

2. 삼각 부등식 적용

본 논문에서 GLA의 속도를 향상시키기 위하여 제안하는 방법은 삼각 부등식에 기반을 둔다. x, y, z의 세 점이 있을 때 $d(x,z) \leq d(x,y) + d(y,z)$이다. 이 수식은 모든 거리 계산 방법인 d에 상관없이 성립한다.

Lemma 1: x가 데이터의 한 점이고, b와 c가 코드워드일 때, $d(b,c) \geq 2d(x,b)$ 면, $d(x,c) \geq d(x,b)$ 이다[16].

Proof: $d(b,c) \leq d(b,x) + d(x,c)$ 이다. 그래서, $d(b,c) - d(x,b) \leq d(x,c)$이다. 왼쪽 수식을 살펴보면, $d(b,c) - d(x,b) \geq 2d(x,b) - d(x,b) = d(x,b)$ 이다. 그러므로 $d(x,b) \leq d(x,c)$ 이다.

Lemma 1은 다음과 같이 활용할 수 있다. x는 데이터의 한 점이고, c는 x가 현재 소속된 군집의 코드워드이며, c'는 다른 군집의 코드워드이다. lemma에 의해서 $d(c,c')/2 \geq d(x,c)$ 이면, $d(x,c') \geq d(x,c)$이다. 이 경우에는 $d(x,c')$의 거리를 계산할 필요가 없다.

$d(x,c)$를 정확히 알지 못하고, 상계(upper bound)인 u (즉, $u \geq d(x,c)$) 만 알고 있다고 가정하자. 이러한 경우에 $u > d(c,c')/2$ 인 경우에만 $d(x,c')$와 $d(x,c)$를 계산하면 된다.

모든 $c' \neq c$에 대하여 $u \leq \min d(c,c')/2$ 이라면, 데이터 x는 코드워드 c에 소속된 상태로 유지되고, x와 관련된 모든 거리 계산은 생략되게

된다.

Lemma 2: x가 데이터의 한 점이고, b와 c가 코드워드이면, $d(x,c) \geq \text{abs}(d(x,b) - d(b,c))$이다.

Proof: $d(x,b) \leq d(x,c) + d(b,c)$인 것을 알고 있다. 따라서, $d(x,c) \geq d(x,b) - d(b,c)$이다. 마찬가지로, $d(b,c) \leq d(x,c) + d(x,b)$ 이므로, $d(x,c) \geq d(b,c) - d(x,b)$ 이다. 삼각형은 양수만 존재하기 때문에, $d(x,c) \geq \text{abs}(d(x,b) - d(b,c))$ 이다.

Lemma 2에 의해서 x가 데이터의 한 점이고, b와 c가 코드워드이면, $d(x,c)$는 $\text{abs}(d(x,b) - d(b,c)) \leq d(x,c) \leq d(x,b) + d(b,c)$를 만족하게 된다.

GLA를 수행할 때 코드 벡터를 갱신하는 과정이 지속적으로 반복된다. 이 반복 과정을 단계(stage)라고 하며, 각 단계에서 개별 데이터들과 코드워드들과의 거리를 계산하여 가까운 코드워드를 찾는 작업을 수행한다. 이 단계가 GLA를 수행할 때 가장 오랜 시간이 소요되는 과정이다.

x가 데이터의 한 점이고, c는 x가 현재 소속된 군집의 코드워드이며, c^*는 GLA의 반복 과정의 다음 단계에서 사용될 c의 코드워드이다. 이전단계의 군집과 현재 단계의 군집에 소속된 데이터의 변경이 없다면, $c^* = c$ 가 될 것이다. lemma 2에 의해서 $\text{abs}(d(x,c) - d(c,c^*)) \leq d(x,c^*) \leq d(x,c) + d(c,c^*)$ 이다. 그러므로, $d(x,c^*)$의 상계 $u(x)$는 $d(x,c) + d(c,c^*)$이고, 하계(lower bound) $l(x)$는 $\text{abs}(d(x,c) - d(c,c^*))$이다.

시뮬레이티드 어닐링(simulated annealing)은 금속 결정의 원활한 형성을 위해 가열과 냉각을 적절히 조절해야 하는 기술을 최적화 이론에 적용한 것으로, 분자의 자유로운 이동이 가능한 상태를 만드는 것이다. $d(x,c^*)$을 계산할 때 시뮬레이티드 어닐링을 사용하여 계산량을 줄일 수 있다.

$d(x,c^*)$가 상계인 경우에는 코드워드와의 거리가 멀어지기 때문에 데이터 x가 군집 c^*에 속하지 않을 가능성이 올라간다. 반면에 $d(x,c^*)$가 하계인 경우에는 코드워드와의 거리가 가까워지기 때문에 데이터 x가 군집 c^*에 속할 가능성이 올라간다. 시뮬레이티드 어닐링을 이용하여 데이터 x가 현재 소속된 군집에서 벗어나 다른 군집으로 소속을 변경하게 한다. 이를 위해 데이터 x가 소속된 군집의 경우에는 $d(x,c^*) = u(x) = d(x,c) + d(c,c^*)$로 상계로 설정하고, 데이터 x가 현재 소속하지 않은 군집의 경우에는 $d(x,c^*) = l(x) = d(x,c) - d(c,c^*)$로 하계로 설정한다.

이러한 방법으로 $d(x,c^*)$를 계산하면, $d(x,c)$는 이전 단계에서 계산된 결과이므로 계산이 필요하지 않으며, $d(c,c^*)$는 새로 계산해야하지만, 코드 벡터의 크기가 작기 때문에 $d(c,c^*)$의 계산 시간은 작기 때문에, 전체 계산 시간을 줄일 수 있다.

3. 빠른 벡터 양자화 코드북 생성 알고리즘

2절에서 살펴본 것들을 사용한 새로운 빠른 벡터 양자화 코드북 생성 알고리즘은 다음과 같다.

먼저, 임의로 초기 코드북 CB_0를 생성한다. i = 1 로 초기화 한다. 모든 입력 데이터 x를 가장 가까운 코드워드 $c(x) = \text{argmin}_c d(x,c)$ 로 할당한다. Lemma 1을 사용하여 불필요한 계산을 줄인다.

다음으로 아래 과정을 수렴할 때 까지 반복한다.

1) 코드워드에 속한 데이터들의 평균을 구하여 새로운 코드북 CB_i를 계산한다.
2) 코드북 CB_i의 모든 코드워드들인 c, c'에 대하여 둘 사이의 거리 $d(c,c')$을 계산한다. 모든 코드워드 c에 대하여 $s(c)$를 계산한다.

$$s(c) = \frac{1}{2} min_{c' \neq c} d(c,c')$$

3) 코드북 CB_i의 코드워드 c와 이전 단계의 코드북 CB_{i-1}의 코드워드 c^* 사이의 거리 $d(c,c^*)$ 를 계산한다.
4) 모든 코드워드 c에 대하여, 입력데이터와의 거리 $d(x,c)$를 계산한다. Lemma 2에 의하여 x가 이전 단계의 코드워드 c^*에 속해 있었다면,

$$d(x,c) = d(x,c^*) + d(c,c^*)$$

이고, 이전 단계의 코드워드 c^*에 속해 있지 않다면,

$$d(x,c) = \mathrm{abs}(d(x,c^*) - d(c,c^*))$$

이다.

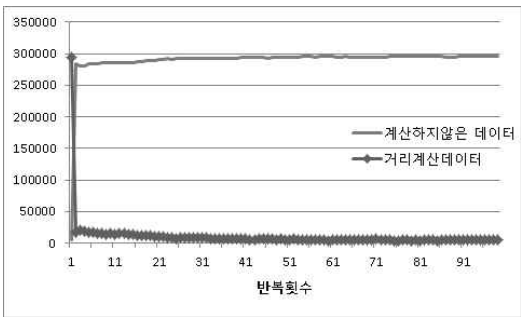
5) 모든 입력 데이터 x에 대하여 가장 가까운 코드워드를 할당한다. 새로 할당된 코드워드 $c(x)$가 이전 단계의 코드워드 $c^*(x)$와 같은 모든 입력 데이터를 찾아내어 새로운 코드워드와 거리 계산을 줄인다.
6) 남아 있는 입력 데이터 x에 대하여 $d(x,c) \leq s(c(x))$인 입력 데이터 x를 찾아낸다.
7) 모든 남아 있는 입력 데이터 x와 코드워드 c에 대하여, 즉,

$$c(x) \neq c^*(x)$$
$$d(x,c) \geq d(c(x), c)/2$$

일 때, 입력 데이터 x와 코드워드 사이의 거리 $d(x,c)$를 계산하고, 가장 가까운 코드워드 $c(x)$에 할당한다.
8) 코드북 CB_i를 평가하고, 평가 결과가 좋으면, 멈추고, 그렇지 않으면, $i = i + 1$을 수행한다.

3 번의 코드워드 사이의 거리 계산은 코드워드의 수가 작아 계산량을 줄일 수 있다. 4 번의 입력 데이터와 코드워드 사이의 거리 계산은 이미 주어진 값(이전 단계의 입력 데이터와 코드워드 사이의 거리와 3 번에서 계산한 거리)을 이용한 덧셈, 뺄셈 계산하여 계산량을 줄 일 수 있다. 계산량이 가장 많은 부분은 7 번째 거리 계산이 부분이다.

제안하는 알고리즘이 거리 계산 횟수가 감소하는 가장 큰 이유는 5 번, 6 번을 통해 거리 계산이 이루어져야 하는 데이터들만 빠르게 발견하기 때문이다. 몇 단계가 수행되고 나면, 코드워드 값의 변화량은 매우 작아진다. 즉, d(c,c*)의 값이 작기 때문에, 5 번에서 거의 대부분의 데이터에서 추가적인 거리 계산을 수행할 필요가 없게 된다. 이를 통해 걸러지지 않는 데이터에 대해서는 6 번을 통해 다시 한번 거르고 있기 때문에, 거리 계산을 수행하지 않을 데이터들을 쉽게 찾아낼 수 있다.



(그림 1) 제안하는 알고리즘에 의해서 찾아진 거리를 계산하지 않는 데이터의 비율

(그림 1)은 각 단계에서 300,000의 거리 계산이 필요한 데이터에 대해 제안하는 알고리즘을 적용한 결과를 보여주고 있다. 1 단계에서는 코드워드의 안정화가 이루어지지 않았기 때문에 거리를 계산해야 할 데이터가 많지만, 2 단계 이후로는 거리를 계산해야 할 데이터가 많이 줄어진 것을 확인할 수 있다. 1 단계에서는 98%의 데이터에 대해 거리 계산을 수행해야 하지만, 2 단계 이후에서는 평균적으로 3%의 데이터에 대해서만 거리 계산을 수행하면 되었다. 따라서, 제안하는 알고리즘이 거리 계산 횟수를 줄이는데 유용한 방법인 것을 확인할 수 있다.

<표 1> 실험에 사용한 데이터 집합

이름	데이터 수	차원 수	설명
birch	100,000	2	10 x 10 의 가우시안 군집, Zhang등의 DS1 [17]
covtype	150,000	54	remote soil cover measurement [18]
kddcup	95,413	56	KDD cup 1998 데이터 [19]
mnist50	60,000	50	NIST 필기체 훈련 차원 축소 데이터 [20]
mnist784	60,000	784	NIST 필기체 훈련 원형 데이터 [20]

4. 실험환경 및 결과

본 연구는 2.8GHz CPU 시스템에서 C# 언어로 구현하였다. 사용된 컴파일러 버전은 .Net Framework 4 이다.

실험은 5개의 데이터 집합에 적용하여 실험하였다. 5개 중 4개는 큰 차원을 가지고 있다. 데이터 집합에 대한 설명은 <표 1>에 있고, 실험 결과는 <표 2>에 있다. <표 2>에서 “standard”와 “Elkan”, “suggest”에 의한 값은 거리 계산 수행 횟수이고, “speedup”은 “suggest”가 “standard”에 비해 어느 정도 속도 향상을 보였는지를 나타낸다. “Elkan”은 Elkan에 의해 제안된 방법이고, “suggest”는 본 논문에서 제안하는 방법에 의한 결과이다.

실험 결과의 항상성을 위하여 입력 데이터 집합에 대하여 고정된 초기화 방법을 사용하였다. 거리를 최적화하는 방법은 초기값을 구하는 시간은 오래 걸리지만, 항상 같은 결과를 보이고, 우수한 성능을 보인다. 본 논문에서는 거리를 최적화하는 방법 중 SCS (Simple Cluster Seeking)을 사용하였다[21].

이전 논문들의 실험 결과와 비교하기 위하여 제안하는 알고리즘의 거리 계산 횟수를 측정하여 성능 비교를 하였다. GLA의 성능을 향상시키는 모든 방법은 보조 데이터 구조를 생성하고 갱신하는 오버 헤드(overhead)를 가지고 있다. 즉, 거리 계산 횟수와 비교하여 수행 시간의 감소는 더 적어질 수 있다. 제안하는 방법에 의한 거리 계산 횟수의 감소는 크지만, 수행 시간의 감소는

<표 2> k의 값을 변화시키면서 실험한 결과

		k = 3	k = 20	k = 100
birch	iterations	17	38	56
	standard	5.100e+06	7.600e+07	5.600e+08
	Elkan	4.495e+05	1.085e+06	1.587e+06
	Suggest	4.679e+05	1.188e+06	1.595e+06
	speedup	10.9	64	351
covtype	iterations	18	256	152
	standard	8.100e+06	7.680e+08	2.280e+09
	Elkan	9.416e+05	7.147e+06	7.353e+06
	Suggest	9.643e+05	6.982e+06	7.331e+06
	speedup	8.4	110	311
kddcup	iterations	34	100	325
	standard	9.732e+06	1.908e+08	3.101e+09
	Elkan	6.179e+05	3.812e+06	1.005e+07
	Suggest	6.666e+05	3.648e+06	9.971e+06
	speedup	14.6	52.3	311
mnist50	iterations	38	178	217
	standard	6.840e+06	2.136e+08	1.302e+09
	Elkan	1.573e+06	9.353e+06	3.159e+07
	Suggest	1.555e+06	8.863e+06	2.881e+07
	speedup	4.4	24.1	45.2
mnist784	iterations	63	60	165
	standard	1.134e+07	7.200e+07	9.900e+08
	Elkan	1.625e+06	7.396e+06	3.055e+07
	Suggest	1.688e+06	7.310e+06	2.886e+07
	speedup	6.72	9.85	34.3

이러한 오버헤드로 인하여 조금 감소되었다. 하지만, 일반적인 GLA에 보다 제안하는 방법은 항상 성능상에 우수성을 보였다.

<표 2>를 보면 제안하는 방법은 군집의 수인 k가 증가하면서 효율이 더 증가하는 것을 알 수 있다. 거리 계산 횟수는 k와 반복 횟수인 l이 증가하면 같이 증가하기는 하지만, 그 기울기는 훨씬 작은 값을 가진다. 따라서, 일반적인 GLA의 거리 계산 횟수인 nkl (n은 입력 데이터의 수)에 비하여 작은 값을 가진다.

$k \geq 20$ 인 경우에 Elkan[16]과 Moore[18] 보다 더 좋은 효율을 보이고 있다. “covtype” 데이터 집합을 살펴보면, Moore의 경우 k가 변화함에 따라, 각각 24.8, 11.3, 19.0의 속도 향상이 있었고, Elkan은 8.6, 107, 310의 속도 향상을 보이고 있다. 제안하는 방법은 8.4, 110, 311의 속도 향상을 보이고 있다. 다른 데이터 집합에 대하여 Elkan과 비교한 결과를 살펴보아도, “birch” 데이터 집합을 제외하면, 모두 $k \geq 20$ 인 경우에 Elkan 보다 좋은 결과를 보이는 것을 알 수 있다. “birch”의 경우에는 차원의 수가 작아서 $k = 20$ 의 경우에 제안하는 방법이 더 좋은 결과를 보이지 못하고 있지만, $k = 100$ 인 경우에는 더 좋은 결과를 보이는 것을 확인할 수 있다.

$k = 3$ 인 경우를 전체적으로 Elkan보다 안 좋은 결과를 보이고 있다. 즉, 제안하는 방법은 대용량 데이터에 대해 더 좋은 결과를 보이고, 대용량이 아닐 경우에 즉, 데이터 수가 작거나, 차원이 작거나, 군집의 수가 작은 경우에는 속도를 증가시키는 효율이 떨어지는 것을 알 수 있다.

동일한 반복 횟수 후의 결과를 살펴보면, 제안하는 방법은 GLA에 기반하고 있기 때문에, GLA의 결과와 소속 군집이 동일하고, 소속 군집과의 거리도 동일한 결과를 보이고 있다. 제안하는 방법의 주요 강점은 일반적인 GLA의 군집 성능은 유지하면서, 효율적으로 동일한 결과를 보일 수 있다는 점이다.

제안하는 방법은 거리 계산을 수행하지 않을 데이터를 계산할 때 거리 정보만을 이용하기 때문에 차원에 영향을 적게 받으며, 그 우수성은 실험 결과로 확인 할 수 있었다. 차원을 변화 시킨 경우를 살펴보면 산술적인 기대치 보다 더 높은 속도 향상을 보였다.

5. 결 론

본 논문에서는 삼각 부등식을 이용해서 군집이 변화하지 않을 데이터를 미리 찾아냄으로써 데이터와 코드 벡터와의 거리 계산 횟수를 줄여서 GLA 알고리즘을 효율적으로 개선한 알고리즘을 제안했다. 제안하는 알고리즘은 GLA, Moore, Elkan의 방법과 비교하여 입력 데이터의 차원과 입력 데이터의 개수, 군집의 개수가 큰 경우에 좋은 성능을 보였다. 하지만, 입력 데이터의 차원, 입력 데이터의 크기, 군집의 개수가 작은 경우에는 제안하는 방법이 GLA보다는 좋지만, 다른 방법 보다는 더 효율적이라고 하기는 힘들다. 따라서, 작은 크기의 데이터에 대해서도 좋은 성능을 보이게끔 제안하는 방법의 개선이 필요하다. 이를 위해서는 제안하는 방법의 초기와 마지막 단계에서의 개선이 필요하다. 이전 단계에서의 거리를 이용하여 현재 단계의 거리를 계산하기 때문에 반복 횟수가 작을 때 오버 헤드가 된다. 마찬가지로, 마지막 단계에서는 전체 거리를 다시 한 번 계산해서 최종 값을 취하게 되는데, 이것도 오버 헤드로 작용하게 된다. 따라서 코드 벡터를 계산하는 과정을 수정해서 변경되는 데이터만 사용해서 코드 벡터를 계산하는 방법을 사용할 필요가 있다.

제안하는 방법은 이전 코드벡터와 현재 코드벡터와의 거리, 이전 코드벡터와 입력 데이터와의 거리를 유지하고 있어야 하기 때문에 일반적인 GLA보다 메모리 사용량은 최악의 경우에 2배정도 더 필요하게 된다. 하지만, 이에 따른 계산 시간 감소라는 측면에서 본다면, 감수할 만한 증가량이라고 할 수 있다.

제안하는 방법은 거리 계산 함수의 종류에 영향을 받지 않는다. 일반적으로 사용하는 유클리디안 거리나 맨하탄 거리와 같은 거리 계산 함수나 코사인 상관계수나 피어슨 상관계수 같은 상관 계수 함수에도 자유롭게 적용이 가능하다.

각 단계에서 모든 코드워드 사이의 거리는 다시 계산해야 한다. 이는 군집의 개수가 작을 때는 상관없지만, 군집의 개수가 많을 경우에는 계산 시간에 영향을 끼치는 요인이 된다. 따라서, 코드워드 사이의 거리 계산을 단축할 수 있는 방법에 대한 연구가 필요하다.

참 고 문 헌

[1] Tzu-Chuen Lu, and Ching-Yun Chang, "A Survey of VQ Codebook Generation," Journal of Information Hiding and Multimedia Signal Processing, vol. 1, no. 3, pp. 190-203, 2010.

[2] Gersho A, and Gray RM, "Vector Quantization and Signal Compression," Boston, MA: Kluwer Academic Publishers, 1991.

[3] Liaw YC, Lai JZC, and Lo W, "Image restoration of compressed image using classified vector quantization," Pattern Recognition 35: 329-340, 2002

[4] Theodoridis S, and Koutroumbas, "Pattern Recognition. 2nd edn.," New York: Elsevier Academic Press, 2003.

[5] Lai JZC, and Liaw YC, "Fast-searching algorithm for vector quantization using projection and triangular inequality," IEEE Trans. Image Processing 13: 1554-1558, 2004

[6] C. W. Tsai, C. Y. Lee, M. C. Chiang, and C. S. Yang, "A Fast VQ Codebook Generation Algorithm via Pattern Reduction, Pattern Recognition Letters", vol. 30, pp. 653-660, 2009.

[7] Xiao-Gang W, and Yue L, "Web mining based on user access patterns for web personalization," ISECS International Colloquium on Computing, Communication, Control, and Management. 1: 194-197, 2009.

[8] Gan G, Ma C, and Wu J, "Data Clustering: Theory, Algorithms, and Applications," Philadelphia, PA: SIAM, 2007.

[9] Krishnamoorthy R, Kalpana J, "Minimum distortion clustering technique for orthogonal polynomials transform vector quantizer," Proc. 2011 Inter. Conf. Communication, Computing & Security. 443-448, 2011.

[10] Dumitrescu S, Wu X, "On properties of locally optimal multiple description scalar quantizers with convex cells," IEEE Trans. Inform. Theor. 55: 5591-5606, 2009.

[11] Lai JZC, and Lue CC, "Fast search algorithms for VQ codebook generation," J. Vis. Comm. Image Represent. 7: 163-168, 1996.

[12] Huang CM, Bi Q, Stiles GS, and Harris RW, "Fast full search equivalent encoding algorithms for image compression using vector quantization", IEEE Trans. Image Process. 1: 413-416, 1992.

[13] Pelleg D, and Moore A, "Accelerating exact K-means algorithms with geometric reasoning," Proc. ACM SIGKDD. 277-281, 1999.

[14] Kanungo T, Mount D, Netanyahu N, Piatko C, and Silverman R, "An efficient K-means clustering algorithm: Analysis and implementation," IEEE Trans. Pattern Anal. Mach. Intell. 24: 881-892, 2002.

[15] 김성재, 안철응, 김승호, "벡터 양자화를 위한 삼각 부등식을 이용하는 빠른 코드북 탐색법," 한국정보과학회 1998년도 가을 학술발표논문집 제25권 제2호(II), pp. 526-528, 1998.

[16] Charles Elkan, "Using the Triangle Inequality to Accelerate k-Means," Proceedings of the Twentieth International Conference on Machine Learning, 147-153, 2003.

[17] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: an efficient data clustering method for very large databases," Proceedings of the ACM SIGMOD, pp. 103-114, 1996.

[18] A.W. Moore, "The anchors hierarchy: Using the triangle inequality to survive high dimensional data," Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence, pp. 397-405, 2000.

[19] KDDCUP 1993, www.cs.purdue.edu/commugrate/data/kddcpu/1998

[20] MNIST Dataset, yann.lecun.com/exdb/mnist

[21] Ji He, Man Lan, Chew-Lim Tan, Sam-Yuan Sung and Hwee-Boon Low, "Initialization of clusters refinement algorithms: a review and comparative study," International Joint Conference on Neural Networks, pp. 25-29, 2004.

이 현 진

1996년: 순천향대학교 전산학 학사

1998년: 연세대학교 대학원 컴퓨터과학 석사

2002년: 연세대학교 대학원 컴퓨터과학 박사

2003년 ~ 현재: 숭실사이버대학교 컴퓨터정보통신학과 부교수

관심분야 : 이러닝, 기계학습, 데이터마이닝, 클라우드 컴퓨팅