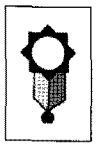


대용량 자료 분석을 위한 밀도기반 이상치 탐지*

김 승** · 조남욱***† · 강석호**

Density-based Outlier Detection for Very Large Data

Seung Kim** · Nam Wook Cho*** · Suk-ho Kang**

■ Abstract ■

A density-based outlier detection such as an LOF (Local Outlier Factor) tries to find an outlying observation by using density of its surrounding space. In spite of several advantages of a density-based outlier detection method, the computational complexity of outlier detection has been one of major barriers in its application.

In this paper, we present an LOF algorithm that can reduce computation time of a density based outlier detection algorithm. A kd-tree indexing and approximated k-nearest neighbor search algorithm (ANN) are adopted in the proposed method. A set of experiments was conducted to examine performance of the proposed algorithm. The results show that the proposed method can effectively detect local outliers in reduced computation time.

Keyword : Density Based, Outlier Detection, Anomaly Detection Kd Tree, Local Outlier Factor(LOF)

논문접수일 : 2010년 03월 30일 논문게재확정일 : 2010년 05월 13일
논문수정일(1차 : 2010년 05월 12일)
* 본 논문은 2009년도 한국경영과학회 추계학술대회 경쟁부문(이론) 우수논문상 수상논문으로 소정의 심사과정을 거쳐 게재 추천되었음.
** 서울대학교 산업공학과
*** 서울산업대학교 산업정보시스템공학과
† 교신저자

1. 서 론

데이터 마이닝 기법은 주로 전체 데이터 베이스로부터 빈번하고 중요한 데이터 패턴, 즉, 잠재적으로 유용성이 있을법한 미지의 사실, 지식을 찾아내는 방법임에 반해 이상치 탐지는 그와 반대로 대부분의 데이터 개체가 생성된 메커니즘과는 다른 방법으로 만들어졌을지 모르는 소수의 데이터 개체를 찾아내는 것을 목적으로 한다.

이상치 탐지는 최근 여러 분야에서 중요성을 인식 받고 있으며, 특히 신용카드 부정사용 등의 금융사기 탐지[24], 휴대폰 불법 복제[9], 키보드 타이핑 기반의 사용자 인증[16], 컴퓨터 네트웍의 불법 침입 탐지[26], CCTV를 통한 영상 감시[19] 등에서 활용되고 있다.

데이터 마이닝 또는 머신 러닝 기법의 관점에서 본다면 이러한 이상치는 빈번한 데이터 패턴의 형성에 기여하지 못하는 노이즈로 인식된다. 또한, 통계학적인 관점에서 본다면 이상치는 확률밀도함수에서 상대적으로 낮은 발생확률을 갖는 개체로 볼 수 있다.

인터넷의 출현과 컴퓨팅 파워의 증가로 인해 여러 분야에서 방대한 양의 데이터가 수집되고 있으나 이상치 탐지를 위해 기존에 사용되어 왔던 기법은 여러 가지 한계점이 존재한다. 통계학적인 관점의 연구[17, 22]는 이상치 탐지를 위해 데이터 개체의 발생확률에 대한 분포의 가정이 필요하다는 점과 분포의 가정이 적당하다 하더라도 분포의 모수를 추정하기 어렵다는 점에서 적용에 어려움이 있다. 또한 데이터 개체가 다변량(multi attribute)인 경우 활용에 제약을 받을 수 있다.

지도학습(Supervised learning) 방식의 클러스터링 기법[2, 8, 14, 18, 25]에서는 이상치를 정상적인 개체들로 이루어지는 클러스터에 속하지 않는, 소수의 비정상 개체로 형성되는 클러스터에 포함되는 개체로 인식한다. 하지만 1) 대량의 훈련데이터가 수집되어야 한다는 점, 2) 정상 클래스와 비정상 클래스의 수작업 분류가 필요하다는 점, 3) 정상과 비정상을 분류해줄 분류기(classifier)의 성능이 훈련집합(training set)의 질에 좌우된다는 점, 4) 학습에서 이용되지 않은 새로운 형태의 비정상 개체에 대해서는 분류가 곤란해진다는 점, 5) 비정상 클래스에 속하는 개체의 수가 정상 클래스에 속하는 개체의 수에 비해 극소하므로 정상클래스에 과도학습(over training)되기 쉽다는 점 등에서 한계가 있다.

최근, LOF(Local Outlier Factor) 알고리즘[7]이 이상치 탐지에 성공적으로 적용되어온 바 있다. LOF 알고리즘은 밀도기반의 이상치 탐지 방식으로 다차원 공간상에서의 이상치는 정상 개체들이 갖는 주변 밀도에 비해 밀도가 극히 낮다는 성질을 이용해 이상치를 탐지한다. LOF 알고리즘은 이러한 주변 밀도를 이용함으로써 기존의 여타 이상치 탐지 알고리즘에 비해 전체 데이터 개체에 대해 이상 정도를 말해주는 하나의 단일 지표값(LOF : local outlier factor)을 계산할 수 있고, 전체 이상치가 아니라 부분 이상치에 대해서도 탐지가 가능하다는 장점을 갖는다.

LOF 알고리즘은 개체 데이터의 차원에 따라 시간복잡도가 달라진다. 데이터 개체의 수를 n이라 할 때 차원의 수가 적당할 때는 $O(n\log n)$의 시간복잡도를 가지지만, 차원의 수가 커질 때에는 $O(n^2)$의 복잡도를 갖게 된다[7]. 하지만 실제 데이터는 개체의 개수도 클뿐더러 차원의 수도 커지므로 LOF 알고리즘을 적용하기에 수행시간 측면에서 무리가 생기며, 특히 즉각적인 이상치 탐지의 요구가 큰 상황에서는 더욱 그러하다.

본 연구에서는 LOF 알고리즘을 대용량 데이터에 적용하기 위하여 계산시간을 줄이기 위한 방안에 대해서 모색한다. 효율적인 밀도계산을 위해 kd-tree 색인 구조를 사용하여 데이터 개체들의 밀도계산 시간을 줄이고 과도한 계산 부하를 요구하는 정확한 밀도를 계산하는 대신 밀도의 추정치를 사용함으로써 속도향상을 꾀하였다. 제안된 기법의 타당성 입증을 위해 다양한 데이터 에 대한 실험을 진행하였다.

2. 배경이론

LOF 알고리즘[7]의 기본 아이디어는 각각의 전

체 데이터 개체에 대해 개별적인 개체 마다 이상치 정도를 나타내는 측정치를 계산하는 것이다. 이러한 측정치를 local outlier factor(LOF)라고 한다. 다른 개체들에 비해 가장 가까운 주변 개체들과의 밀도가 낮게 측정되는 개체는 높은 LOF 값을 가지며 강한 이상치임을 암시하게 되고, 다른 개체들에 비해 가장 가까운 주변 개체들과의 밀도에 큰 차이가 없으면 이상치가 아닌 것으로 판단한다. LOF 알고리즘은 다음의 절차로 이루어진다.

1) 모든 데이터 개체 q에 대해 *k-distance*(q)를 q와 q의 k번째 인접이웃(k-th nearest neighbor; k-nn) 사이의 거리로 정의 하고 계산한다.
2) 모든 데이터 개체 p와 q사이의 도달가능거리(reachability distance)를 아래의 식으로 구한다.

$$reach\text{-}dist_k(q, p) = \max\{d(q, p), k\text{-}distance(p)\} \quad (1)$$

단, d(q, p)는 p와 q사이의 유클리드 거리이고, k는 LOF[7] 알고리즘상의 *MinPts*로 정의되는 파라미터다. k(= MinPts)는 LOF 알고리즘에서 이상치로 판단시키지 않게 하고 싶은 최소 군집의 크기(cardinality) 보다 같거나 작은 값이다.

3) 데이터 개체 q의 국소 도달가능밀도(local reachability density; lrd) 계산

$$lrd_k(q) = 1/\left(\frac{\sum_{p \in N_k(q)} reach - dist_k(q, p)}{|N_k(q)|}\right) \quad (2)$$

단, $N_k(q)$는 q의 k번째 인접이웃(k-nn)집합. 개체 q의 lrd는 위 식에서와 같이 개체 q의 k-nn집합에 속하는 개체들의 평균적인 도달가능거리의 역수이다.

4) 데이터 개체 q의 LOF 계산

$$LOF_k(q) = \frac{\sum_{p \in N_k(q)} \frac{lrd_k(p)}{lrd_k(q)}}{|N_k(q)|} \quad (3)$$

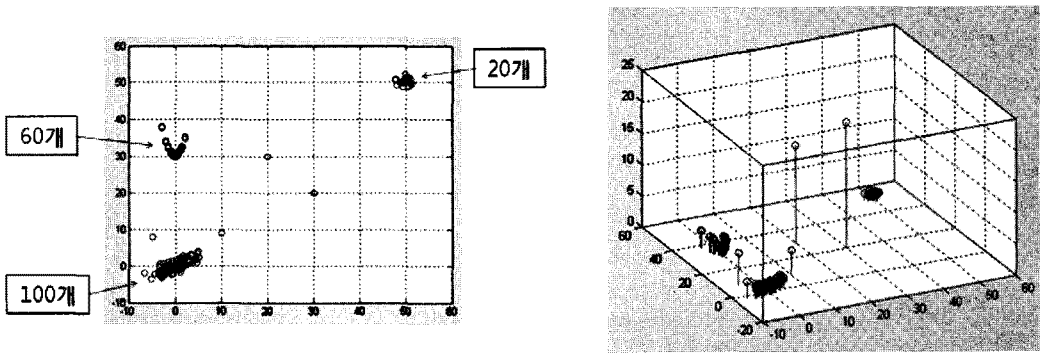
여기서, 데이터 개체 q의 LOF는 위 식에서와 같이 개체 q의 lrd와 개체 q의 k개로 이루어진 인접이웃 집합에 속하는 개체들의 평균적인 lrd와의 비율로 계산된다.

데이터 개체의 LOF 값은 자기 자신의 밀도와 주변 개체들의 밀도와 비교되어 주변 개체들과의 밀도가 비슷한, 다시 말해 클러스터 내 개체에 대해서는 개체들 간의 밀도차이가 거의 나지 않으므로 그 값이 식 (3)에 따라 1로 근접하게 되고, 주변 개체들과의 밀도가 자신의 이웃 개체들의 밀도에 비해 차이가 많이 나는 경우에는 그 정도에 비례하여 1을 상회하는 값을 갖게 된다.

기존의 통계학적인 방법이나 데이터 마이닝 또는 클러스터링 기법과 비교했을 때 이와 같은 다차원 공간상의 밀도에 기반을 둔 LOF 접근법의 장점은 아래와 같다.

1) 데이터 개체들의 전체적인 이상치뿐만 아니라 k값으로 조정할 수 있는 지역적인 이상치도 검출이 가능하다.
2) 데이터 개체들의 분포에 대한 가정이 필요치 않으므로 개체들의 통계적 분포에 무관하게 이상치를 검출할 수 있다.
3) 이상치 계산에 있어 분포의 가정이 필요할 때 부가되는 파라미터가 필요치 않으며, 정상치 군집으로 판단 할 수 있는 최소한의 군집내의 개체 수, k만을 파라미터로 필요로 한다.
4) 모든 개체에 대해 이상치 정도를 단일 지표값으로 나타낼 수 있다.

[그림 1]은 임의로 만들어낸 2차원의 184개 데이터 개체에 대해서 LOF 값을 계산하고 그 값을 도시한 것이다. 각각 60개, 20개, 100개로 이루어지는



[그림 1] LOF 계산 예제(Min_pts = 15)

군집들에 포함되는 개체들의 LOF 값은 1에 가깝게 나오고 있으나 (5, 9), (10, 10), (20, 30), (30, 20)에 위치하는 개체들에 대해서는 가장 가까운 15개의 주변 개체들과의 밀도의 차이에 따라 LOF 값이 1을 훨씬 상회하도록 계산이 되어 이상치로 검출된다.

LOF 알고리즘은 이전의 이상치 탐지 기법들에 비해 여러 가지 장점을 가지고 있으나 LOF 값의 계산시간 측면에서 불리한 점이 많다. 데이터 개체가 동적으로 증가하거나 DB에서 개체가 삭제되거나 변화하는 경우에는 전체 데이터 개체의 LOF 값을 처음부터 다시 계산해야 한다는 단점이 있어 이를 회피하기 위해 Pokrajac et al.은 LOF 값 계산을 점진적으로 계산하는 알고리즘[20]을 제안하였으며, Agyemang et al.은 LOF 알고리즘의 계산 시간을 줄이기 위해 LOF 알고리즘과 유사한 밀도기반의 이상치 탐지 기법인 LSC-Mine 알고리즘[3]을 제안하였다. LSC-Mine 알고리즘은 LOF 알고리즘이 위 식 (1)에서와 같이 max{d(q, p), k-distance(p)}로 정의되는 개체 q의 도달가능거리 계산에 있어 개체 q의 k-nn에 포함되는 모든 개체에 대해 개체 q와의 거리를 계산해야 하며, 이 계산 부하는 k가 커질수록 매우 커진다는 점과, 전체 데이터 개체중 소수의 이상치 탐지를 위해 전체 데이터 개체를 대상으로 LOF를 계산해야 한다는 단점을 보완하여, 도달가능거리 계산이 필요한 도달가능밀도(lrd) 대신에 개체간의 실거리를 사용하는 이상치 비율(local sparsity ratio)을 사용하고, 최종적인 이상치(local sparsity coefficient) 계산 이전에 이상치가 아닐 것으로 예상되는 개체는 최종 이상치 계산에서 제외시키는 프루닝 요소(pruning factor)를 제안하였다.

LOF 알고리즘의 계산부하의 원인을 LSC-Mine에서 착안한 도달가능거리 계산의 생략이나 프루닝 접근 외에 더욱 근본적인 관점에서 다음 두 가지 측면으로 생각해볼 수 있다. 먼저 데이터 개체의 차원의 수가 커질수록 "Curse of Dimensionality"로 표현되는 데이터 개체가 분포하는 공간 차원의 확대 문제이다. 공간의 차원이 커질수록 LOF 계산에 필요한 국소 도달가능거리 및 LOF 계산에 기반이 되는 유클리드 거리 및 인접이웃 계산의 부하가 지수적으로 증가하게 된다. 이를 위해서는 Singular Value Decomposition(SVD)[21, 23]이나, Karhunen-Loève(KL)[10, 13] 또는 FastMap[10]과 같은 차원 감축 기법이 적용될 수 있다. Aggarwal 과 Yu는 [1]에서 높은 차원 공간에서의 이상치 탐지에 있어 최적의 차원감축 프로젝션을 찾기 위해 유전연산에 기반을 둔 알고리즘을 사용하였다.

LOF의 계산 부하를 증가시키는 두 번째 요소는 k 인접이웃(k nearest neighbors)의 계산에 있다. k 인접이웃 계산은 실수의 d차원 공간에 존재하는 n개의 데이터 개체, $[p_i]_{i=1\sim n}$로 구성되는 데이터 집합 $S \subset R^d$에서 주어진 쿼리 개체 $q \in R^d$에 대해 q까지의 유클리드 거리가 k 번째까지 작은 모든 데이터 개체를 S에서 찾는 문제라 할 수 있다. 이 문

제는 간단히 S 내부의 모든 데이터 개체와 q까지의 거리를 계산하고 이를 정렬하여 얻을 수 있다. 그러나 이러한 단순(brute-force) 계산은 앞서 언급한데로 차원의 크기 d와 S의 크기 n이 매우 큰 현실적인 데이터에 적용하기에는 힘들뿐더러, 데이터 셋의 크기가 커질수록 지역적 이상치로 판단하지 않아야 할 최소 군집 크기(cardinality)를 의미하는 k 역시, 현실적인 이상치의 의미를 갖기 위해서는, 비례적으로 커져야 하므로 LOF 계산량 또한 k가 증가함에 따라 상당한 계산 부하를 생성시킨다.

LOF의 계산에 있어 필요한 k인접이웃 계산의 탐색시간을 줄이기 위해 본 연구에서는 먼저, kd-tree 색인 구조[6, 11]를 사용하여 최종적인 LOF 계산시간을 줄였다. 또한, 이에 더하여 kd-tree 색인 구조에 기반한 approximated k-nn 탐색 알고리즘(ANN)[4] 을 사용하여 LOF 계산의 질 저하를 최소화하며 계산시간을 획기적으로 감소시켰다.

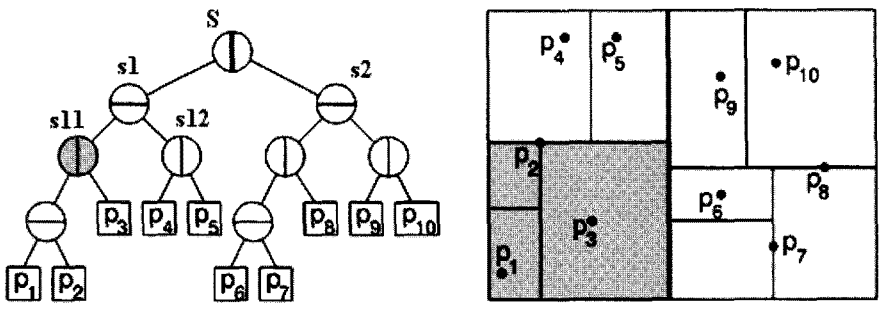
3. 대용량 데이터 분석을 위한 LOF 방법론

3.1 kd-tree Indexing and Querying

k-nn 계산에 있어 앞서 언급한 단순 계산법은 추가적인 데이터 색인 구조를 요구하지 않는 대신에 과도한 계산시간을 요구한다. kd-tree 데이터 색인 구조와 탐색 알고리즘은 Friedman et al.이 [12]에서 처음 제안하였으며 Arya and Mount는 [4]에서 kd-tree를 기반으로 탐색과정에서 적정 수준의 불확실성을 추가함으로써 고차원 탐색공간에서 탐색의 효율성을 높일 수 있는 approximated k-nearest neighbor 알고리즘을 제안하였다.

kd-tree 색인은 k차원의 데이터 공간상에 존재하는 데이터 개체들을 구조화 하기위해 공간을 분할해가며 데이터 개체를 구조화 시켜가는 데이터 색인 구조이다. 기본적인 kd-tree 색인은 데이터 집합 S를 포함하는 초월공간(hyperspace)에서 시작한다. 이 최초의 공간은 더 작은 초월공간으로 반복적으로 분해된다. 데이터 공간을 분할하는 초월평면(hyperplane)을 어떤 방향으로 분할할지는 미리 정의된 분할규칙(splitting rule)에 따른다. 이 반복적인 분할과정은 분할로 생성되는 초월공간에 포함된 개체수가 미리 정해진 할당량(bucket-size) 만큼 작아질 때까지 계속된다. 따라서 완성된 트리의 잎(leaf) 노드는 할당량 이하로 개체의 수가 줄어든 공간에 대응된다.

[그림 2]는 (b)에서 주어지는 2차원 공간의 10개 데이터에 대해 할당량이 1인 경우에 생성된 kd-tree의 예를 (a)에서 보여주고 있다. 원으로 표시된 내부 노드에서 가로와 세로선은 그에 해당하는 분할의 방향을 의미한다. [그림 2] (a)의 뿌리 노드는 전체 데이터 개체 S를 포함하는 사각형에 대응하고 이를 좌, 우의 세로 방향으로 분할하게 되면 p1, p2,



(a) (b)

[그림 2] 2차원 kd-tree 공간의 분할

p3, p4, p5를 포함하는 좌측 사각형 (s1)과 p6, p7, p8, p9, p10을 포함하는 우측 사각형 (s2)로 나뉘게 된다. 다음번의 분할은 상, 하의 가로 방향의 분할이 되어 s1은 p1, p2, p3을 포함하는 아래쪽 사각형 (s11)과 p4, p5를 포함하는 위쪽 사각형 (s12)가 된다. 이러한 가로방향과 세로방향의 분할을 계속 이어나가 미리 정의된 할당량만큼의 개체가 분할된 공간에 존재하게 될 때까지 트리의 구성은 계속 이어진다.

이러한 자료구조를 이용할 때 인접이웃 탐색에 영향을 주는 요인으로는 분할 규칙(splitting rule), 탐색기법(search algorithm), 할당량(bucket-size)이 있다.

분할 룰은 데이터 공간을 나누는 기준이 된다. 어떤 룰을 따를지는 데이터 개체의 분포에 영향을 받는다. 널리 알려진 방법은 standard-split[12]와 midpt-split[12] 및 fair-split[4]이 있다. Standard-split은 분할되는 차원에 대한 최소값과 최대값의 중앙값(median)에 수직방향으로 공간을 분할한다. 이렇게 분할할 때 트리는 항상 $\lceil \log_2 n \rceil$ 을 갖게 되고 크기는 O(n)이 되어, 트리의 구조는 탐색에 유리하게 형성되나, node들에 해당하는 공간의 가장 긴 방향의 차원의 길이와 가장 짧은 차원의 길이의 비율인 aspect ratio가 임의의 높은 값을 가질 수 있다. Midpt-split은 항상 분할되는 차원의 중간 값을 기준으로 차원을 분리한다. 이 분할은 공간의 aspect ratio 측면에서는 유리하나, 한 차원에서 모든 데이터가 차원의 중간 값보다 이하의 공간이나, 이상의 공간에 모두 위치하게 되는 경우에는 trivial split을 만들게 되고, 이에 따라 트리의 size가 불필요하게 매우 커질 수 있다. Fair-split은 다차원 개체 공간의 가장 긴 방향의 차원의 길이와 가장 짧은 차원의 길이의 비율이 항상 3 : 1을 넘지 않도록 분할을 이어간다. Fair-split을 쓰면 적절한 aspect-ratio도 유지하며, 트리의 형상도 균형 잡힌 형태를 만들 수 있으므로 standard-split과 midpt-split의 장점을 모두 갖고 있다고 볼 수 있고, 결국 이러한 특징은 탐색의 효율에 영향을 준다. [그림 3]은 [그림 1]의 데이터에 대한 각 3개의 분할방법의 차이를 보여준다.

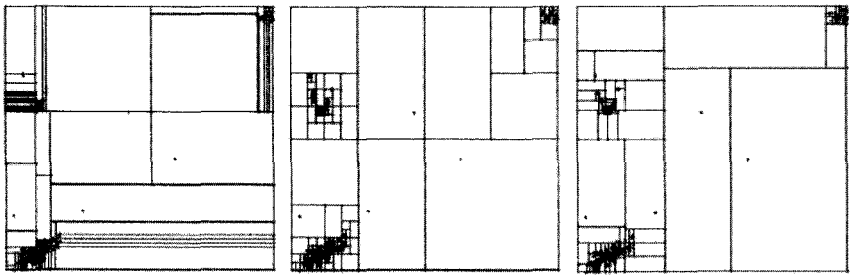
kd-tree에서 사용할 수 있는 탐색기법 중 대표적으로 알려진 방법에는 표준탐색(standard search)과 우선탐색(priority search)이 있다. 전자는 주어진 대상 개체가 어떤 리프노드(leaf node)에 속하는지를 루트 노드에서부터 아랫방향으로 탐색해가며 찾아간 후 리프노드에 도착하게 되면 리프노드에 속한 모든 개체들과의 정확한 거리를 계산하여 인접이웃을 찾는다. 이런 방식의 트리 탐색을 통해 해당 노드에 포함되는 소수의 개체들에 대해서만 쿼리 개체와의 거리계산을 할 수 있기 때문에 인접이웃 계산시간이 대폭 줄어들게 된다. 만약 리프노드에 속하는 개체수가 k개 이상일 때는 인접이웃 계산에서 추가적인 리프노드로의 탐색이 필요하지 않으나, 리프노드에 포함된 개체수가 k개 이하일 때는 해당 리프노드를 나타내는 초월공간과 절단면을 공유하는 공간에 대응하는 리프노드들을 추가적으로 탐색해 가며 인접이웃을 찾게 된다.

우선탐색 기법은 트리의 루트노드로부터 쿼리 개체를 포함한 리프노드 까지 차례로 트리를 탐색해가며 찾아가는 과정에서 쿼리 개체와 트리 노드의 경계가 가장 가까운 순서대로 우선순위 대기열(priority queue)을 형성한다. 말단노드에까지 이르는 대기열이 만들어지면 이후 쿼리개체를 포함한 노드들과의 거리가 가까운 순서대로 노드를 방문해가며 인접이웃을 탐색하게 된다. 쿼리 개체와 대기열에 남은 미방문 노드와의 거리가 지금까지 찾은 k-nn과의 거리보다 크면 탐색이 종료된다.

이러한 방법으로 kd-tree 색인을 사용하게 되면 LOF 계산에 필요한 인접이웃 계산 시간의 감소를 가져올 수 있으며 이는 LSC-Mine을 사용하는 이상치 탐지에서도 똑같이 활용될 수 있다. 구체적인 계산시간의 감소는 제 4장에서 설명한다. 또한, kd-tree에 기반한 approximated k-nearest neighbor indexing 및 querying을 사용하면 더욱더 큰 폭의 계산시간 감소를 가져올 수 있게 된다.

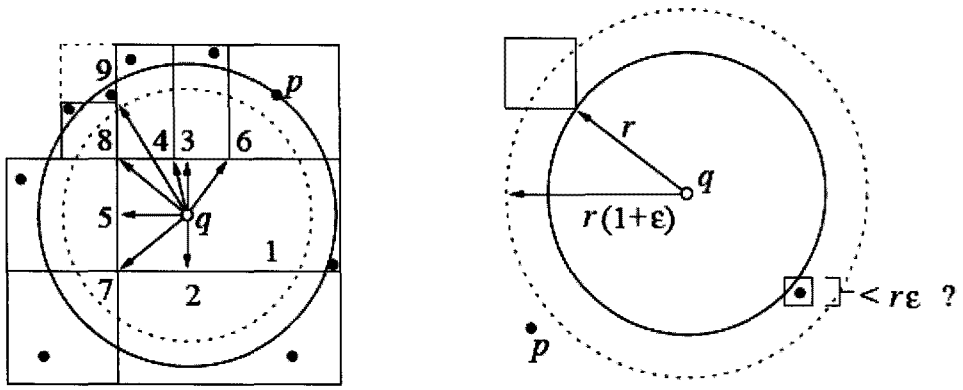
3.2 Approximated k-nearest Neighbor(ANN)

kd-tree 색인구조를 사용하면 인접이웃 계산시간



(a) standard-split (b) midpt-split (c) fair-split

[그림 3] kd-tree 분할 방법의 차이



(a) (b)

[그림 4] ANN search algorithm의 탐색 전략(Arya et al.[4])

을 크게 줄일 수 있으나 대용량 데이터 처리 시 요구되는 수준으로 줄이기는 어렵다. 따라서 계산을 정확하게 하기 보다는 감수할 수 있을 만큼의 불확실성을 추가하여 계산의 질의 감소는 최소화 하면서 데이터 색인으로 얻을 수 있는 계산시간의 이점에 추가적인 이점을 얻을 수 있는 방법이 Arya et al.이 제안한 Approximated k-nearest Neighbor(ANN) [4] 알고리즘이다. 정해진 오차항 $\varepsilon(>0)$에 대해 만약 쿼리 개체를 q라 하고 q의 실제인접이웃을 p'라 하며, 알고리즘이 계산해내는 근사인접이웃(approximated nearest neighbor)을 p라 할 때, p와 q와의 거리 dist(p, q)는 $(1+\varepsilon)$dist(p', q)보다 항상 같거나 작게 인접이웃을 추정할 수 있다. 즉, ε는 p와 p'사이의 최대 에러(maximum relative error)이다.

ANN의 개념을 [그림 4]에 도식화 하였다. q를 쿼리 개체라고 할 때 우선탐색(priority search)과정에서 쿼리 개체 q와 데이터 개체를 포함한 셀(셀; 트리 노드가 나타내는 공간)과의 거리에 따라 탐색 대상 셀의 순서가 정해진다. [그림 4]에서 p가 6번 셀까지 방문했을 때 현재까지의 탐색과정에서 찾아진 가장 가까운 개체라 하자. 이때, 아직 방문하지 않은 7, 8, 9의 셀 중 q로부터 셀의 경계까지의 거리가 dist(p,q)/(1+ε)보다 가까운 경우에는 해당 셀 내부의 점들과 q간의 거리를 비교하여, p를 갱신한다.

[그림 4]의 (a)에서 점선으로 표시된 원은 p가 가장 가까운 개체인 상황에서, dist(p, q)/(1+ε)를 반지름으로 중심을 q로 하는 원이다. 이때, 추가적인 탐색 대상에서 9번 셀은 q로 부터의 거리가 dist(p, q)/(1+ε)를 초과하므로 추가적인 탐색대상에서 제외되고, 실제 근접점은 9번 셀의 개체임에도 불구하고, 현재까지 발견된 가장 가까운 개체인 p를 가장 가까운 개체로 판단하여 탐색을 종료한다. 즉,

ANN 탐색에서는 실제인접이웃과 쿼리개체와의 거리를 r이라할 때 r(ε-1) 만큼 실제보다 멀리 떨어진 개체를 인접이웃으로 판단할 수 있다.

개체 p의 LOF 계산은 개체 p의 주변밀도(lrd)와 그 개체의 k-nn에 포함되는 개체들의 평균적인 주변밀도와의 비율이 되고, 이러한 밀도는 그 개체와 그 개체의 k-nn에 포함되는 개체들과의 평균적인 도달가능거리의 역수가 된다. 만약 k-nn의 탐색에서 ANN을 사용하게 되면 도달가능거리가 정확하지 않은 추정 값이 되고, 이로 인해 주변밀도 계산 값이 정확하지 않은 추정 값이 되게 되어 LOF 값 자체도 정확한 값이 아니라 추정 값이 된다.

LOF 값의 계산 식 (3)은 밀도의 비율이며, 모든 개체들에 대한 이 밀도(lrd)의 계산에서 항상 모든 개체에 대해 동일한 정도만큼의 오차가 비례하여 발생한다면, 즉, 오차 발생에서 일관성이 존재한다면, 근사인접이웃을 이용하여 lrd를 계산하더라도 최종적인 LOF 값에는 전혀 변화가 없게 된다. 밀도의 비율인 LOF 식의 분자, 분모항의 개입오차가 서로 상쇄되기 때문이다. 따라서 인접이웃 계산에 소요되는 과다한 시간을 허비하지 않고 LOF 값의 질의 저하 없이 개체들의 LOF 값을 훨씬 빨리 계산할 수 있게 된다. 이를 정리하면 다음과 같다.

정리 1 : 모든 데이터 개체에 대한 밀도(lrd) 계산에서 k-nn과의 거리에 동일한 비율로 비례하는 오차를 가질 때 산출된 LOF 값은 정확한 밀도 값을 이용하여 계산한 LOF 값과 동일하다.

(증명)

데이터 개체 집합 S의 임의의 개체 q에 대해, q의 LOF 식은 다음과 같이 쓸 수 있다. 단, avg는 cardinality k인 평균을 나타냄.

$$LOF_k(q) = \frac{\sum_{p \in N_k(q)} \frac{lrd_k(p)}{lrd_k(q)}}{|N_k(q)|} = \frac{\sum_{p \in N_k(q)} lrd_{k(p)}}{|N_k(q)| \times lrd_k(q)} = \frac{avg(lrd_k(p)}{lrd_k(q)} \quad (4)$$

$$= \frac{avg\left(\frac{|N_k(p)|}{\sum_{r \in N_k(p)} reach - dist_k(q,\ p)}\right)}{\frac{|N_k(p)|}{\sum_{p \in N_k(q)} reach - dist_k(q,\ p)}}$$

q와 q의 k번째 실제인접이웃 사이의 거리를 r_1로 두고, $p(\in N_k(q),\ |N_k(q)| = k)$와 p의 k번째 인접이웃사이의 거리를 r_{2i} $(i = 1,\ 2,\ \cdots,\ k)$로 두면, $LOF_k(q)$는 식 (5)가 된다.

$$LOF_k(q) = \frac{avg\left(\frac{|N_k(p)|}{\sum_{r \in N_k(p)} reach - dist_k(p,\ r)}\right)}{\frac{|N_k(q)|}{\sum_{p \in N_k(q)} reach - dist_k(q,\ p)}} \quad (5)$$

$$= \frac{\frac{\sum_{p \in N_k(q)}}{|N_k(q)|}\left(\frac{|N_k(p)|}{\sum_{r \in N_k(p)} reach - dist_k(p,\ r)}\right)}{\frac{|N_k(q)|}{\sum_{p \in N_k(q)} reach - dist_k(q,\ p)}}$$

$$= \frac{\frac{k}{k}\left(\frac{k}{\sum_{i=1}^{k} r_{2i}}\right)}{\frac{k}{k \times r_1}} = \frac{\frac{k}{\sum_{i=1}^{k} r_{2i}}}{\frac{1}{r_1}} = \frac{kr_1}{\sum_{i=1}^{k} r_{2i}}$$

모든 개체에 대한 도달가능밀도 계산에서 해당 개체와 그 개체의 k-nn사이의 거리에 비례하는 에러 δ가 발생한다면 이때의 q의 $LOF_k(q)'$는

$$LOF_k(q)' = \frac{\frac{\sum_{p \in N_k(q)}}{|N_K(q)|}\left\{\frac{|N_k(p)|}{\sum_{r \in N_k(p)} reah - dist_k(p,\ r)}\right\}}{\frac{|N_k(q)|}{\sum_{p \in N_k(q)} reah - dist_k(q,\ p)}} \quad (6)$$

$$= \frac{\frac{k}{k}\left(\frac{k}{\sum_{i=1}^{k}(r_{2i} \times \delta)}\right)}{\frac{k}{k \times r_1 \times \delta}} = \frac{\frac{k}{\sum_{i=1}^{k}(r_{2i} \times \delta)}}{\frac{1}{r_i \times \delta}}$$

$$= \frac{kr_1\delta}{\delta \times \sum_{i=1}^{k} r_{2i}} = \frac{kr_1}{\sum_{i=1}^{k} r_{2i}}$$

으로 $LOF_k(q) = LOF_k(q)'$이다. (증명끝)

정리 1에서 살펴본바와 같이 실제 인접이웃와 추정치와의 상대적인 오차 δ가 커지더라도 개체들의 오차발생에 대한 일관성이 유지되면 LOF의 추정에서 질의 저하는 없다. 그러나 추정된 이웃과 쿼리개체와의 거리는 참값을 r이라 할 때, r과 rε사이의 범위 내에 존재하는 개체 중에서 임의로 선택되므로 정리 1의 가정, 즉 개체 들별로 동일하게 적용되는 오차비율 δ는 만족되지 않으며, ANN 탐색은 항상 실제거리에 비해 과대평가하게 되며 국소 도달가능밀도는 항상 실제밀도에 비해 과소평가되게 된다. 이러한 오차가 LOF 값에 끼치는 영향을 정리 2 및 정리 3에 요약하였다.

정리 2 : ANN 탐색을 이용한 LOF 추정 값은 실제의 LOF 값과 비교해서,
(1) 쿼리개체에 대한 k개의 인접이웃 개체들의 국소 도달가능밀도 계산은 정확하게 이루어지고, 쿼리개체 자신의 국소 도달가능밀도 계산은 최대로 과소평가 될 때 LOF 추정 값은 실제 LOF 값에 비해 양의 방향으로 가장 큰 에러를 갖는다.
(2) 쿼리개체에 대한 국소 도달가능밀도 계산은 정확히 이루어지고, 쿼리개체의 k개 인접이웃 개체들의 국소 도달가능밀도 계산이 최대로 과소평가 될 때 LOF 추정 값은 실제 LOF 값에 비해 음의 방향으로 가장 큰 에러를 갖는다.

쿼리 개체와 인접이웃과의 실제거리를 r로 둘 때, r과 rε의 범위 내에서 추정된 인접이웃을 찾게 되는 ANN 탐색은 lrd 계산에서 항상 실제의 lrd보다는 낮게 예측된 값을 계산한다.

LOF 식은 분모항인 쿼리 개체 자신의 lrd와 분자항인 쿼리 개체의 k-nn들의 평균 lrd의 비율이며, 두 lrd 모두 ANN 탐색을 이용해 추정을 하게 되면 과소평가를 하게 된다. 그러나 정리 2에서 살펴본 것과 같이 분자항의 과소평가는 전혀 없고, 분모항의 과소평가가 최대가 되는 상태, 즉, 분자항에서의 인접이웃 탐색은 항상 실제의 인접이웃을 탐색해내고, 분모항에서의 인접이웃 탐색은 항상 실제거리 r보다 ε배 곱해진 거리에 위치하는 개체를 인접이웃으로 추정할 때, LOF의 추정치는 실제 값에 비해 양의 방향으로 최대의 오차를 갖게 된다. 이러한 관계는 LOF의 추정치가 실제값에 비해 음의 방향으로 최대의 오차를 갖게 되는 경우에도 동일한 방법으로 설명할 수 있다.

LOF 추정치의 최대, 최소 값의 범위는 위 정리 2와 같으나, 실제의 추정치는 데이터 개체의 분포에 따라 달라진다. 만약, 쿼리 개체에 대한 인접이웃 추정에서 실제의 인접이웃과의 거리를 r로 두고 인접이웃의 추정을 ε의 bound로 ANN 탐색 한다고 할 때, 실제 데이터가 r과 rε사이의 범위의 중앙, (r(ε-1)/2)을 평균으로, r과 rε 사이의 범위를 $\pm 3\sigma$로 갖는 정규분포를 이룬다고 가정하면 LOF의 추정치는 정리 3과 같은 확률밀도 함수를 갖게 된다.

정리 3 : 전체 데이터 개체들($q_i \in S$, $i = 1, 2, \cdots, N$)의 개체와 kth-nn간의 거리 계산에서 그 실제거리, r_i에 비례하여 오차 ε_i가 발생하되, ε_i가 $N(r_i(\varepsilon-1)/2, (r_i(\varepsilon-1))^2/36)$을 따르고 서로 독립이라면 데이터 개체 q의 LOF의 확률변수 추정치 $LOF_k(q)$'를 확률변수 Z로 둘때, Z는 다음과 같은 pdf를 갖는다.

$$p_z(z) = \frac{b(z)\cdot c(z)}{a^3(z)} \frac{36}{\sqrt{2\pi}kr_1r_{2i}(\epsilon-1)^2}\left[2\Phi\left(\frac{b(z)}{a(z)}\right)-1\right] + \frac{36}{a^2(z)\pi\sqrt{k}r_1r_{2i}(\epsilon-1)^2}e^{\left(-0.5\left(\frac{9\cdot(\epsilon+1)^2)}{(\epsilon-1)^2}+\frac{9k\cdot(\epsilon+1)^2}{(\epsilon-1)^2}\right)\right)}$$

$$a(z) = \sqrt{\frac{36}{r_1^2(\epsilon-1)^2}Z^2+\frac{36\cdot k}{r_{2i}^2(\epsilon-1)^2}} \quad (7)$$

$$b(z) = \frac{18\cdot(\epsilon+1)}{r_1(\epsilon-1)^{2z}}+\frac{18\cdot k\cdot(\epsilon+1)}{r_{2i}(\epsilon-1)^2}$$

$$c(z) = \exp\left(\frac{1}{2}\frac{b^2(z)}{a^2(z)}-\frac{1}{2}\left(\frac{9\cdot(\epsilon+1)^2}{(\epsilon-1)^2)}+\frac{9\cdot k\cdot(\epsilon+1)^2}{(\epsilon-1)^2}\right)\right)$$

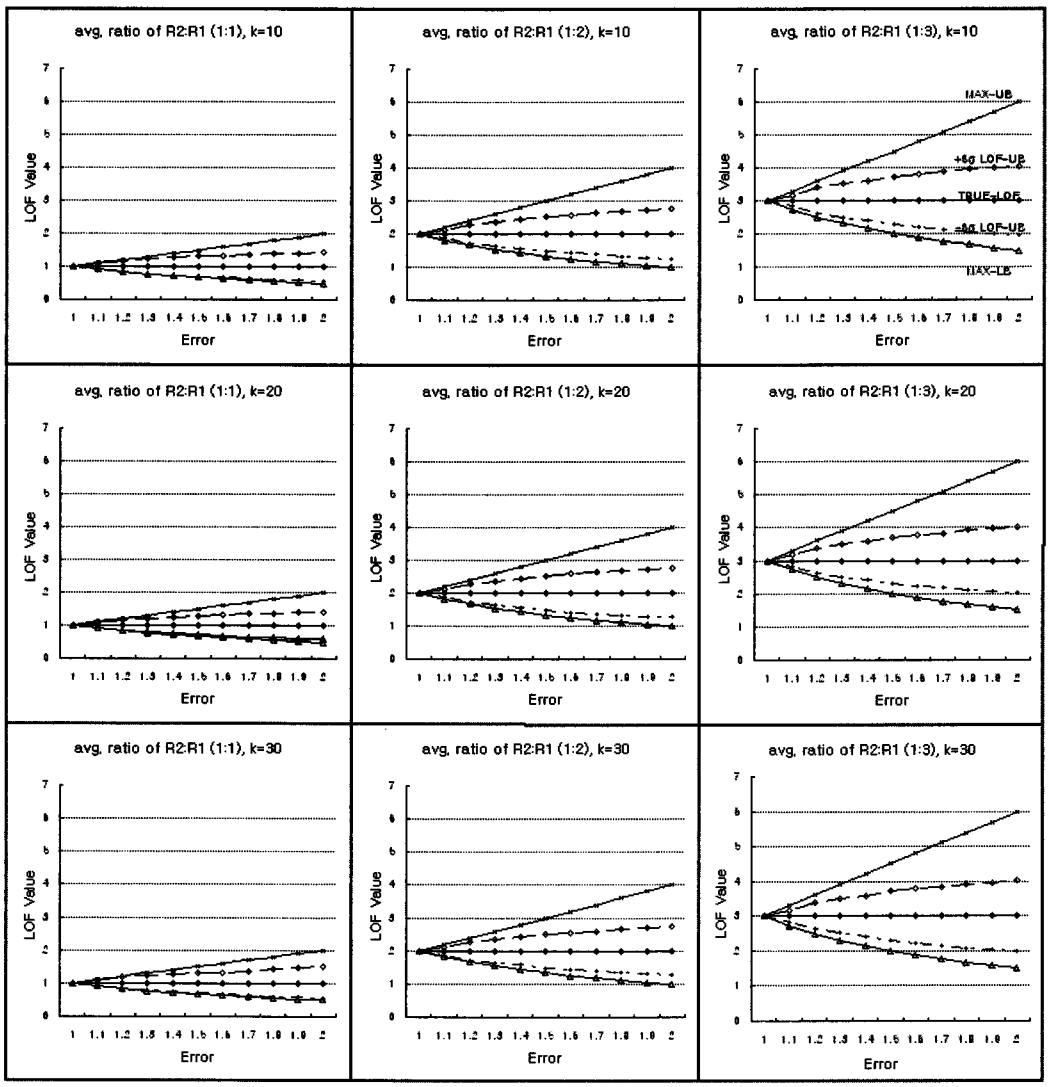
$\Phi(z) = \int_{\infty}^{z}\frac{1}{\sqrt{2\pi}}exp\left(-\frac{1}{2}u^2\right)du$ 이다.

(증명)

q와 q의 k번째 실제 인접이웃 사이의 거리를 r_1라 하자. 추정된 $\hat{r_1}$는 에러의 상한이 ε이고 r_1과 $r_1\varepsilon$ 사이에서 중심이 $r_1(\varepsilon-1)/2$이고 분산이 $r_1^2(\varepsilon-1))^2/36$인 정규분포를 따르는 에러가 r_1에 더해진 값이라고 하자. 여기서 설정한 분산은 $r_1\varepsilon$과 r_1사이의 범위에서 실제로 q의 추정된 인접이웃의 99.73%의 개체가 구해진다고 가정할 때, 즉, $r_1\varepsilon$과 r_1사이의 범위를 $\pm 3\sigma$로 둘 때, 아래의 식 (8)과 같이 계산된 값이다.

$$6\sigma = r_1\epsilon - r_1 = r_1(\epsilon-1) \qquad 8)$$
$$\sigma = \frac{r_1(\epsilon-1)}{6}$$
$$\sigma^2 = \frac{r_1^2(\epsilon-1)^2}{36}$$

이러한 가정 하에 추정된 $\hat{r_1}$은 중심이 $r_1(\varepsilon+1)/2$이고 분산이 $r_1^2(\varepsilon-1)^2/36$인 정규분포를 따르는 확



[그림 5] 실제 LOF와 LOF의 추정치의 확률분포

〈표 1〉 실험 데이터

데이터 셋	개체의 수	속성의 수	설 명
IonoSphere	351	34	17개의 레이더 펄스 신호에 따른 전리층의 구조 데이터
Musk	476	166	Musk의 분자구조, 형상 데이터와 전문가가 이를 musk/non-musk로 구분해 놓은 데이터
Synthetic Control Chart	600	60	임의로 생성된 시계열 데이터

률분포가 된다.

또한, p_i와 p_i의 k번째 실제 인접이웃 사이의 거리를 r_{2i}로 둘 때(i=1, 2, …, k), 추정된 $\hat{r}_{2i}$는 $N(r_{2i}(\varepsilon+1)/2, r_{2i}^2(\varepsilon-1)^2/36)$를 따르는 확률변수가 된다. 또한, 두 확률변수는 가정에 의해 서로 독립이다.

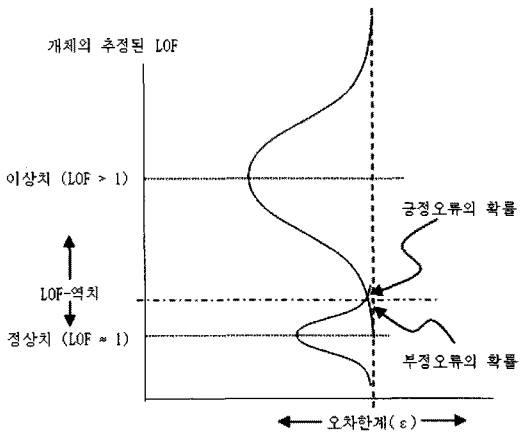
정리 1의 결과에서

$$LOG_k(q)' = \frac{k\hat{r}_1}{\sum_{i=1}^{k}\hat{r}_{2i}} = \frac{\hat{r}_1}{\sum_{i=1}^{k}\hat{r}_{2i}/k}$$

이므로, $\hat{r}_1$는 $N(r_1(\varepsilon+1)/2, r_1^2(\varepsilon-1)^2/36)$을 따르게 되고, $\sum_{i=1}^{k}\hat{r}_{2i}/k$는 정규분포를 따르는 확률변수들의 합의 분포에 대한 평균이므로 $N(r_{2i}(\varepsilon+1)/2, r_{2i}^2(\varepsilon-1))^2/(36*k))$를 따르게 된다. 즉, $LOF_k(q)'$는 정규분포를 따르는 두 확률변수 $\hat{r}_1$과 $\sum_{i=1}^{k}\hat{r}_{2i}/k$의 비율로 나타나는 분포가 된다. 서로 독립이고 평균이 0, 분산이 1인 표준정규분포를 따르는 두 확률변수 X, Y의 비율로 나타나는 확률변수 Z는 코시분포를 따른다는 것은 쉽게 증명할 수 있다[15]. 또한 서로 독립이며, 상관관계가 없는(cor(X, Y) = 0) 두 확률변수 $X \sim N(u_x, \sigma_x^2)$, $Y \sim N(u_y, \sigma_y^2)$의 비율 Z(= X/Y)의 확률밀도 함수를 계산한 Hinkley의 연구[15]를 이용하면 Z의 분포를 식 (7)과 같이 나타낼 수 있다. (증명 끝)

[그림 5]는 정확한 LOF 값과 비교하여 ANN 탐색을 했을 때의 LOF의 추정치의 변화폭을 식 (7)의 확률밀도함수를 이용하여 나타냈다. MAX-UB와 MAX-LB는 정리 2에서 언급한 LOF의 추정치의 이론적인 상한과 하한을 표시한 것이다. 실제로 LOF 추정에 있어 정리 3과 같은 가정이 성립한다면 추정치의 99.73%는 $+3\sigma$ LOF-UB와 -3σ LOF-LB의 범위 내에 존재하게 된다. 이 범위는 ε가 커질수록, 그리고 평균적인 쿼리 개체의 k-nn들의 k번째 인접이웃의 거리와 쿼리 개체의 k번째 인접이웃과의 비율이 커질수록, 즉, 쿼리 개체가 이상치에 가까울수록 추정의 범위가 넓어진다.

LOF의 값을 추정을 통해 얻게 될 때 발생하는 오류를 (1) 이상치가 아닌 개체를 이상치로 판단하는 긍정오류(false positive), (2) 이상치를 아닌 개체로 판단하는 부정오류(false negative)로 나누어 생각해보자. [그림 5]와 [그림 6]에서 보는 것과 같이 긍정오류 확률보다는 부정오류 확률이 동일한 ε에서 더욱 크다. 오류확률은 이상치로 판단할 역치값이 1에 가까울수록 급격히 작아진다. 따라서 추정의 방법을 사용한 이상치 판단의 질은 ε와 LOF-역치의 값에 따라 달라진다. 이상치를 목표변수로 두고 역치 값을 다양하게 변화시켜 가며 정확도



[그림 6] LOF의 추정치의 오류 확률

(precision)와 재현율(recall)을 통해 추정의 질을 판단할 수 있다.

다음 제 4장에서는 주어진 데이터 집합에 대해 kd-tree 및 ANN 색인을 이용하여 LOF를 계산 및 추정함에 있어 얻어지는 속도 향상 및 추정의 질에 대해서 실험을 진행하였다.

4. 실 험

4.1 실험 데이터셋

본 연구에서는 3개의 UCI machine learning repository[5]의 데이터를 사용하여 실험을 진행하였다. 실험에 사용된 데이터를 <표 1>에 정리하였다.

실험의 목표는 단순 알고리즘(brute-force algorithm) 계산에 비하여 kd-tree 색인을 사용했을 때와 ANN 방법을 적용했을 때 각각 생기는 시간상의 이점을 비교 분석하였다. 특히 이상치 검출 능력을 크게 저하시키지 않으면서 시간상의 이점이 큰 방법론을 찾아보고자 한다.

kd-tree 색인 및 ANN을 이용한 LOF의 계산 시간에 영향을 주는 요소로는 앞서 살펴본대로 분할방식(standard-split, midpt-split, fair-split)과 탐색 알고리즘(standard, priority), 할당량(bucket-size), k-인접이웃 계산방법(exact or approximate) 및 ANN에서의 ε의 정도 등이 있다.

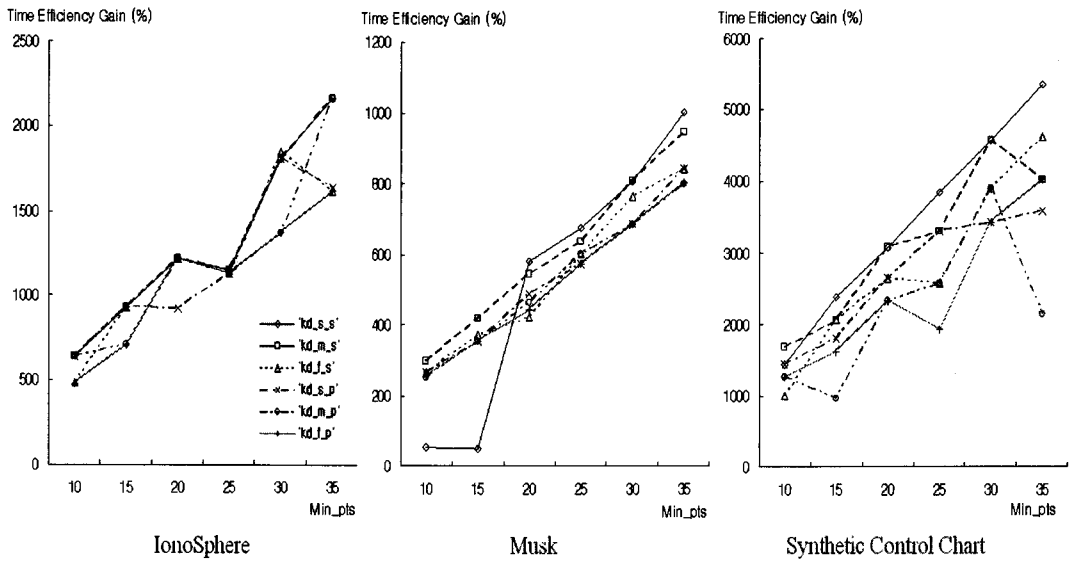
LOF 계산 값의 질을 분석하기 위해 긍정오류 횟수 와 부정오류횟수 및 이를 통해서 얻을 수 있는 정확도와 재현율을 이용하였다. 이때 영향을 주는 요소는 ε와 LOF-역치이다

실험은 2.66GHz Core 2 Duo CPU의 PC에서 수행되었고, 알고리즘은 Mount의 ANN library [27]를 MATLAB 7.0에서 DLL로 호출하여 수행되었다.

4.2 계산시간 측면

먼저 제안된 알고리즘을 kd-tree 색인을 이용한 방법과, ANN을 이용한 방법으로 나누어 수행시간 측면에서 단순 알고리즘과 비교하였다.

[그림 7]은 3개 데이터 셋에 대해 단순알고리즘의 계산시간(t_{brt})대비 kd-tree 색인 방법론의 계산



[그림 7] kd-tree색인기법을 이용한 비교 실험

시간(t_{kd})의 차이를 %단위로 환산하여($100(t_{kd}-t_{brt})$) 세로축에 나타냈다. 또한, 공간 분할 방식의 차이와 탐색 알고리즘의 성능 비교를 위해 standard-split, midpt-split, fair-split과 표준탐색(standard search) 및 우선탐색(priority search)의 6개 조합으로 이루어지는 모든 경우에 대해 LOF 계산시간을 비교하여 나타내었다. 할당량(bucket-size)은 20으로 모든 데이터 셋에 동일하게 사용하였다.

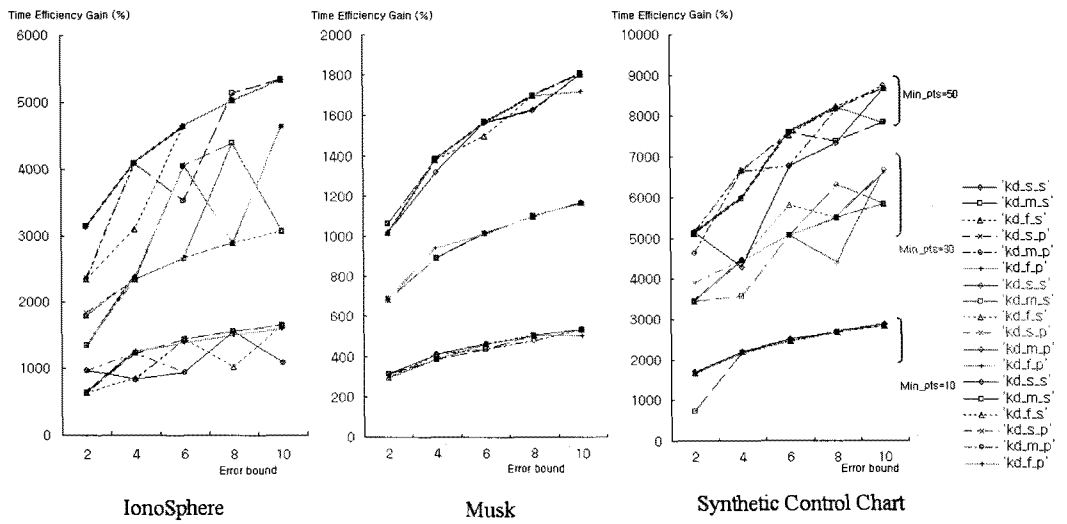
실험결과 속도향상은 최하 49%(Musk, Min_pts = 10, standard split, standard search)에서 최고 5348%(Synthetic Control Chart, Min_pts = 35, standard split, standard search)까지 있었다. 또한, 이러한 속도향상은 Min_pts가 증가할수록 더욱 크게 나타났다. 이는 단순알고리즘 계산에서는 매 데이터 개체별로 전체 데이터 개체에 대한 거리계산을 해야 하고 또한, Min_pts의 크기에 해당하는 데이터 개체들에 대해서 전체 데이터 개체에 대한 거리계산이 필요하므로 이러한 계산 부하 때문에 시간이 많이 걸리지만 kd-tree 색인을 이용하게 되면 이러한 계산을 색인을 이용할 수 있으므로 크게 단축시킬 수 있게 되기 때문이다.

kd-tree의 분할 방법에 따른 성능의 유의한 차이가 있지는 않았다. 다만, 탐색 전략은 우선탐색보다는 표준탐색이 공통적으로 약간의 우위를 보였으나 그 차이는 크지 않았다. 우선탐색은 탐색과정에서 탐색 큐(queue)를 추가적으로 유지하기 때문에 성능에 영향을 끼친 것으로 보이나 이러한 차이는 데이터의 분산된 정도나 이외의 실험설정에 따라 달라질 수 있으므로 일반적인 결론으로 보기는 어렵다.

[그림 8]은 동일한 데이터에 대해 단순알고리즘의 계산시간(t_{brt})대비 ANN의 계산시간(t_{kd})과의 차이를 %단위로 환산하여($100(t_{kd}-t_{brt})$) 가로축에 나타냈다. 할당량(bucket-size)은 kd-tree의 경우에서와 마찬가지로 20으로 동일하게 사용하였으며, 가로축엔 오차한계(ε)를 나타내었다.

실험결과 속도향상은 최하 293%(Musk, Min_pts = 10, fair split, priority search, ε = 2)에서 최고 8727%(Synthetic Control Chart, Min_pts = 50, mid split, priority search, ε = 10)까지 나타났다.

kd-tree 색인만을 이용한 경우와 마찬가지로 Min_pts가 증가할수록, 또한 오차한계가 증가할수록 속도향상이 있었다. 또한, kd-tree 색인만을 이용한 경우와 같이 트리의 분할 방법이나 탐색 전략에서의 성능의 유의한 차이는 크게 나타나지 않았다.



IonoSphere Musk Synthetic Control Chart

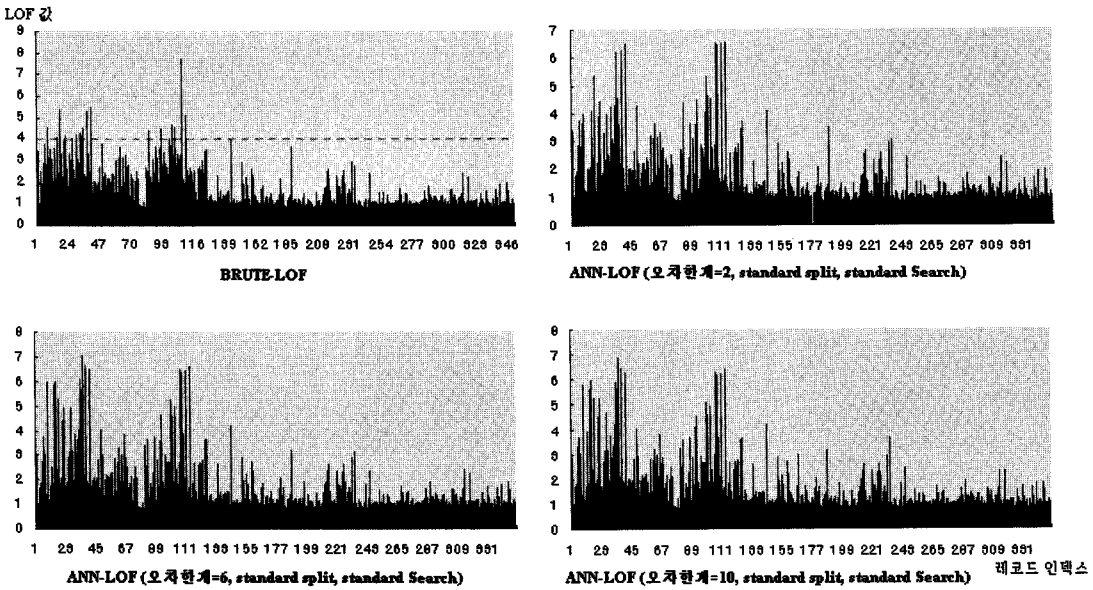
[그림 8] ANN 기법의 비교 실험

4.3 계산의 질 측면

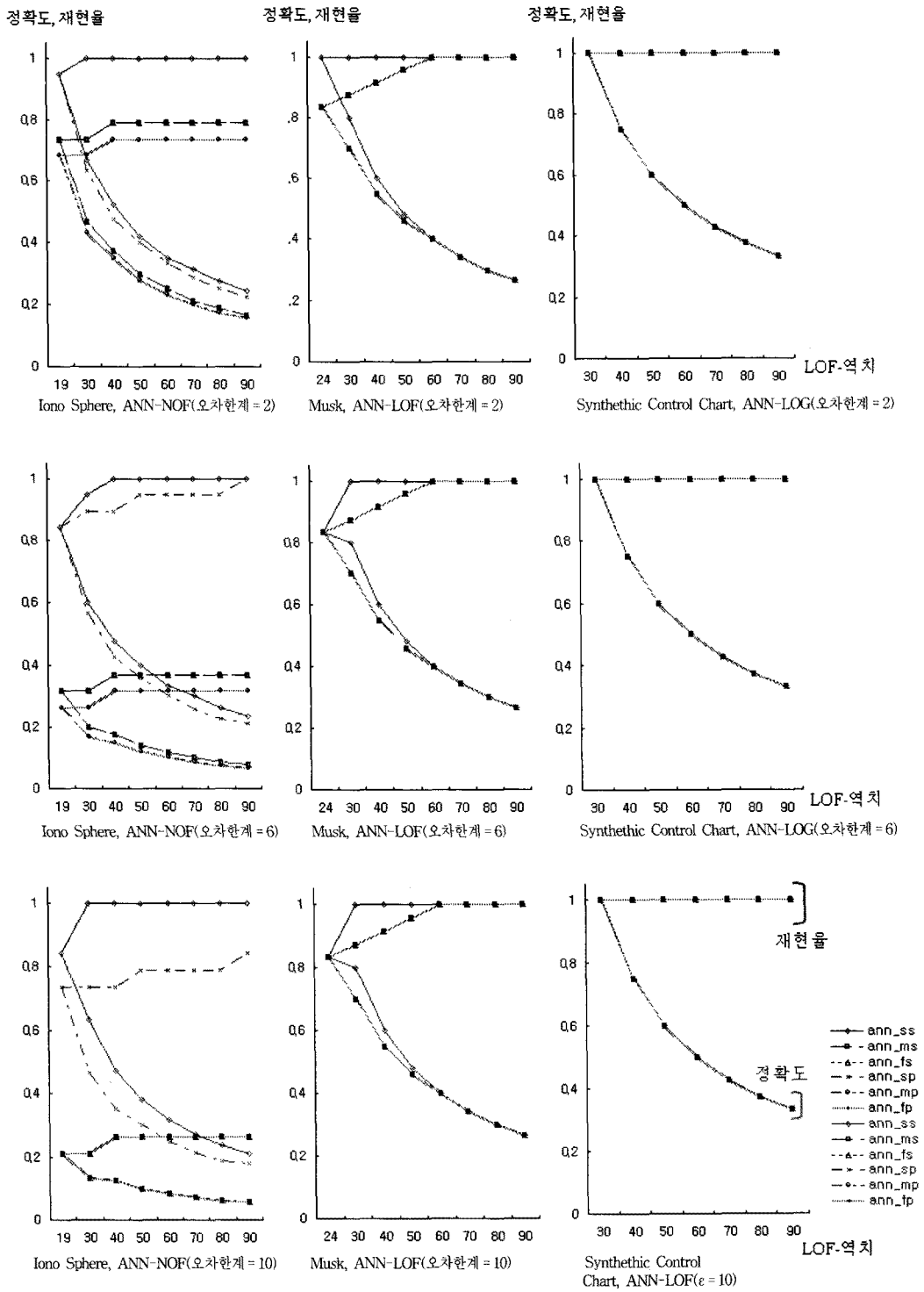
ANN을 이용하게 되면 LOF 값 자체가 추정 값이 되므로 오류가 발생할 수 있다. [그림 9]는 IonoSphere 데이터의 전체 351개 개체에 대한 LOF값을 나타낸 것이다. 상단 좌측의 히스토그램은 추정 오차가 개입되지 않는 데이터 개체들의 정확한 LOF 값을 단순 알고리즘(BRUTE-LOF)으로 계산한 결과이고, 상단 우측 및 하단의 두 히스토그램은 각각 오차한계가 2, 6, 10으로 변화할 때 그에 상응하여 변화되는 데이터 개체들의 LOF 값을 나타내었다. 그림에서 전반적으로 보이는 것과 같이 ANN을 사용한 LOF 계산은 개체의 인접이웃 계산에서 오차를 포함하므로 그 값이 정확한 LOF 값과 다르게 나타난다.

IonoSphere를 포함한 총 3개의 데이터 셋에 대해 추정오차가 개입되지 않는 정확한 LOF 값을 단순 알고리즘으로 계산한 후 각각의 데이터 셋 크기대비 5%의 개체 수만큼의 상위 LOF 값을 갖는 개체를 이상치로 보고 이 이상치들을 ANN으로도 얼마나 잘 찾아내는지를 정확도와 재현율을 통해 [그림 10]에서 나타내었다. IonoSphere 데이터 셋의 경우 전체 351개 개체 중 단순 알고리즘으로 계산했을 때의 LOF 값이 높은 순서대로의 상위 19개체를 이상치로 두었고 [그림 9]에서 실선으로 나타낸 LOF 값 4.0가 그 기준이다. 이외 Musk와 Synthetic Control Chart 데이터의 이상치는 단순 알고리즘의 계산결과인 상위 LOF 개체 24개, 30개를 각각 설정하고 오차한계와 역치 관점에서 ANN을 이용한 LOF 계산결과(ANN-LOF)의 해의 질을 비교하였다.

[그림 10]의 IonoSphere와 Musk의 예에서와 같이 오차한계가 증가함에 따라 해의 질은 떨어진다. 그러나 분석에 사용된 모든 데이터에서는 적절한 LOF-역치를 설정하여 긍정오류가 포함된 이상치로 예상되는 해의 수를 2배정도 증가시키면 실제의 이상치를 모두 검출 할 수 있었다. 물론 이러한 해의 질의 감소에도 불구하고 빠른 계산시간이 필수적으로 필요한 환경에서는 이러한 전략이 유용할 것이라 생각된다. 또한, 모든 데이터개체에 대해서 LOF를 계산할 것이 아니라, ANN을 이용한 LOF 값을 적절



[그림 9] ANN을 이용했을 때 LOF 값의 변화(IonoSphere)



[그림 10] ANN을 이용한 추정된 LOF의 오차한계 따른 해의 질 비교

한 역치를 통해서 여유 있게 검출한 후, 검출된 LOF에 대해서만 정확한 LOF 값을 재계산 하게 되면 계산시간과 계산의 질을 모두 만족할 수 있게 된다.

ANN에 있어 kd-tree의 분할방법은 standard split이 모든 데이터에 대해 가장 나은 성능을 보였고, 탐색전략 역시 표준탐색이 가장 우수한 결과를 도출했다. Synthetic Control Chart 데이터에 대해서는 오차한계 값에 무관하게 추정치가 정확한 LOF와 항상 일치하는 결과를 나타내었다.

5. 결 론

밀도기반의 이상치 탐지 기법중의 하나인 LOF 알고리즘은 여러 장점에도 불구하고 계산 부하로 인한 단점 때문에 이상치 탐지 분야에서 제한적으로 활용되어 왔다. 본 연구에서는 LOF 알고리즘을 대용량 데이터에 적용하기 위하여 계산시간을 줄일 수 있는 방법론을 제시하고 실제 데이터를 활용한 실험을 통해 유용성을 입증하였다. 본 연구에서 제시한 LOF 알고리즘은 향후 다양한 이상치 탐색 응용 분야로 적용이 가능할 것으로 기대된다.

본 연구에서 제안한 방법은 LOF 뿐만 아니라 밀도나 거리를 기반으로 하는 다른 이상치 탐지기법에도 적용가능하다. 추후 연구과제로는 LOF 계산의 질을 최대한 희생하지 않으면서 계산시간을 단축시키는 방법론에 대한 연구를 고려해 볼 수 있다. 예를 들면 의도적으로 긍정오류를 많이 포함하여 이상치로 추정되는 개체들을 탐지한 후 이 개체들에 대해서만 정확한 LOF 값을 계산한다면 계산의 질과 계산시간 양쪽 모두를 만족시킬 수 있는 이상치 탐지기법으로 사용될 수 있을 것으로 기대한다.

참 고 문 헌

[1] Aggarwal, C.C. and P.S. Yu, "Outlier detection for high dimensional data," *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, New York : ACM Press, (2001), pp.37-46.

[2] Agrawal, R., J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications," *Proceeding of the ACM SIGMOD International Conferenceon Management of Data, Seattle, Washington*, New York : ACM Press, (1998), pp.94-105.

[3] Agyemang, M. and C.I. Ezeife, "LSC-Mine: Algorithm for Mining Local Outliers," *Proceedings of the 15th Information Resource Management Association (IRMA) International Conference, New Orleans*, Hershey, PA U.S.A. : IRM press, Vol.1(2004), pp.5-8.

[4] Arya, S., D.M. Mount, N.S. Netanyahu, R. Silverman, and A. Wu, "An optimal algorithm for approximate nearest neighbor searching," *Journal of the ACM,* Vol.45, No.6 (1998), pp.891-923.

[5] Asuncion, A. and D.J. Newman, "UCI Machine Learning Repository," [http://www.ics.uci.edu/~mlearn/MLRepository.html], Irvine, CA : University of California, School of Information and Computer Science, 2007.

[6] Bentley, J.L., "K-d trees for semidynamic point sets," *Proceedings of 6th Annual ACM Symposium Computational Geometry*, New York : ACM Press, 1990, pp.187-197.

[7] Breunig, M.M., H.P. Kriegel, R.T. Ng, and J. Sander, "LOF : Identifying Density Based Local Outliers," *Proceedings of the ACM SIGMOD Conference, Dallas, Texas*, New York : ACM, (2000), pp.93-104.

[8] Ester, M.M., H.P. Kriegel, J. Sander and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases

with Noise," *Proceeding for 2nd International Conference on Knowledge Discovery and Data Mining(KDD '96), Portland, Oregon,* AAAI Press, (1996), pp.226-231.

[9] Ezawa, K.J. and S.W. Norton, "Constructing Bayesian Networks to predict Uncollectible Telecommunications Accounts," *IEEE Expert,* Vol.11, No.5(1996), pp.45-51.

[10] Faloutsos, C. and K.I. Lin, "FastMap : A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets," *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, California,* New York : ACM Press, Vol.24, No.2(1995), pp.163-174.

[11] Friedman, J.H., J.L. Bentley, and R.A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Transaction on Mathematical Software,* Vol.3, No.3 (1977), pp.209-226.

[12] Friedman, J.H., J.L. Bentley, and R.A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Transaction on Mathematical Software,* Vol.3, No.3 (1977), pp.209-226.

[13] Fukunaga, K., *Introduction to Statistical Pattern Recognition,* Academic Press, 2nd edition, 1977.

[14] Guha, S., R. Rastogi, and K. Shim, "Cure : An Efficient Clustering Algorithm for Large Databases," *Proceeding of the ACM SIGMOD International Conference on Management of Data, Seattle, Washington* New York : ACM Press, (1998), pp.73-84.

[15] Hinkley, D.V., "On the Ratio of Two Correlated Normal Random Variables," *Biometrika,* Vol.56, No.3(1969), pp.635-639.

[16] Hwang, S.S., S. Cho, and S. Park, "Keystroke dynamics-based authentication for mobile devices," *Computers and Security,* Vol.28 (2009), pp.85-93.

[17] Lewis, V.B., *Outliers in Statistical Data,* John Wiley and Sons, 1994.

[18] MacQueen, J., "Some Methods for Classification and Analysis of Multivariate Observations," *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, University of California, Berkeley,* Berkeley, Calififormia : University of California Press, (1967), pp.291-297.

[19] Medioni, G., I. Cohen, S. Hongeng, F. Bremond, and R. Nevatia, "Event Detection and Analysis from Video Streams," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol.8, No.23(2001), pp.873-889.

[20] Pokrajac, D., A. Lazarevic, and L.J. Latecki, "Incremental Local Outlier Detection for Data Streams," *IEEE Symposium on Computational Intelligence and Data Mining (CIDM), Honolulu, Hawaii,* New York : IEEE Press, (2007), pp.504-515.

[21] Press, W.H., B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, *Numerical Recipes in C,* Cambridge University Press, 1988.

[22] Ramaswamy, S., R. Rastogi, and K. Shim, Efficient algorithms for mining outliers from large data sets, *Proceedings of the International Conference on Management of Data, Dallas, Texas,* New York : ACM Press, (2000), pp.427-438.

[23] Strang, G., *Linear Algebra and its Applications,* Academic Press, 2nd edition, 1980.

[24] Yue, D., X. Wu, Y. Wang, Y. Li, and C.H. Chu, "A Review of Data Mining-Based Financial Fraud Detection Research," *Proce-*

edings of 2007 International Conference on Wireless Communications, Networking and Mobile Computing, Shanghai, P.R. China, Sponsored by IEEE, (2007), pp.5514-5517.

[25] Zhang, T., R. Ramakrishnan, and M. Livny, "Birch: An Efficient data clustering method for very large databases," *Proceedings for the ACM SIGMOD Conference on Management of Data, Montreal, Canada*, New York : ACM Press, (1996), pp.103-114.

[26] Zhang, J. and M. Zulkernine, "Anomaly Based Network Intrusion Detection with Unsupervised Outlier Detection," *Proceedings of 2006 IEEE International Conference on Communications, Istanbul, Turkey*, New York : IEEE Press, (2006), pp.2388-2393.

[27] http://www.cs.umd.edu/~mount/ANN/Files/1.1.1/ANNmanual_1.1.1.pdf.