

YONSEI UNIVERSITY, DEPARTMENT OF APPLIED STATISTICS

---

# Subspace Outlier Detection

---

12기 신보현

February 8, 2019

## 0. Reference Paper

Outlier Detection in Axis-Parallel Subspaces of High Dimensional Data by Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek

pdf [http://www.dbs.ifi.lmu.de/Publikationen/Papers/pakdd09\\_SOD.pdf](http://www.dbs.ifi.lmu.de/Publikationen/Papers/pakdd09_SOD.pdf)

## 1. Intuition

ABOD에 이어서 고차원 데이터에서 이상치를 찾는 기법을 소개한다. ABOD는 고차원 데이터에서 데이터들 간의 거리가 가지는 의미가 없어짐에 따라, 거리와 추가적으로 각도의 개념을 혼합하여 이상치를 분석하는 기법이었다. 하지만 이 때, 모든 feature를 사용하였다. 다시 말해, 고차원 데이터는 많은 feature들이 있을 것이고, 그 중 반응변수와 관련이 없는, noise한 feature들이 있을텐데, ABOD는 우선 이러한 feature들을 모두 사용한다. SOD는 이러한 방법에 의문을 던진다. 반응변수와 관련성이 떨어지는 feature를 포함하여 이상치를 탐지하려고 한다면 왜곡된 결과가 나올 수 있음을 먼저 지적한다.

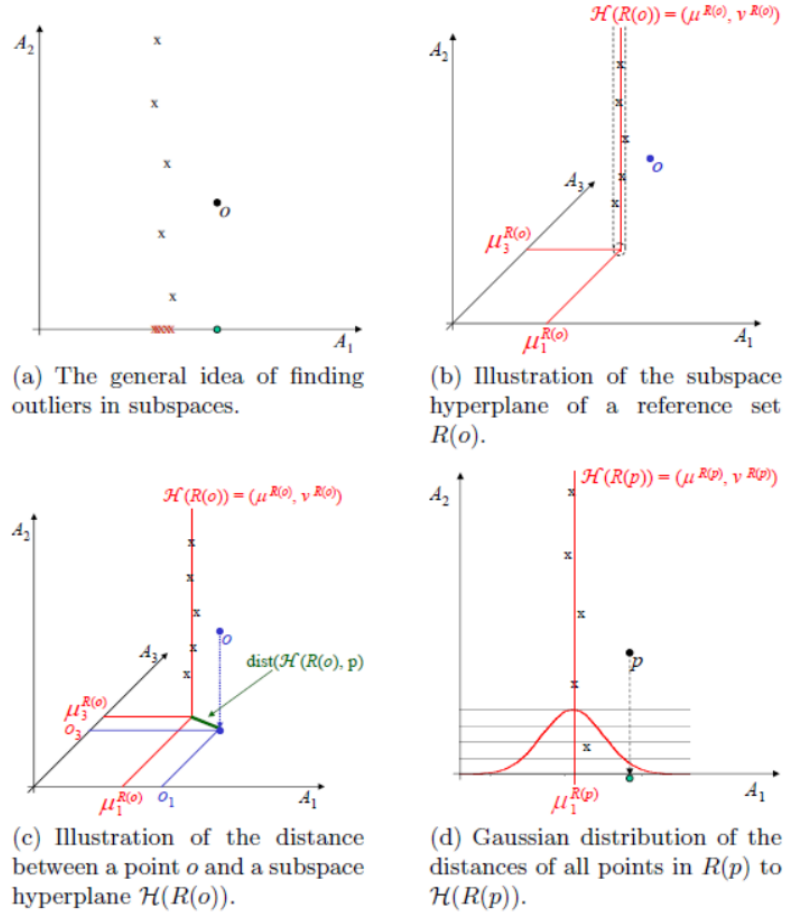


Figure 1

Figure 1의 (a)을 보자. full dimension으로 축  $A_1, A_2$ 를 모두 고려해보자. 2차원 평면에서  $o$ 라고 표시된 점은  $x$ 로 표시된 다른 점들과 크게 달라 보이지 않는다. 거리상으로, 밀도상으로 '이상치'라고 탐지할만큼의 특이한 메커니즘은 눈으로 봤을 때도 나타나지 않는 듯 하다. 바로 이러한 상황이 고차원 데이터에서 모든 차원을 고려하여

이상치를 탐지하려고 할 때 나타나는 문제점이다.

이제 축  $A_1$  만 고려해보자. 축  $A_1$  만을 기준으로 생각 한다면 x로 표시된 점들은 꽤 가까이 있고 o로 표시된 점은 이 점들과는 다소 멀리 떨어져 있음을 알 수 있다.

다음으로 축  $A_2$  만 고려해보자. 축  $A_2$  만을 기준으로 생각한다면 x로 표시된 점들이나 o로 표시된 점이나 다소 큰 차이가 없어 보인다. 즉 모든 점들이 uniform하게 퍼져있음을 알 수 있다.

축  $A_1$  과 축  $A_2$  는 어떤 차이점을 가질까? 축  $A_1$  만 고려를 하면 앞서 살펴보았듯이, 데이터들의 경향성을 파악할 수 있었다. 즉, 이 축은 반응변수와 연관되어 있는 변수라고 볼 수 있다. 그러나 축  $A_2$  만 고려를 하면 축  $A_1$  만 봤을 때 어떤 군집을 이루던 데이터들이 uniform하게 퍼져있게 보여, 마치 데이터들이 아무 상관이 없는 것처럼 보인다. 다시 말하면, 축  $A_2$  는 noise한 변수라고 볼 수 있다.

위의 논의를 종합해보자. Figure 1의 (a)의 간단한 예시에서, 데이터들의 메커니즘과 관련되어 있는 변수(축)을 기준으로 데이터를 살펴 보았을 때, 데이터들의 경향성을 파악하고 이상치를 탐지하기도 쉬웠다. SOD는 이러한 직관에서 출발한다.

## 2. Algorithm

SOD의 기본 아이디어는 각 점이 reference points에 의해 생성(span)된 부분 공간(subspace)에 얼마나 잘 맞는지를 분석하는 것이다. 어떤 점이 이상치인지 판별하기 위해 reference points를 정하고 그 reference points가 만드는(span) 부분공간은 noise 축에 평행한 초평면이다(axis-parallel hyperplane) 이렇게 설정된 초평면에 해당 데이터가 유의미하게 떨어져있으면 해당 데이터는 바로 그 초평면에 수직(perpendicular)인 부분공간에 대해서 이상치로 고려된다.

좀 더 자세하게 살펴보자. 우선 notation부터 정리를 한다. 이제부터 여러 notation이 정의될테니, 하나하나 꼼꼼하게 읽고 넘어가자. 그래야 이후 나올 개념들이 헛갈리지 않는다.

*suppose  $\mathcal{D} \subseteq \mathbb{R}^d$  is a database of  $n$  points in a  $d - \text{dimensional}$  feature space*

*and  $\text{dist}$  is a metric distance function on the points in  $\mathcal{D}$*

*$\mathcal{S}$  is the reference set*

*For any point  $p \in \mathbb{R}^d$ , we denote the projection of  $p$  onto attribute  $i$  by  $p_i$*

위에서 잠깐 언급했지만, reference set인  $\mathcal{S}$ 가 만드는(span하는) 부분공간을 noise 축에 평행하게 설정 한다(axis-parallel hyperplane) 이해를 위해 Figure 1의 (b)을 보자. 이후에 설명을 하겠지만, 여기서  $\mathcal{S}$ 가 만드는 부분공간은 빨강색 굵은 선이다. 자세히 보면 이것이 noise한  $A_2$  축과 평행함을 확인할 수 있다. 즉, 다른 축과 달리  $A_2$  축 기준으로 보면, 데이터들의 분산이 크다. 다시 말하면, 균등하게 퍼져있다. 따라서  $A_2$  축이 noise하다고 판단하고 이에 평행하게(parallel) 부분공간을 설정한 것이다. 이 논문의 제목 중 일부인 Axis-Parallel 이라는 말도 여기서 비롯된 것으로 보인다. 이렇게 noise 축에 대해서 평행하게 부분공간을 설정하면 다른 축( $A_1, A_3$ )과는 수직(perpendicular)인 것을 확인할 수 있다. Figure 1의 (b)에서  $A_1, A_3$  축은 의미가 있는 축이다. 다시 말해, 이 축에 대해서 데이터들의 분산은 작고 o라고 표시된 점만 동떨어져 있음을 확인할 수 있다. 여태까지의 논의를 정리하면, reference set인  $\mathcal{S}$ 가 만드는 부분공간은 어떻게든 설정할 수 있다.  $A_1, A_2, A_3$  축 중 어느 것과

평행하게, 수직하도록, 또는 특정한 각도를 이루도록 설정할 수 있는데, 이 논문에서는 특정하게 noise한 축과 평행한 부분공간을 생각한다. 그렇게 설정하면 자연스럽게 noise한 축과 수직인 축은 유의미한 축이 되기 때문이다. 따라서  $\mathcal{S}$ 의 부분공간인 초평면은 해당 초평면에 대해서  $\mathcal{S}$ 안에 있는 점들의 분산이 높고 그 초평면과 수직인 부분공간에 대해서는  $\mathcal{S}$ 안에 있는 점들의 분산이 낮은 특징을 가진다.  $\mathcal{S}$ 의 분산인  $VAR^{\mathcal{S}}$ 은 평균인  $\mu^{\mathcal{S}}$ (논문에서는 mean value라 나와고 있었음)에 대한  $\mathcal{S}$  점들의 평균 제곱 (average squared) 거리이다, ie.,  $VAR^{\mathcal{S}} = \frac{\sum_{p \in \mathcal{S}} dist(p, \mu^{\mathcal{S}})^2}{Card(\mathcal{S})}$ , 여기서  $Card(\mathcal{S})$ 은  $\mathcal{S}$ 의 점의 갯수를 의미한다. 유사하게, 어떤 attribute(feature, predictor, 변수라고 해석하면 됨)에 대한 분산은  $var_i^{\mathcal{S}} = \frac{\sum_{p \in \mathcal{S}} dist(p_i, \mu_i^{\mathcal{S}})^2}{Card(\mathcal{S})}$ 로 정의된다. 여기서,  $p_i$ 의 정의는 점  $p$ 를 attribute  $i$ 에 projection 시킨 점임을 다시 한번 상기하자. 다시 말하면, 어떤 attribute에 대한 분산은 데이터를 그 attribute(=축)으로 projection한 점들의 분산을 구한 것이다.

이제 점  $p$ 에 대한 reference points를  $R(p) \subseteq \mathcal{D}$ 로 정의하고 이를 reference set이라하자 (여태까지 계속 언급한  $\mathcal{S}$ 을 '점  $p$ '에 대한 reference set으로 다시 정의한 것일 뿐이다) 방금 정의한 reference set을 통해서  $p$ 의 outlieriness가 평가될 것이다.

또 다른 notation을 정의한다.  $v^{R(p)} \in \mathbb{R}^d$ 을 subspace defining vector라고 정의하고 이는  $R(p)$ 에 의해 정의된 부분공간의 relevant attributes를 보여준다. 즉, subspace defining vector인  $v^{R(p)}$ 은 어떤 축이 noise한 축이고 어떤 축이 유의미한 축인지 알려주는 벡터라고 생각하면 편하다. 여기서 의미있는 축, 즉 의미있는 attribute은 앞서 많이 언급했듯이, 그 축에대한 분산의 높고 낮음으로 판단한다. 그에 따라서 높은 분산과 낮은 분산의 기준이 필요할텐데, 이를 아래와 같이 정의한다.

모든  $d$ 개의 attributes(위에서 살펴보았듯이 feature라고 생각하면 편함)들 중에서 점들은 total variance인  $VAR^{R(p)}$ 을 가진다. 따라서  $i$ 번째 attribute가 가질 것으로 기대되는 분산은  $VAR^{R(p)}/d$ 이다.(어째서 갑자기 그 갯수만큼 나누는지 정확히 이해가 가지 않는 부분이다. 논문의 내용을 우선 그대로 옮겼다.) 만약  $var_i^{R(p)}$ 가 기대되는 분산( $VAR^{R(p)}/d$ )에  $\alpha$ (미리 정해진 계수)배를 한 것보다 작으면  $i$ 번째 attribute에 대한 점들의 분산이 낮다고 판단을 한다.  $R(p)$ 가 낮은 분산을 가지는 (방금 전 문장의 조건이 충족될 때) attribute에 대해서 해당 subspace defining vector인  $v_i^{R(p)}$ 를 1이라고 설정하고 나머지는 0이라고 설정한다. 이를 다시 표현하면 아래와 같다.

$$v_i^{R(p)} = \begin{cases} 1 & \text{if } var_i^{R(p)} < \alpha \frac{VAR^{R(p)}}{d} \\ 0 & \text{o.w} \end{cases} \rightarrow .$$

쉽게 말하면 subspace defining vector은 요소 값으로 유의미한 축에 대해서는 1, 무의미한 축에 대해서는 0의 값을 가진다.

reference points인  $R(p)$ 의 부분공간 초평면 (subspace hyperplane, 위에서 언급한 Axis-Parallel Subspace)인  $\mathcal{H}(R(p))$ 는  $R(p)$ 의 평균인  $\mu^{R(p)}$ 와 subspace defining vector인  $v^{R(p)}$ 로 정의된다, ie.  $\mathcal{H}(R(p)) = (\mu^{R(p)}, v^{R(p)})$   $\mu^{R(p)}, v^{R(p)}$ 로 정의된다는 말은 마치 정규분포를 평균과 분산으로 정의한다는 느낌으로 받아들이자.

여태까지 여러 개념을 정의했는데, Figure1의 (b)을 보며 정의한 개념을 정리해보자.  $R(o)$ 는 점  $o$ 에 대한 reference points이다. 또한  $\mathcal{H}(R(o))$ 은  $R(o)$ 의 subspace hyperplane이고 축  $A_2$ 에 대해서는 큰 분산을 가지지만 축  $A_1, A_3$ 에 대해서는 작은 분산을 가짐을 확인할 수 있다. 즉, 이를 통해 축  $A_2$ 는 irrelevant attribute이고 축  $A_1, A_3$ 은 relevant attribute임을 알 수 있다.  $R(o)$ 에 대한 subspace defining vector인  $v^{R(o)}$ 을 살펴보자. 방금 말했듯이, 축  $A_1, A_3$ 이 relevant하고 축  $A_2$ 는 그렇지 않으므로  $v^{R(o)} = (1, 0, 1)^T$ 임을 확인할 수 있다. 또한 정의에 의해  $\mathcal{H}(R(o)) = (\mu^{R(o)}, v^{R(o)})$ 이고 축  $A_1, A_3$ 와 수직인 빨강색 굵은 선으로 표시가 되어 있다.

이제 점  $o$ 가 subspace hyperplane인  $\mathcal{H}(R(o))$ 와 얼마나 떨어져있는지 측정할 수 있다. 떨어진 정도는  $o$ 과  $\mu^{R(o)}$ 의 거리를 subapce defining vector인  $v^{R(o)}$ 로 가중치를 한 weighted Euclidean distance으로 측정을 한다.

$$dist(o, \mathcal{H}(R(o))) = \sqrt{\sum_{i=1}^d v_i^{R(o)} \cdot (o_i, \mu_i^{R(o)})^2}$$

수식으로만 보면 피부로 와닿지 않을 것이다. Figure 1의 (c)에 있는 예시를 보자. 점  $o$ 와  $\mathcal{H}(R(o))$ 와의 거리가 초록색으로  $dist(\mathcal{H}(R(o)), p)$ 라고 표시되어 있다(여기서  $p$ 는  $o$ 와 동일하다)

축  $A_1$ 에 있는  $o_1$ 을 보자. 앞서 정의했듯이,  $p_i$ 는 점  $p$ 를  $i$ 번째 attribute에 projection한 점이다. 따라서  $o_1$ 은 점  $o$ 를 첫 번째 attribute, 여기서는  $A_1$ 로 projection한 점이다.  $o_3$ 도 마찬가지이다.  $\mu_1^{R(o)}, \mu_3^{R(o)}$ 도 동일하게 이해하자. 이제  $dist(\mathcal{H}(R(o)), p)$ 을 빗변으로 직각 삼각형에 주목 하자. 피타고라스 정리에 의해서  $dist(\mathcal{H}(R(o)), p)$ 의 제곱은 나머지 두 변의 제곱의 합이다. 방금 살펴본  $dist(\mathcal{H}(R(o)), p)$  값을 구하는 과정은 위의  $dist(\mathcal{H}(R(o)), p)$  정의와 동일하다. weight vector인  $v_i^{R(o)}$ 은 그 성분은 1 또는 0이고 이는 indicator value에 불과하다. 즉, irrelevant하다고 생각되는 attribute을 제외한 유의미한 attribute에 대해서 거리를 계산하라는 뜻으로, Figure 1의 (c)와 동일한 맥락이다.

여기서는  $dist(o, \mathcal{H}(R(o)))$ 을 이해하기 위해서 피타고라스 개념을 이용했다. 하지만 필자의 생각으로, 이는 Figure 1의 (c)의 저차원에서나 적용할 수 있을듯하다. 우선  $dist(o, \mathcal{H}(R(o)))$  공식을 저차원에서 이해했으니 고차원에서도 이와 비슷하게 작동할 것이라고 생각하자.

만약  $dist(\mathcal{H}(R(o)), p)$ 이 0에 가깝다면 어떠한 점  $o$ 가  $\mathcal{H}(R(o))$ 에 잘 맞는(원문에는 fits라는 표현)다는 의미이고 이는 곧 이상치가 아니라는 뜻이다. 반면에 높은  $dist(\mathcal{H}(R(o)), p)$ 은 이상치일 가능성이 높다는 뜻이다. 마지막으로 Subspace Outlier Degree(SOD)는 아래와 같이 정의된다.

$$SOD_{R(p)}(p) := \frac{dist(p, \mathcal{H}(R(o)))}{||v^{R(p)}||}$$

기존의 모델들과는 달리 SOD는 왜 점  $p$ 가 이상치인지를 알려준다. 점  $p$ 가 이상치라고 판단이 되면, 어떤 부분공간에서 점  $p$ 가 이상치인지를 알아낼 수 있다. 앞서 정의된 subspace defining vector인  $v^{R(p)}$ 는 reference points가 만들어낸 subspace hyperplane에 수직인 부분공간에 대한 정보를 담고 있다. 따라서  $v^{R(p)}$ 을 통해 이를 살펴봄으로써 점  $p$ 가 이상치라고 판명이 되는 부분공간을 살펴볼 수 있다.

이제 의미있는 reference set을 선택하는 과정에 대해서 살펴보자. 현재 존재하는 local (full dimensional) outlier detection 모델은 점  $p$ 의 local neighborhood을 살펴본다. 즉, 다시 말해 Euclidean distance를 기반으로 해서 가장 가까운  $k$ 개의 점들을 살펴본다. 하지만 서두에서 말했듯이, 고차원 데이터일 수록 차원의 저주로 인해서 점들 간의 거리가 가지는 의미가 없어진다.

SNN 접근법은 공통으로 가장 가까운 이웃의 갯수에 기반하여 점들의 유사도를 측정한다. SNN 접근법은 어떠한 점  $p$ 에 대해서 모든 점들이 거의 같은 거리에 있을지라도, 가장 가까운 이웃의 'Ranking'은 여전히 의미가 있다는 생각에서 출발한다.

동일한 메커니즘에 의해서 생성된 두 점,  $p$ 와  $q$ 를 생각해보자. 이 두점은 동일한 메커니즘에 의해서 생성되었기 때문에, 이들은 이웃이거나 동일한 메커니즘과 관련된 부분공간에서의 유사한 이웃을 가질 것이다. 여기에 불필요한 attributes을 추가하는 것은 절대적인 거리로 측정한 이웃 관계를 흐리게 할 것이다. 그럼에도, 가장 가까운

데이터는 크게 변하지 않을 것이다. 따라서 이들이 같은 메커니즘에 의해서 생성되었다면 점  $p$ 와  $q$ 는 공유하는 이웃들이 많을 것이다.

위의 아이디어를 정리해보자.

*Let  $N_k(p) \in \mathcal{D}$  be the  $k$  - nearest neighbors  $p \in \mathcal{D}$  w.r.t the distance function  $dist$*

*The shared nearest neighbor similarity between two points  $p, q \in \mathcal{D}$  is defined as*

$$sim_{SNN}(p, q) = Card(N_k(p) \cap N_k(q))$$

이제 reference set인  $R(p)$ 을  $sim_{SNN}$ 을 사용하여  $p$ 와 가장 가까운  $l$ 개의 점이라고 정의한다. 즉,

*the reference set  $R(p)$  of  $p$  is the set of  $l$  - nearest neighbors of  $p$  using  $sim_{SNN}$*

*in other words, a subset of  $\mathcal{D}$  that contains  $l$  points according to the following condition :*

$$\forall o \in R(p), \forall \hat{o} \in \mathcal{D} \setminus R(p) : sim_{SNN}(\hat{o}, p) \leq sim_{SNN}(o, p)$$

마지막 줄의 condition을 살펴보자. 이를 다시 말로 풀어보면 다음과 같다.

reference set에 있는 모든 점  $o$ 과  $\mathcal{D}$ 에서 reference set을 제외한 모든 점  $\hat{o}$ 에 대해서 점  $o$ 와 점  $p$ 의 SNN 값이 점  $\hat{o}$ 와 점  $p$ 의 SNN 값보다 크다는 뜻이다. 위에서 살펴보았듯이, 어떤 두점의 SNN 값이 크다는 것은, 각 점에서 가장 가까운  $k$ 개의 이웃을 뽑았을 때, 공통으로 있는 이웃의 수가 크다는 것이고, 이는 즉 동일한 메커니즘에서 나왔을 가능성이 큼을 의미한다. 여기서 주목해야할 점은, 점  $p$ 는 우리가 이상치인지 확인하고자 하는 점이라는 것이다. 즉 점  $o, p$ 의 SNN 값을 구한다는 것은, 이상치인지 확인하고자 하는 점과 이를 확인하기 위한 reference set의 점들간의 SNN를 구한다는 것이다. 위의 조건을 살펴보면 이상치인지 확인하고자 하는 점인  $p$ 와 그에 대한 reference set의 점 간의 SNN 값이 크다는 조건이고 이는 곧, SNN 정의에 따르면  $p$ 와 reference set의 점들이 동일한 메커니즘에서 나왔을 가능성이 크다는 뜻이다.

이상치를 판별하기 위해 그와 동일한 메커니즘에서 나올법한 점들을 뽑는다는 것이 얼핏 보면 이해가 가질 않는다 (SNN 값이 크다는 것은 두 점이 동일한 메커니즘에서 나왔다는 가능성이 크다는 말은 논문에서 직접적으로 언급되었다) 동일한 메커니즘에서 나왔을 법하다는 말을 두 점간의 '거리'가 가깝다는 뜻으로 받아들이면 조금은 쉽게 이해할 수 있다. 물론 SNN은 두 점간의 '거리'를 고차원 데이터에서 차원의 저주로 인해서 Euclidean distance로 구하지는 않고 SNN 값을 통해 '우회적'으로 구현한 것이다. 따라서 SNN 값이 크다는 것은 두 점이 동일한 메커니즘에서 나왔다, 즉 두점 간의 '거리'가 가까운 것이라고 할 수 있다. 이러한 생각을 가지면,  $sim_{SNN}(\hat{o}, p) \leq sim_{SNN}(o, p)$ 을 쉽게 이해할 수 있다. reference set의 점인  $o$ 와  $p$ 와의 SNN 값이 reference set이 아닌 점인  $\hat{o}$ 와  $p$ 와의 SNN 값보다 크다는 뜻이다. 즉, reference set의 점들을 SNN 값이 큰 점들로, 다시 말하면 점  $p$ 와의 '거리'(여기서는 SNN을 통한 우회적인 거리)가 가까운 점들로 설정한다는 뜻이다. 이를 다시 생각해보자. 어떤 점  $p$ 의 이상치 정도를 확인하고자 할 때, 우선 가장 가까운 점들과의 거리를 보는 것은 나름 합당한 판단 기준으로 보인다. 왜냐하면 점  $p$ 의 멀리 있는 점을 본다면, 실제로는 멀리 있는 것이 맞지만, 알고보니 그 중간에 많은 점들이 있을 수 있기 때문이다. 하지만 가장 가까운 거리점들을 본다면, 가장 가까운 점이 실제로는 멀리 있을 수도, 가까울 수도 있는데 가장 가깝다는 말로 인해서 그 중간에 다른 점들이 없다. 이러한 이유로 인해서 reference

set을 정할 때, 가장 가까운 거리에 있는 점들을 설정하고, 이때, 고차원의 저주를 고려하여 Euclidean Distance 으로 거리를 계산하지 않고 SNN을 이용하여 우회적으로 거리의 개념을 표현하는 것이다.

SOD 모델은 두 parameters에 의존한다. 먼저 reference set을 결정하는데 사용되는  $sim_{SNN}$ 에서의  $k$ 이다. 이는 그렇게 중요한 모수는 아니며 점들간의 공통의 메카니즘을 알아내기 위해 적당히 크게 설정하면 된다.

둘 째로 reference set의 크기인  $l$ 이다. 위와 동일한 이유로 너무 작게 설정하지 않으면 된다. 당연히,  $l$ 은  $k$ 보다 작거나 같게 설정되어야 할 것이다.

셋 째로 subspace defining vector에서 attribute의 중요도에대해서 결정하기 위한 threshold인  $\alpha$  또한 정해야 한다. 이 논문에서는  $\alpha = 0.8$ 로 설정했으니 이와 같이 설정하라고 하고 구현된 R 코드에서도 0.8이 기본값이기도 하다.

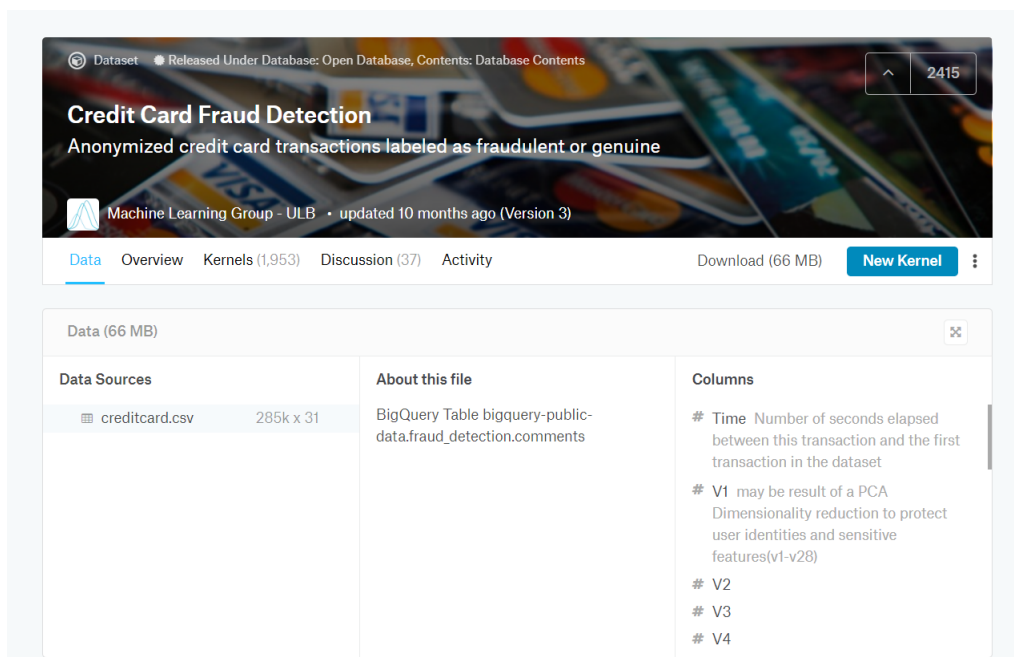
### 3. Codes in python and R

python → 찾지 못했음.

R → <https://cran.r-project.org/web/packages/HighDimOut/HighDimOut.pdf>

여기에는 저번에 발제한 ABOD 기법도 있다. 고차원 데이터에서 이상치 발굴하기에 특화된 패키지!!!

### 4. Kaggle data



The screenshot shows the Kaggle dataset page for 'Credit Card Fraud Detection'. The dataset is described as 'Anonymized credit card transactions labeled as fraudulent or genuine' and is from the 'Machine Learning Group - ULB', updated 10 months ago (Version 3). The page includes tabs for Data, Overview, Kernels (1,953), Discussion (37), and Activity. The 'Data' tab is selected, showing a table with the following columns: Data Sources, About this file, and Columns. The 'Data Sources' column lists 'creditcard.csv' with a size of 285k x 31. The 'About this file' column states it is a BigQuery Table from 'bigquery-public-data.fraud\_detection.comments'. The 'Columns' column lists: '# Time: Number of seconds elapsed between this transaction and the first transaction in the dataset', '# V1: may be result of a PCA Dimensionality reduction to protect user identities and sensitive features(v1-v28)', '# V2', '# V3', and '# V4'.

```
data = read.csv("C:/Users/sbh0613/Desktop/와이빅타/이상치분석/creditcardfraud/creditcard.csv", header=T)
library(HighDimOut)

## Warning: package 'HighDimOut' was built under R version 3.5.2

table(data$class)
```

---

```
##
##      0      1
## 284315    492

which(data$Class == 1)[1:10]

## [1] 542 624 4921 6109 6330 6332 6335 6337 6339 6428

which(names(data) == 'Class')

## [1] 31

start_time = Sys.time()
sod.result = Func.SOD(data[500:550,-31],k.nn=5,k.sel=5,alpha=0.8)

## Warning: executing %dopar% sequentially: no parallel backend registered

end_time = Sys.time()
#running time
end_time - start_time

## Time difference of 4.865951 secs

sod.result.mat = cbind(sod.result,500:550)
sod.result.mat = sod.result.mat[order(sod.result,decreasing=TRUE),]
sod.result.mat[1:10,]

##      sod.result
## [1,] 0.5835459 536
## [2,] 0.5772464 514
## [3,] 0.4201774 545
## [4,] 0.3554055 520
## [5,] 0.3246576 542
## [6,] 0.3074953 530
## [7,] 0.2728791 522
## [8,] 0.2250221 534
## [9,] 0.2239918 505
## [10,] 0.2145887 511
```



꽤나 높은 등수!! parameter을 바꿔보자.

```
sod.result = Func.SOD(data[500:550,-31],k.nn=3,k.sel=5,alpha=0.8)
sod.result.mat = cbind(sod.result,500:550)
sod.result.mat = sod.result.mat[order(sod.result,decreasing=TRUE),]
sod.result.mat[1:10,]
```

```
##      sod.result
## [1,] 0.9153295 536
## [2,] 0.4122684 545
## [3,] 0.3211322 542
## [4,] 0.3000382 530
## [5,] 0.2390452 514
## [6,] 0.2239918 505
## [7,] 0.2179487 522
## [8,] 0.2166830 531
## [9,] 0.2145887 511
## [10,] 0.2137710 534
```

이제 모든 행에 해보자. 하지만 에러..

```
Error in do.call("External", c(list(CFUN, x, y, pairwise, if (!is.function(method)) get(method) else
negative length vectors are not allowed
```

구글링 → <https://stackoverflow.com/questions/42479854/merge-error-negative-length-vectors-are-not-allowed>

요약: 계산이 너무 많아서 R이 감당하기 힘들.

총 28000개 정도의 행이 있는데 이 중 만개의 행을 해보자.

```
sod.result.mat = cbind(sod.result,1:10000)
sod.result.mat = sod.result.mat[order(sod.result,decreasing=TRUE),]
colnames(sod.result.mat) = c("value","index")
head(sod.result.mat)
fraud.index = which(data$Class == 1)[ which(data$Class == 1) <= 10000 ]
rank.vec = c()
for (i in 1:length(fraud.index)){
  for (j in 1:length(sod.result.mat[, "index"] )){
    if (fraud.index[i] == sod.result.mat[, "index"][j]) rank.vec[i] = j
```

```
}  
}  
cbind(fraud.index,rank.vec)
```

```
> start_time = Sys.time()  
> sod.result = Func.SOD(data[1:10000,-31],k.nn=3,k.sel=5,alpha=0.8)  
> end_time = Sys.time()  
> end_time - start_time  
Time difference of 24.96106 mins
```

24분이 걸렸다! ABOD는 몇 시간을 뒤도 결과가 안 나왔는데, SOD가 계산량이 적음을 알 수 있었다.

	value	index
[1,]	380.386259	8136
[2,]	23.560617	8872
[3,]	22.100107	5295
[4,]	17.246510	5975
[5,]	15.778148	7528
[6,]	15.203974	9153
[7,]	14.330490	8960
[8,]	11.747445	5938
[9,]	10.077419	9120
[10,]	9.403858	8873

Result 1

위의 결과는 1~10000번째까지 행을 뽑아서 SOD를 연산한 뒤에, SOD 값이 큰 순서부터 (이상치일 가능성이 가장 큰 순서부터) 나열한 결과다 (상위 10개의 행) value는 SOD 값이고 index는 data에서의 행 index이다.

이제 실제로 신용카드 사기를 친 행이 결과 1에서 몇 번째 행에 있었는지, 즉 이상치 순위에서 몇 등을 했는지 알아보자. 아래는 그 결과다.

```
> cbind(fraud.index,rank.vec)
      fraud.index rank.vec
[1,]          542    1298
[2,]          624    2154
[3,]         4921    1232
[4,]         6109     754
[5,]         6330     898
[6,]         6332     338
[7,]         6335     392
[8,]         6337    4618
[9,]         6339    1887
[10,]        6428     165
[11,]        6447     957
[12,]        6473     261
[13,]        6530     160
[14,]        6610     996
[15,]        6642    1170
[16,]        6718     164
[17,]        6720     163
[18,]        6735     905
[19,]        6775     171
[20,]        6821     914
[21,]        6871     903
[22,]        6883     757
[23,]        6900     589
[24,]        6904    1268
[25,]        6972    1009
[26,]        8297     134
[27,]        8313    1413
[28,]        8336     123
[29,]        8616     126
[30,]        8618    2144
[31,]        8843     753
[32,]        8846     742
[33,]        8973     116
[34,]        9036     132
[35,]        9180     150
[36,]        9253     162
[37,]        9488     209
[38,]        9510     188
> |
```

Result 2

결과 2에서 fraud.index는 데이터에서 실제로 신용카드사기를 한 행의 index이고 rank.vec은 결과 1로부터 도출한 이상치 순위이다. 즉, 결과 2에서 542번째 신용카드사기 행은 만 개의 데이터에서 SOD를 돌리고 내림차순으로 정렬한 결과 그 값이 1298번째로 컸다는 의미이다. 데이터가 총 10000개가 있으며 1등부터 10000등까지 있다고 생각할 때, 위의 결과 2는 나름 결과가 잘 나온듯 하다. 몇몇 네 자리수를 제외하면 대부분 세 자리수이며 500등 이내에 진입한 행도 많이 보인다.

Func.SNN은 input으로 Func.SOD와 동일하게 k.nn, k.sel이 필요하다. k.nn은 shared nearest neighbors을 계산하기 위해 필요한 값이고 k.sel은 shared nearest neighbors의 수이다. 즉, SOD에서의 reference set의 수라고 보면 된다. 이 함수를 실행하면 각 관측치에 대해서 상위 k개의 shared nearest neighbors을 알려준다.

```
#SNN
start_time = Sys.time()
snn.result = Func.SNN(data[1:10000,-31],k.nn=3,k.sel=10)
end_time = Sys.time()
snn.result
```

---

```

x1  1  2  3  4  5  6  7  8  9 10
1  9  4 13 21 57 78 86 91 104 1
2  6  7 10 11 12 18  1  2  3  4
3 21 86 135 148 236 250 258  1  2  3
4  1  5  9 13 57  2  3  4  6  7
5  4  8 13 15 19  1  2  3  5  6
6  2  7 10 11 12 18  1  3  4  5
> dim(snn.result)
[1] 10000  10
> start_time - end_time
Time difference of -16.16536 mins
>

```

정리해보면 Func.SOD 함수 안에 이미 Func.SNN 함수를 통해 SNN 값을 계산 한 후에 마지막으로 SOD까지 나오는 것이다. 그렇다면 왜 따로 Func.SNN이 있을까? 이상치 정도가 높은 관측치에 대한 reference set을 살펴봄으로써, 이 부분공간에 대해서 이상치로 판별되었다고 알 수 있기 때문인듯 하다.