The COPOD Algorithm and its Applications under Multiple Scenarios

BAI,Qian 20786466, TONG,Jinjian 20788658, YUAN,Dian 20785278,
WANG,Weihong 20789341, and YU,Ximeng 20803628

GROUP 02
Type of this project: Implementation
This project is done solely within the course but not other scopes.

1 Introduction

In data science, outliers refer to the data points which deviate dramatically from other normal observations. It might be bad data from human input error, or it might be important data which is worth studying. In either way, the existence of outliers could have a great impact on the model training. For example, outliers can importantly affect the choice of cluster center. To avoid its influence on our insight to the observation, it is important to detect the outliers, and study it when it is natural, or remove it when we need to preserve the integrity of the model.

Over the last decades, different kinds of outlier detection algorithm are proposed, including covariance based model (i.e. MCD), proximity based model (i.e. KNN), linear model (i.e, SVM), ensemble base model (i.e. LSCP), and neural network based models (i.e. SO-GAAL and MO-GALL). All these models have different drawbacks.

First, they all suffer from the curse of dimensionality, and their performance drops and the response time climbs as the dimensionality of data increases. Second, many of those models need extra amounts of work, including hyper-parameter selection, tuning, or choosing layers of classification. Not to mention that different choices of hyperparameters can lead to different outcomes and have different performance. Therefore, it is important to develop a new algorithm that both has great performance, even with high dimension data, and does not need a heavy amount of work.

The author of the article [2] proposes a method called COPOD which stands for **Cop**ula-Based **O**utlier **D**etection.

COPOD does not need to calculate the distance between samples, so the running overhead is small and the speed is faster, and it can be easier to expand large data sets. COPOD can easily process data sets with 10,000 features and 1,000,000 observations on a standard personal laptop. Secondly, because COPOD is based on the empirical cumulative distribution function (ECDF), it does not need to be adjusted and can be called directly, which avoids the challenges of hyperparameter selection and potential deviations. The third is the effect of COPOD, which ranks the highest in comparison with other 9 mainstream

algorithms (such as isolated forest, LOF) on more than 30 data sets. Its ROC-AUC score is 1.5% higher, and the average accuracy is 2.7% higher than the second-best detector. In addition, COPOD can provide some interpretability for which dimensions caused the anomaly, and quantify the anomaly contribution of each dimension through the dimension outline map. In this way, we can directly find the dimensions that cause the most anomalies for in-depth analysis. This allows practitioners to focus on certain subspaces to better improve data quality.

2 Algorithm

2.1 Inverse Sampling

General programming languages have (pseudo) random number generation functions, such as python's *random.random*() function, but they usually only correspond to uniform distribution. So if we want to generate random numbers that conform to any probability distribution, we need to use "inverse transform sampling".

Theorem 1. *Let X be a continuous random variable, its cumulative distribution function is $CDF(x) = P(X \leq x)$, if the random variable $Y = CDF(X)$, then Y follows a uniform distribution on the interval [0,1].*

Proof. Claim: $CDF(x) \in [0, 1]$, and $CDF(x)$ is a monotonic increasing function.

$$\begin{aligned}
\forall y < 0, \qquad & P(Y \leq y) = P(CDF(X)) \leq y) = 0 \\
\forall 0 \leq y \leq 1, \qquad & P(Y \leq y) = P(CDF(X)) \leq y) \\
& \qquad = P(X \leq CDF^{-1}(y)) \\
& \qquad = P(X \leq x) \\
& \qquad = CDF(x) \\
& \qquad = y \\
\forall y > 1, \qquad & P(Y \leq y) = P(CDF(X)) \leq y) = 1
\end{aligned} \tag{1}$$

Hence, Y follows a uniform distribution on the interval [0,1].

This method can be used to generate a random sample from the distribution when the c.d.f. of any probability distribution is known.

2.2 Copula Description

Sklar[5] believes that the joint distribution of N random variables can be decomposed into their respective marginal distributions and a Copula function to separate the randomness and coupling of the variables. Among them, the randomness of each random variable is described by the marginal distribution, and the coupling characteristics between random variables are described by

the Copula function. In other words: copula allows us to calculate the joint distribution between variables through the marginal distribution of variables.

In sklar theory, if $H(x, y)$ is a binary joint distribution function of $F(x)$ and $G(y)$ with continuous marginal distribution, then there is a unique Copula function such that $H(x, y) = C(F(x) + G(y))$

If H, F, G are known

$$C(u, v) = H(F^{-1}(u), G^{-1}(v)) \tag{2}$$

Here $F^{-1}(u)$ represents the inverse function of $F(u)$, or the inverse cumulative distribution function.

Different distribution functions are used to characterize the data characteristics of different variables, and then a connection function (Copula) is used to "connect" the cumulative distribution functions of the two variables. In this way, the joint distribution characteristics between the variables are finally described. Furthermore, the multivariate Copula function can be constructed by generalizing the two-dimensional Copula function to multiple variables.

2.3 COPOD algorithm

Algorithm 1 Copula Outlier Detector

Inputs: input data X
Outputs: outlier scores $O(X)$

for each dimension d **do**
  Compute left tail ECDFs: $\hat{F}_d(x) = \frac{1}{n}\sum_1^n \mathbb{I}(X_i \leq x)$
  Compute right tail ECDFs: $\hat{\hat{F}}_d(x) = \frac{1}{n}\sum_1^n \mathbb{I}(-X_i \leq -x)$
  Compute the skewness coefficient according to Equation 11.
end for
for each i in $1, \ldots n$ **do**
  Compute empirical copula observations
    $\hat{U}_{d,i} = \hat{F}_d(x_i)$,
    $\hat{V}_{d,i} = \hat{\hat{F}}_d(x_i)$
    $\hat{W}_{d,i} = \hat{U}_{d,i}$ if $b_d < 0$ otherwise $\hat{V}_{d,i}$
  Calculate tail probabilities of X_i as follows:
    $p_l = -\sum_{j=1}^d \log(\hat{U}_{j,i})$
    $p_r = -\sum_{j=1}^d \log(\hat{V}_{j,i})$
    $p_s = -\sum_{j=1}^d \log(\hat{W}_{j,i})$
  Outlier Score $O(x_i) = \max\{p_l,\ p_r,\ p_s\}$
end for
Return $O(X) = [O(x_1), \ldots, O(x_d)]^T$

Fig. 1: COPOD Algorithm

Since copula allows us to describe the joint distribution of using only their marginals. This gives a lot of flexibility when modelling high dimensional datasets, as we can model each dimension separately, and there is a guaranteed way

to link the marginal distributions together to form the joint distribution. In COPOD, researches assume the seperate dimension is uncorrelated and take sum of negative log of each dimension as outlier score of a data record, which avoids diminishment of mutilplied probalities.

Outlier scores are between (0, $+\infty$), which indicate the relative measure of how likely X_i is an oultlier when compared to other points in the dataset. The bigger $O(X_i)$ is, the more likely X_i is an outlier. Hence, we can selet a threshold to find outlier which exceed the threshold or find top k% data record as outlier.

If a point has a large outlier score, then at least one of the three tail copulas is large. Intuitively, the smaller the tail probability is, the bigger its minus log, and so we claim a point is an outlier if it has a small left tail probability, a small right tail probability, or a small skewness corrected tail probability.

* Input: a $n * d$ input dataset X
* Output: Outlier score (label) for each data $Xi(i = 1, ...n)$

(1) Step 1:

* Fit empirical lef tail cumulative distribution function for each dimension.
* Fit empirical right tail cumulative distribution function for each dimension.
* Calculate skewness coefficient vector for estimating skewness. The standard formula for estimating skewness is as follows.

$$b_i = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x}_i)^3}{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x}_i)^2}^3} \tag{3}$$

(2) Step 2:

* Compute the left tail empirical copula observations for each data record x_i
* Compute the right tail empirical copula observations for each data record x_i
* Calculate the skewness corrected empirical copula observations based on the following condition.

$$\hat{W}_{d,i} = \hat{U}_{d,i}\ if\ b_d < 0\ otherwise \hat{V}_{d,i} \tag{4}$$

(3) Step 3:

We compute the probability of observing a point at least as extreme as each x_i along each dimension. We take the maximum of the negative log of the probability generated by the left tail empirical copula, right tail empirical copula and skewness corrected empirical copula to be the outlier score. Return the outlier score of X.

3 Implementation of COPOD Algorithm

In our implementation, we divide the work into two parts. BAI,Qian is responsible for calculating the corresponding copula using ecdf, and WANG,Weihong is

responsible for calculating the tail probability, and marking the output result with outlier tag.

In the ecdf(self, X) function, we call the ecdf function in statsmodels. In the get_copula_observations(self, X, type) function, empirical copula observations of left tail ecdf, right tail ecdf and ecdfs with skewness correction are calculated according to different input types.

In the outlier_out function, first call get_ecdf_func to calculate the tail probabilities, and then choose the greatest tail probability. Finally tag the label 1 means outlier, 0 means non-outlier, and then return the result.

```
class COPOD():
    def __init__(self, threshold):
        super(COPOD, self).__init__(threshold=threshold)

    # Call ecdf func
    def ecdf(self, X):
        ecdf = ECDF(X)
        return ecdf(X)

    # Compute empirical copula observations
    def get_copula_observations(self, X, type):
        # Compute left tail ECDFs
        if type == "left":
            self.U = np.apply_along_axis(self.ecdf, 0, X)
            return self.U
        # Compute right tail ECDFs
        elif type == "right":
            self.V = np.apply_along_axis(self.ecdf, 0, -X)
            return self.V
        # Compute Skewness Corrected ECDFs
        elif type == "skewness":
            # Compute bi for Skewness Correction
            self.b = np.sign(np.apply_along_axis(skew, 0, X))
            # if b<0 W = U | if b>0 W = V
            self.W = self.U * -1 * np.sigh(self.b-1)
                   + self.V * np.sign(self.b+1)
            return self.W

    def outlier_out(self, X):
        # Compute the tail probabilities
        self.left = pd.DataFrame(-1*np.log(
               self.get_copula_observations(X, "left")))
        self.right = pd.DataFrame(-1*np.log(
               self.get_copula_observations(X, "right")))
        self.skewness = pd.DataFrame(-1*np.log(
               self.get_copula_observations(X, "skewness"))
```

```
# choose the greatest tail probability
self.max_out = np.maximum(
           self.left, self.right, self.skewness)

self.ol_scores = self.max_out.sum(axis=1).to_numpy()

# tag the label
# 1 means outlier, 0 means non-outlier
for i in range(len(self.ol_scores)):
    if self.ol_scores[i] >= self.threshold:
        self.labels[i] = 1
    else:
        self.labels[i] = 0
return self.labels
```

4 Application

After understanding the backgrounds of outlier detection and the concepts of the COPOD algorithm, we then explored its application potential by trying to adapt the algorithm to solve some real-life problems. As we have stated in our proposal, we have made implementation attempts focusing on three distinct scenarios: image, text, and audio.

4.1 Image Application

One possible image-related application scenario of COPOD is fingerprint recognition, which has been a common functionality in mobile devices like phones and laptops. With the help of COPOD for preprocessing, we hoped that the fingerprint recognition will have a higher chance of receiving meaningful data, whcih is fingerprint. In this scenario, the inliers are the fingerprints which need further validation from the recognition program, while the outliers are patterns which are distorted fingerprints or patterns that are not fingerprint at all and are meaningless to be progressed to recognition. There are two main benefits of this implementation. Firstly, it can save time and power since COPOD is computationally efficient. Secondly, it may improve the performance of fingerprint recognition programs by keeping them from handling meaningless data.

The dataset of fingerprints we used is UPEK Fingerprint Database [1] which contains 128 fingerprint sample pictures, and the dataset for pattern outliers is the Clothing Pattern Dataset from GitHub [4] We trained two models to test how COPOD performs. In the first model, the outliers are defined as clothing patterns, and in the second model, the outliers are defined as distorted or incomplete fingerprints.

In the first model, the dataset contains 160 samples in total, 128 of which are fingerprints and 32 are cloth pattern pictures. After fitting with the COPOD, the

```
On Training Data:
COPOD ROC:0.8779, precision @ rank n:0.5312
```

Fig. 2: Results of the first model.

training precision was only 0.5312. When fitting the data points of just outliers, we found that precision further dropped to 0.3125, which is not desirable.

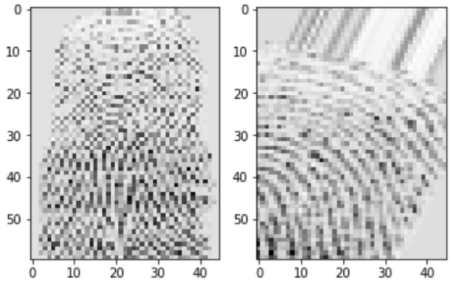

Fig. 3: Sample result of image distortion. left: origin. right: distorted

In the second model where the outliers are distorted fingerprints, to make the datasets for training and testing, we generated outliers from the origin sample fingerprints, which were distorted by the `ImageDataGenerator` package from `keras.preprocessing.image`. Sample output of such distortion is shown in Fig. 3. We then built our dataset by combining the origin pictures and the distorted ones. Fig. 4 is the score of COPOD fitting the new dataset.

```
On Training Data:
COPOD ROC:0.8569, precision @ rank n:0.5625
```

Fig. 4: Results of the second model.

The ROC scores of COPOD on training data and test data were 0.8759 and 0.865 respectively, which were pretty nice, but the precision is not high enough. The precision did not increase compared to the first model. However, we were still interested in testing how this model performs in the scenario we discussed in the first graph. Therefore, we manually fed some pattern pictures to the model, and Fig. 4 shows its performance. The model only failed on the fingerprint-like trademark, and passes all the tests which are not fingerprint-like. This is really surprising since the model is not trained with such data at all. To test it more thoroughly, we used the second model to predict the outlier dataset from the first model and achieved a precision with 0.6875, which was much higher than the precision from the first model.

To conclude, the model does not perform very well in this scenario. For raw pictures, the total dimension could be really high, and in this test, the total dimension reaches 2700. Even though the model has reasonable speed on processing the data, the performance is also not desirable enough.

4.2 Audio Application

In the audio-related setting, the problem we proposed to solve using the COPOD algorithm is that, in music pieces of similar music genre, identifying those with

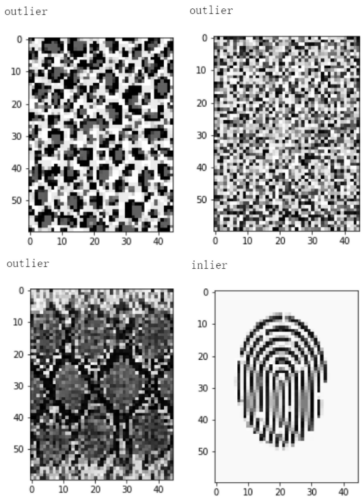

Fig. 5: First two: leopard patterns; left in the second row: serpentine pattern; right in the second row: a fingerprint-like trademark.

unusual genre. Solving such problem can be useful in multiple situations: a music library management system may want to detect songs that are labeled to wrong genre tags; a music player software can also provide helpful information about music pieces that are distinct from rest of its album in terms of genre.

By applying the COPOD algorithm in the genre outlier detection problem, we can utilize some of its advantages. For example, because of COPOD's simplicity and high performance, such detection can be performed in real-time, given all pieces in the music library have been vectorized. This enables use cases like genre comparison among selected songs. Also, COPOD's explainability brings finer understanding of the results than naïve approaches, such as calculating the Euclidean distance.

We first attempted building an end-to-end solution with COPOD. With the librosa audio processing library [3], we can convert audio files into Mel Spectrogram, which is a matrix representation of music's amplitude information in the time-frequency space. We then flattened and normalized the matrix data and tried fitting them with the COPOD. Not surprisingly, the end-to-end approach led to unsatisfactory results, where the precision of detecting segments of music pieces with distinct genres is below 20%. This echoes our previous understanding that high-level information like music genre is too subtle to be captured by the COPOD algorithm in raw audio data. It's clear that we need a higher starting point before performing outlier detection.

We therefore decide to perform the outlier detection task on the outputs of existing genre classifiers. The musicnn proposed by Pons et al. [6] is a set of pre-trained neural network models that can calculate genre likelihood scores from the Mel Spectrogram. The output is formatted as 50-dimensional vector where each dimension is a 0-to-1 probability for the input music being a certain genre. We built our dataset by defining the genre likelihood vectors of albums of a certain

genre as inliers, and the vectors of music pieces from many different genres as outliers. Fitting with the COPOD then gives promising results where nearly all the fitted outliers are correct, and the outliers not fitted also has above-average outlier scores. According to Fig. 6, it achieved ROC score of 0.925 and precision score of 0.9.

out.csv

```
Truth,Fitted,OD,File Path,Judgement: Dimentional OD Score (Likelihood Score / Median Likelihood Score)
0,0,64.435,the distant journey to you/morningview+/01 ministry of Truth.m4a,"more beautiful:3.401(0.049/0.023), more
0,0,47.840,the distant journey to you/morningview+/02 head-trick.m4a,"more indie rock:2.708(0.142/0.057), more 00s:2.
0,0,46.840,the distant journey to you/morningview+/03 Melt-Down-Comet.m4a,"more indie:1.792(0.217/0.153), more 80s:1.
0,0,57.794,the distant journey to you/morningview+/04 Here we can never stop marching.m4a,"more indie:2.708(0.313/0.
0,0,58.627,the distant journey to you/morningview+/05 Derelict.m4a,"more indie:2.303(0.272/0.153), more indie rock:2.
0,1,67.791,the distant journey to you/morningview+/06 reach for the sky.m4a,"more indie:3.401(0.381/0.153), more ind
0,0,59.498,the distant journey to you/never knows best/01 never knows best.m4a,"more hard rock:2.303(0.033/0.016), mo
0,0,58.848,"the distant journey to you/Proof of Existence/01 slumbers, shatters.m4a","more male vocalists:2.303(0.00
0,0,56.444,the distant journey to you/Proof of Existence/02 1000 Shining Needles.m4a,"more classic rock:3.401(0.061/
0,0,59.589,"the distant journey to you/Proof of Existence/03 swimming among the stars, unable.m4a","more blues:3.401
0,0,52.491,the distant journey to you/untitled/01 15 miles.m4a,"more 80s:3.401(0.105/0.015), more catchy:2.708(0.004
0,0,48.330,the distant journey to you/untitled/02 head-trick.m4a,"more blues:2.303(0.036/0.014), more 60s:2.303(0.01
0,0,50.502,the distant journey to you/untitled/03 spring of silver snow.m4a,"more ambient:2.708(0.154/0.050), more be
0,0,48.686,the distant journey to you/untitled 2/01 Ultimate Truth.m4a,"more Progressive rock:2.708(0.123/0.055), mor
0,0,55.153,the distant journey to you/untitled 2/03 a frozen pocketwatch.m4a,"more alternative rock:2.708(0.091/0.04
0,0,52.332,the distant journey to you/untitled 2/03 impermanence and decline.m4a,"more beautiful:2.708(0.047/0.023),
0,0,44.380,"the distant journey to you/untitled 3/01 Reawakening ~ ``second'' chance,.m4a","more metal:2.015(0.075/0.
0,0,47.725,the distant journey to you/untitled 3/02 Border of Lies.m4a,"more alternative rock:2.303(0.086/0.045), mor
0,0,60.842,the distant journey to you/untitled 3/04 a bloodstained sundial.m4a,"more metal:3.401(0.157/0.022), more h
0,0,57.192,the distant journey to you/untitled 3/04 The Finalists of the 1949 English.m4a,"more alternative rock:3.40
1,1,71.546,Foxtail-Grass Studio/Re_Circulation/01 振り子時計の見る世界.m4a,"more chillout:3.401(0.146/0.028), less 90s
1,1,72.380,Foxtail-Grass Studio/かぜがたり。/01 日出ずる国の神語り.m4a,"more jazz:2.708(0.142/0.035), more soul:2.708(0
1,1,66.803,Foxtail-Grass Studio/なつかぜメモリア/01 萃う夢を想う.m4a,"more oldies:3.401(0.019/0.004), more easy listeni
1,1,93.952,TUMENECO/Re_TMNC Acoustic Collection/01 エンドロールのそのあとで.m4a,"more pop:3.401(0.197/0.040), more fema
1,0,66.230,Halozy/Grand Slam/1-06 見えない手.mp3,"more ambient:3.401(0.220/0.050), more female vocalists:2.708(0.093/
1,1,66.485,Halozy/犬猫的電子座曲/01 着席.m4a,"more jazz:3.401(0.359/0.035), more Hip-Hop:3.401(0.025/0.003), more funk
1,1,72.198,DJ OKAWARI/Kaleidoscope/01 Encounter.m4a,"more soul:3.401(0.060/0.006), more chillout:2.708(0.140/0.028),
1,1,81.539,Nami Haven/Eyes On You Remix Compilation/01 Eyes On You (くるぶっこちゃん Remix).m4a,"more sexy:3.401(0.008
1,1,66.732,Nami Haven/Void Ocean / Heavy Rain/01 Fuga Romantica ~Intro~.m4a,"more electronic:3.401(0.565/0.075), more
1,1,87.169,Nami Haven/Wasted Neverland/01 The Train To Nowhere.m4a,"less rock:3.401(0.014/0.201), less alternative:3.
```

Fig. 6: Sample output of genre outlier detection between albums[1]

Thanks to the dimensional outlier scores of the COPOD algorithm, not only can we understand more about how each dimension in the data contributes to the results, but we can also possibly apply further optimization based on the dimensional outlier scores. In this genre outlier detection case, we would like to focus on genre features that are more related to our data and get rid of the distractions of outlier scores from uncommon genres. Therefore, we can weight the dimensional outlier scores based on both the corresponding genre features' proportion in a data point and their overall presence in the dataset. Specifically, we tried weighting dimensional outlier score according to the formula

$$\mathrm{WDOS}_{i,j} = \mathrm{median}(\mathrm{GLS}_{d,j} \forall d) \times \mathrm{GLS}_{i,j} \times \mathrm{DOS}_{i,j}$$

where $\mathrm{WDOS}_{i,j}$ stands for weighted dimensional outlier score for data i and feature j, GLS stands for genre likelihood score. After performing such weighting, we observed that outlier data stands out more distinctively in terms of weighted outlier score. This serves as a good example of how the COPOD's explanability helps in its application.

[1] Full content at `https://github.com/yaindrop/musicnn/blob/master/out1.csv`

out.csv

```
Truth,Fitted,OD,File Path,Judgement: Dimentional OD Score (Likelihood Score / Median Likelihood Score)
0,0,2.672,Nami Haven/Eyes On You Remix Compilation/01 Eyes On You (くるぶっこちゃん Remix).m4a,"less electronic:0.650(0.398/0.
0,0,1.722,Nami Haven/Eyes On You Remix Compilation/02 Eyes On You (MiraGevot Remix).m4a,"more dance:0.404(0.368/0.292), more
0,0,1.358,Nami Haven/Eyes On You Remix Compilation/03 Eyes On You (cnsouka Remix).m4a,"more electronic:0.430(0.537/0.481), m
0,0,2.758,Nami Haven/Eyes On You Remix Compilation/04 Eyes On You (Joulez_s _From the S.m4a,"more dance:1.151(0.524/0.292),
0,0,1.798,Nami Haven/Eyes On You Remix Compilation/05 Eyes On You (rmk Remix).m4a,"more electronic:0.337(0.505/0.481), more
0,0,1.600,Nami Haven/Eyes On You Remix Compilation/06 Eyes On You (NceS Remix).m4a,"less electronic:0.390(0.460/0.481), less
0,0,2.606,Nami Haven/Eyes On You Remix Compilation/07 Eyes on You (Tenyu Remix - Powere.m4a,"more electronic:0.992(0.602/0.4
0,0,1.861,Nami Haven/Eyes On You Remix Compilation/08 Eyes On You (buvid with Project N.m4a,"more electronic:0.464(0.543/0.4
0,0,2.141,Nami Haven/Void Ocean / Heavy Rain/01 Fuga Romantica ~Intro~.m4a,"more electronic:0.573(0.565/0.481), less dance:0.
0,0,1.939,Nami Haven/Void Ocean / Heavy Rain/02 Summer Wave.m4a,"less electronic:0.529(0.434/0.481), less dance:0.192(0.292/0
0,0,2.240,Nami Haven/Void Ocean / Heavy Rain/03 Rough Sea.m4a,"more electronic:0.753(0.571/0.481), more ambient:0.328(0.126/0
0,0,2.559,Nami Haven/Void Ocean / Heavy Rain/04 Feel the Ocean Stomp.m4a,"less electronic:0.724(0.374/0.481), less dance:0.24
0,0,2.231,Nami Haven/Void Ocean / Heavy Rain/05 Wander in the Rain.m4a,"more electronic:0.645(0.567/0.481), more electro:0.55
0,0,1.359,Nami Haven/Void Ocean / Heavy Rain/06 Save the Water.m4a,"more electronic:0.384(0.529/0.481), more dance:0.259(0.31
0,0,4.193,Nami Haven/Void Ocean / Heavy Rain/07 Rustring.m4a,"more ambient:0.701(0.213/0.055), more chillout:0.462(0.140/0.02
0,1,4.690,Nami Haven/Void Ocean / Heavy Rain/08 永久の狂い踊り子 ∽ Eternal Lunatic Dancer.m4a,"less electronic:1.253(0.208/0.4
0,0,2.823,Nami Haven/Wasted Neverland/01 The Train To Nowhere.m4a,"more House:0.885(0.268/0.103), more dance:0.563(0.417/0.2
0,0,1.539,Nami Haven/Wasted Neverland/02 Beyond Her Eyeshot.m4a,"more electronic:0.406(0.535/0.481), less dance:0.195(0.281/
0,0,1.669,Nami Haven/Wasted Neverland/03 Memoria.m4a,"more electronic:0.515(0.557/0.481), more dance:0.289(0.322/0.292), mor
0,0,2.455,Nami Haven/Wasted Neverland/04 Phantom X.m4a,"more dance:0.984(0.515/0.292), more House:0.549(0.211/0.103), less e
0,0,3.838,Nami Haven/Wasted Neverland/05 Wested Neverland.m4a,"more dance:2.008(0.609/0.292), less electronic:0.812(0.335/0.
0,0,1.624,Nami Haven/Wasted Neverland/06 Eyes On You.m4a,"more dance:0.645(0.429/0.292), more electronic:0.327(0.496/0.481),
0,0,3.434,Nami Haven/Wasted Neverland/07 忘郷 ～ Journey.m4a,"less electronic:0.919(0.281/0.481), more dance:0.773(0.459/0.29
0,0,3.622,"Nami Haven/Wasted Neverland/08 All Covet, All Lose.m4a","more dance:1.393(0.535/0.292), less electronic:0.585(0.3
0,0,1.908,Nami Haven/Wasted Neverland/09 煙る夢.m4a,"less electronic:0.352(0.479/0.481), less dance:0.212(0.237/0.292), more
0,1,4.523,Nami Haven/Wasted Neverland/10 Midnight Wanderland.m4a,"less electronic:1.057(0.243/0.481), more pop:0.534(0.205/0
0,1,7.107,Nami Haven/Wasted Neverland/11 幻 ～ In her dreams.m4a,"less electronic:1.586(0.055/0.481), more Progressive rock:
```

Fig. 7: Sample output of inter-album genre comparison[2]

4.3 Text Application

Our initial attempt of applying the COPOD algorithm in the text-related direction is to detect whether there are any people use formal language when they are commenting other people's twitter. We download the dataset online which contains several elements, such as the records of the ID of commenters, the name of those commenters, their comments and so on. We cannot use this data directly because the only thing we need is their comments. The other problem is we need to find a way to vectorize all those comments to make them became a vector with the same length, because COPOD cannot resolve the text directly. The other difficulty of preprocessing the comments we met is that the comments contain myriads of unreadable or meaningless characters and there are some words do not even make any sense. We need to find a way to deal with all those situations.

In our group meeting, we believed that using the tf-idf algorithm to vectorize each comment can worth a try, because it can vectorize the comment with high efficiency. After vectorizing, the length of each comment vector will be the number of different words in the whole training set. We then added labels to the data which tells the COPOD which comments should be formal and which comments should be informal. After fitting the tf-idf vectors with the COPOD, we failed to achieve meaningful results: not only was the precision score low, we also couldn't observe any useful pattern between the outlier scores and the ground truth. We believe the reason for such results can be the ambiguity of word choice preference in both formal and informal text, which makes the COPOD hard to detect statistical patterns. Therefore, we decided to adjust our direction.

[2] Open problem, no truth defined, full content at `https://github.com/yaindrop/musicnn/blob/master/out.csv`

After realizing the need for some dataset containing more distinct patterns. We immediately thought of detecting the positive and negative sentiments of comments. We applied the same preprocessing procedure but utilized the sentiment labeling from the dataset. We added 5000 negative comments into the training set, as well as 50 positive comments. After fitting the training set with COPOD, we also built a testing set which contains 1000 negative comments and 10 positive comments.

However, we still found it hard to achieve reasonable results or to observe patterns in outlier scores. The Rank score was almost 0 which means COPOD labeled each comment as an outlier. Based on this situation, we made several guesses about why COPOD cannot give us a good outlier detection result. The first guess is vectorized comments have too many dimensions which COPOD cannot handle. The second guess is the information that stored in text vector is not enough for COPOD to detect the outlier. The last guess is TF*IDF is not a good for vectorization because the original purpose of TF*IDF is to calculate the similarity of the text.

```
(5049, 10594) (1009, 10594)
before fit
after fit

On Training Data:
COPOD ROC:0.4902, precision @ rank n:0.02
[104.16759498  94.84452779  63.24431723 ...  49.76886201  53.64863954
  86.38169174]

On Test Data:
COPOD ROC:0.3641, precision @ rank n:0.0
[ 50.54586111 101.13649374  65.12210873 ...  51.22100832  49.9652397
  14.88191544]
(base) yuximengdeMacBook-Air:DM project yuximeng$
```

Fig. 8: Results of the sentimental outlier detection task

After failing twice, we believe that the reason why we keep getting unsatisfactory result is not only because we did not have the dataset with strong enough feature, but also our data is not relevant enough with our goal. Therefore, we applied the third try.

For our third try, we chose to give up on using tf-idf vectors and shifted our focus on another text vectorization method called sentence vector. This method is based on the concept of mapping sentences into higher dimensional space using pretrained neural network models, with the target that sentences with similar meaning will be mapped to closer vectors in such space.

We also switched to a new task of detecting outliers among research paper titles. Therefore, after vectoring the title texts with the sentence vector method, the processed inputs should be highly related to our new task. We define the titles of research papers from a certain academic field as the inlier, and paper titles from other fields as the outlier. We acquired the pretrained models from the library sentence transformer [1].

Although the COPOD algorithm did not detect all the outliers, it is still much better than just doing TF*IDF vectorization. This indicates that making the goal more relate to the data we have can somehow improve the performance of COPOD on the outlier detection.

4.4 Discussion

According to our experience in implementing the three applications scenarios, we can confirm that the COPOD algorithm is not suitable for dealing with raw data, either in picture, text, or audio scenarios. As a purely statistical model, it can make few progress when it comes to latent data features related to the task, because its unsupervised natural makes it impossible to be trained toward any specific target. However, COPOD does perform well when the data shows enough statistical patterns. Considering what we have tried in the audio scenario, we believe that COPOD can be better utilized in post-process tasks.

Therefore, when the application of COPOD in machine learning is needed, which is fast and easy to apply, end-to-end application should be avoided, and COPOD should be combined with other data mining techniques to achieve highest performance.

5 Contribution

* BAI, Qian:
  In this project, I am mainly responsible for the implementation of the COPOD algorithm, collaborating with WeiHong Wang. For me, I implemented the computing function of empirical copula observations in the COPOD algorithm. In addition, for the proposal, I introduced the authors' experiments and their implementations. For the final report, I introduced the COPOD algorithm in details. In report, I analysed the COPOD's idea based on Wang's introduction of Copula and explained the output of COPOD when using COPOD as an Interpretable outlier detector.

* TONG, Jinjian:
  For the project proposal, I introduced outlier detection and the drawbacks of traditional outlier detection. For the project itself, I mainly worked on the application of COPOD with Dian Yuan and Ximeng Yu. We followed what the team discussed in the proposal and came up with three concrete examples. The scenario I worked on is the picture scenario. In the picture scenario, I want to find out how COPOD performs when treating fingerprints as inliers. I trained two models for this. The first model is trained with cloth patterns as outlier, while the second model is trained with distorted fingerprints. I evaluated the performance of both models and discussed the possible application.

* YUAN Dian:
  I proposed the idea of making application attempts in multiple scenarios in the project proposal. I then explored the audio related applications on my own, and created an album genre comparison tool which can be found on GitHub [3]. In the project report, I was responsible for stating my results as

[3] `https://github.com/yaindrop/musicnn/blob/master/musicod.py`

well as some proofreading works of the three application parts.

* WANG, Weihong:
In the proposal, I was responsible for investigating the advantages of COPOD. In the final report, I was responsible for the introduction of Inverse Sampling and copula description. In the implementation collaborating with Qian Bai, I was responsible for the calculation of the tail probability and the output result.

* YU Ximeng:
For the proposal, my job was to figure out what does Copula mean in Copula-based-outlier-detection. After doing some search and understanding, I finished the introduce of what Copula is and what we can do by using Copula. In the following group meeting, I worked with Jinjian Tong and Dian Yuan in the real-world implementation of COPOD direction. My job was to figure out when we can use COPOD algorithm to detect the date in the text format. I tried first two models by myself and cooperate with Dian Yuan to tried the last model. For the final report, I explained what I have done in my direction in details, and we combine our report together.

References

1. Free benchmark - upek fingerprint dataset. `http://www.advancedsourcecode.com/PNGfingerprint.rar`
2. Li, Z., Zhao, Y., Botta, N., Ionescu, C., Hu, X.: COPOD: copula-based outlier detection. In: IEEE International Conference on Data Mining (ICDM). IEEE (2020)
3. McFee, Brian, C.R.D.L.D.P.E.M.M.E.B., Nieto., O.: librosa: Audio and music signal analysis in python. In: The 14th python in science conference (2015)
4. Medeiros, A.J., Stearns, L., Findlater, L., Chen, C., Froehlich, J.E.: Recognizing clothing colors and visual textures using a finger-mounted camera: An initial investigation. In: Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility - ASSETS '17 (2017)
5. Nelsen, R.B.: An Introduction to Copulas. Springer Science Business Media (2007)
6. Pons, J., Serra, X.: musicnn: pre-trained convolutional neural networks for music audio tagging. In: Late-breaking/demo session in 20th International Society for Music Information Retrieval Conference (LBD-ISMIR2019) (2019)