Yonsei University, Department of Applied Statistics

Outlier Analyis - ABOD

12기 신보현

January 25, 2019

0. Introduction

이상치란? 간단하게, 다른 데이터와 비교할 때, 다른 특징을 가지고 있는 데이터.(그 데이터의 일반적인 분포와 맞지 않은 데이터)
이상치 탐지의 중요성 : 이상치를 우연히 일어났다고 간주하는 것보다는, 그 이상치가 다른 mechanism에서 왔다고 보고 이에 주목을 하는 것이 좋다.
ex) credit card abuse in financial transcations data, identification of measurement errors in scientific data...
위와 같은 필드에서, 이상치 데이터를 탐지하여 그에 맞는 대응책을 만드는 것이 효과적일 것이다.
위의 논의를 종합했을 때, 이상치를 멋진 말로 정의하면 an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism.

그렇다면, 다른 메카니즘에서 나왔다는 기준을 어떻게 설정할까? 이 기준을 설정하는 방향에 따라서, 이상치를 탐지하는 기법도 달라진다.

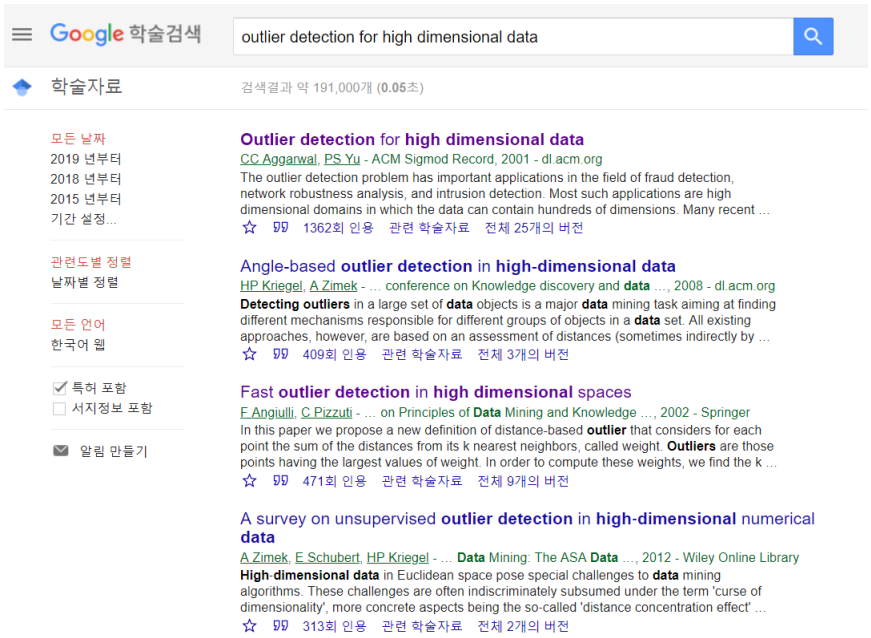
가장 전통적인 기준은 데이터간의 거리를 측정하는 것이다. 데이터간의 거리 측정의 개념은 곧 클러스터링의 개념과 일맥상통한다. 따라서 기존의 이상치 탐색 과정은 KNN, 클러스터링 기법과 연관된 것이 많은 것 같았다.(구글링 결과..)

하지만 이는 분명한 단점을 가지고 있는데, feature가 많아지는 high dimensional data(고차원 데이터)에서 성능이 떨어진다는 것이다. 즉, 다시 말해 차원이 높아질수록 curse of dimensionality(차원의 저주)로 인해서 데이터들 간의 거리가 가지는 의미가 점점 사라진다.

$$\lim_{d\to\infty} \frac{dist_{max} - dist_{min}}{dist_{min}} \longrightarrow 0$$

이를 다른 말로 표현하면, 차원이 높아지면서 모든 점들이 비슷한 거리에 위치하고 있고 이러한 현상을 data sparsity 또는 distance concentration이라고 부른다.
이렇게 차원이 높아질수록 거리가 가지는 의미가 사라지므로 데이터들간의 거리에 기반한 이상치 탐색은 그 성능이 떨어질 수밖에 없을 것이다.

1. Angle-Basd Outlier Dection in High-dimensional Data(ABOD)



나름 유명한 듯.. 409회 인용..! 1362회 인용은 다음에..(너무 어렵다..ㅠㅠㅠ)

Reference: https://imada.sdu.dk/~zimek/publications/KDD2008/KDD08-ABOD.pdf

1-1 Intuition.

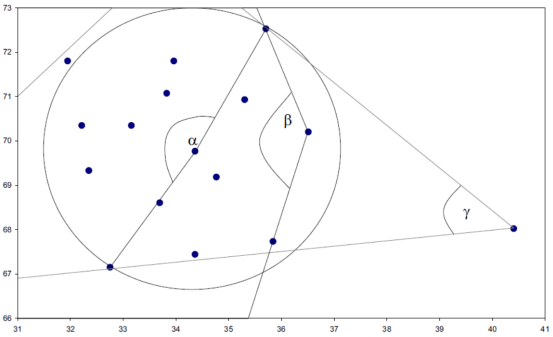


Figure 1

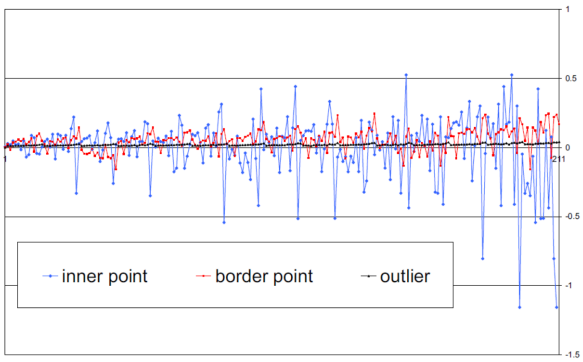


Figure 2

Figure 1을 보면, 각도 α, β, γ가 표시되어 있다. 이상치로 의심이 되는 각 γ가 상대적으로 작음을 확인할 수 있다. 이상치로 의심이 되는 오른쪽 데이터을 기준으로 다른 두 데이터를 정한 후, 세점을 이어 만든 각도들은 그 값의 변동 폭이 작다. 원 안에 있는 점 하나를 정한 후, 앞의 과정을 동일하게 진행하면, 최대값과 최소값의 차이가 상대적으로 더 크다. ABOD는 이러한 intuition에서 만들어졌다.

1-2 Formaula.

하나의 기준 점과 다른 두 점을 잡아서, scalar product(내적)을 계산하고 이를 벡터의 길이 제곱으로 normalize 한다.(벡터의 길이 제곱으로 나눠준다.) 벡터의 길이 제곱의 역수를 곱해주는 것은 어떤 weighting factor을 곱해주는 과정으로 볼 수 있고, 길이가 길면 작은 역수 값이 곱해지는 것으로, 길이가 작으면 큰 역수 값이 곱해지는 것이다. 이는 기존의 이상치 탐지를 할 때 사용되는 거리 개념을 ABOF에 추가한 것으로, 거리와 각도로 ABOF 공식을 정의한다.

정리하면,

두 벡터의 내적 $\longrightarrow$ 두 벡터 사이의 각도를 반영

두 벡터의 제곱합으로 normalize(나눠주기) $\longrightarrow$ 두 벡터의 길이 반영

$\longrightarrow$ 이들의 분산으로 ABOF를 정의함.

<논문에서의 정의>: The angle-based outlier factor ABOF($\overrightarrow{A}$) is the variance over the angles between the difference vectors of $\overrightarrow{A}$ to all pairs of points in D weighted by the distance of the points:

위의 과정을 수식으로 표현하면 아래와 같다.

$$D: \textit{given database},\ \overrightarrow{A}, \overrightarrow{B}, \overrightarrow{C} \in D$$

$$\overline{BC} = \overrightarrow{C} - \overrightarrow{B}$$

$$\begin{aligned} ABOF(\overrightarrow{A}) &= VAR_{\overrightarrow{B},\overrightarrow{C}\in D}\left(\frac{<\overline{AB},\overline{AC}>}{||\overline{AB}||^2 \cdot ||\overline{AC}||^2}\right) \\ &= \frac{\sum_{\overrightarrow{B}\in D}\sum_{\overrightarrow{C}\in D}\left(\frac{1}{||\overline{AB}||\cdot||\overline{AC}||}\cdot\frac{<\overline{AB},\overline{AC}>}{||\overline{AB}||^2\cdot||\overline{AC}||^2}\right)^2}{\sum_{\overrightarrow{B}\in D}\sum_{\overrightarrow{C}\in D}\left(\frac{1}{||\overline{AB}||\cdot||\overline{AC}||}\right)} \\ &- \left(\frac{\sum_{\overrightarrow{B}\in D}\sum_{\overrightarrow{C}\in D}\left(\frac{1}{||\overline{AB}||\cdot||\overline{AC}||}\cdot\frac{<\overline{AB},\overline{AC}>}{||\overline{AB}||^2\cdot||\overline{AC}||^2}\right)}{\sum_{\overrightarrow{B}\in D}\sum_{\overrightarrow{C}\in D}\left(\frac{1}{||\overline{AB}||\cdot||\overline{AC}||}\right)}\right)^2 \end{aligned}$$

1-3 Advantage / Disadvantage.

대부분의 이상치 탐지 모델은 결과에 아주 중요한 역할을 하는 parameter를 사용자가 직접 결정해야한다. 이상치 탐지는 어느 데이터가 이상치인지 애초에 알지 못하므로 비지도 학습의 일부로 볼 수 있는데 이러한 비지도학습에서 parameter 설정은 크게 어려울 수 있다. ABOD는 이러한 parameter 설정을 하지 않는 다는 것이 큰 장점이다.

하지만 어떤 데이터의 ABOD 값을 구하기 위해 다른 모든 점에 대한 ABOD를 계산하기 때문에 계산 시간이 오래걸린다는 단점이 있다.

1-4 Others.

- Speed-up by Approximation (FastABOD): ABOD은 모든 데이터 짝에 대해서 ABOD 값을 계산하기 때문에 시간이 오래 걸린다는 단점이 있다. 이러한 단점을 극복하기 위해, approximation algorithm인 FastABOD가 개발되었다.
  아이디어는 간단하다. 모든 데이터에 대해 ABOD 값을 계산 하는 것이 아니라, 가장 가까운 k개의 데이터에 대한 ABOD 값만을 계산한다. 하지만 k라는 parameter을 설정해야하는 단점 때문에, 어떤 k를 선택하느냐에 따라서 성능이 좌지우지 된다. 즉, parameter을 설정하지 않아도 된다는 장점이 사라지는 것이다.

- Approximation as Filter-Refinement Approach(LB-ABOD): ABOD의 계산상의 문제와 k에 따라서 성능이 좌지우지되는 FastABOD의 단점을 해결한 방법. (이해 못했..)

- Generalization for Arbitrary Data Types - Kernel-based ABOF

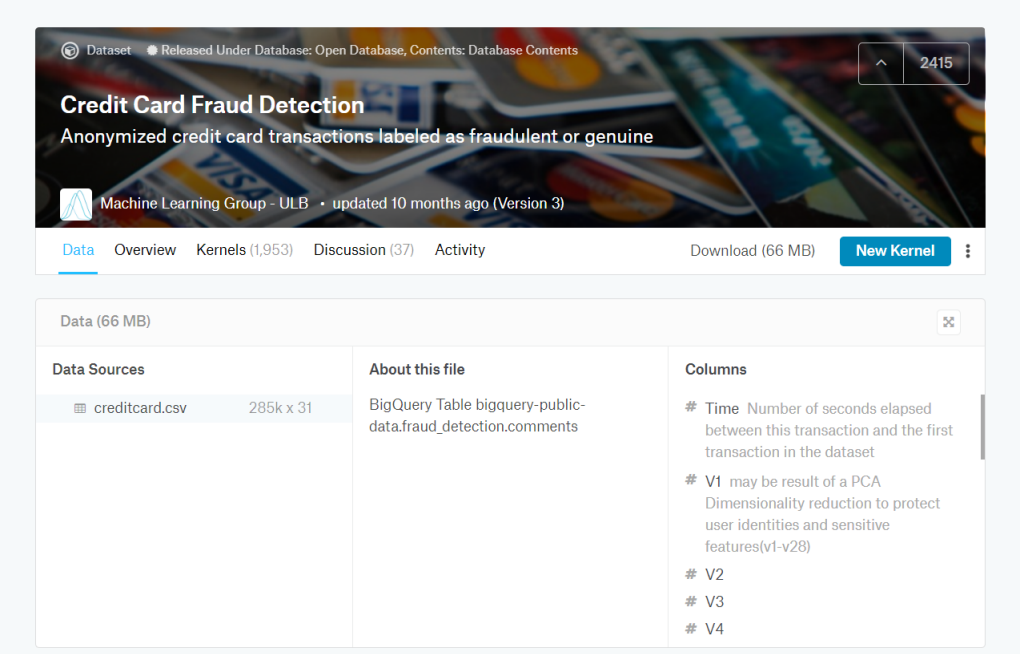
1-5 Usage.
R에서의 사용은 https://cran.r-project.org/web/packages/abodOutlier/abodOutlier.pdf은 참조하면 된다. (ABOD, Fast ABOD)

python에서의 사용은 https://pyod.readthedocs.io/en/latest/pyod.models.html#module-pyod.models.abod을 참조하면 된다. (ABOD, Fast ABOD)

또는 https://github.com/MarinYoung4596/OutlierDetection/blob/master/OutlierDetection/Python%20Implementation/abod.py에 있는 함수를 참조하면 된다. (ABOD, FastABOD, LB-ABOD)

2. kaggle 데이터에 적용해보기

outlier가 있는 데이터를 찾아보려했지만... 힘들어서 매우 불균형 데이터인 캐글 데이터로 해보았다.




```
data = read.csv("C:/Users/sbh0613/Desktop/와이빅타/이상치분석/creditcardfraud/creditcard.csv")
dim(data)
```

```
## [1] 284807     31
```

```
table(data$Class)
```

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

```
table(data$Class)[2]/(table(data$Class)[1] + table(data$Class)[2])
```

```
##           1
## 0.001727486
```

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

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

```
library(abodOutlier)
```

```
## Warning: package 'abodOutlier' was built under R version 3.5.2
```

```
start_time <- Sys.time()
abod.result = abod(data[500:550,],method="complete")
```

```
## Done:  10 / 51
## Done:  20 / 51
## Done:  30 / 51
## Done:  40 / 51
## Done:  50 / 51
```

```
end_time <- Sys.time()
#running time
end_time - start_time
```

```
## Time difference of 1.008519 mins
```

```
abod.result[1:5]
```

```
## [1] 0.1061683 0.1076610 0.1059350 0.4033311 0.1517701
```

```
abod.mat = cbind(abod.result,500:550)
head(abod.mat[order(abod.result),])
```

```
##       abod.result
## [1,] 1.209437e-07 514
## [2,] 1.021846e-01 542
## [3,] 1.059350e-01 502
## [4,] 1.061683e-01 500
## [5,] 1.076610e-01 501
## [6,] 1.127711e-01 550
```