Order Statistics Based Training and Scoring Algorithms for Deep Outlier Detection

Ahmet Zahid Balcıoğlu [*], Erhan Çene [†]

April 30, 2022

Abstract

Autoencoder networks are among the most popular deep learning methods for outlier detection. They work as an unsupervised algorithm that aims to reconstruct original instances through learning a low-dimensional future space. One of the main weaknesses of outlier detection using autoencoders is that the training loss metric is also used for outlier reconstruction errors, which can lead models to learn outliers just as well as normal instances. In this paper, we propose to solve this problem by introducing a robust order statistics based scoring system. We discuss the chosen scoring system and demonstrate its effectiveness through simulation studies. We also show that with our proposed scoring system autoencoder model no longer needs the percentage of anomalies as a hyperparameter to estimate the cut-off point. We also introduce an order statistics augmented loss function, which creates a more robust model by penalizing the 'leveraging effect' of outliers in the loss signal. We show that training with our algorithms show a significant improvement when compared to conventional autoencoder training on image and medical datasets.

Keywords: anomaly detection, outlier detection, order statistics, robust statistics, autoencoder

***NOTE: This is a short draft of my MS thesis prepared for doctoral degree applications.*

1 Outlier Detection Using Autoencoder Networks

1.1 Autoencoder Networks

An autoencoder is a neural network that is trained to attempt to copy its input to its output. [3] It consists of two symmetric parts, with at least a hidden layer in between. The first part, also known as the encoding, or encoder network, tries to learn some low-dimensional feature representation space for the data set. The second part, likewise called the decoder network, tries to recover the original data instances from the low-dimensional representation. The network is trained to minimeze the recovery error, which is most commonly calculated as the L_2 distance between an observation and its recovery value.

The basic formulation is given as follows.

[*]Yıldız Technical University, Department of Statistics, Istanbul, Turkey, e-mail: zahid.balcioglu@std.yildiz.edu.tr
[†]Yıldız Technical University, Department of Statistics, Istanbul, Turkey, e-mail: ecene@yildiz.edu.tr

Definition 1.1 (Autoencoder Network [6]). *An autoencoder network consisting of an encoder network ϕ_e and a decoder network ϕ_d, with the corresponding parameters Θ_e and Θ_d.*

$$\begin{array}{ll} \textit{for} & \mathbf{z} = \phi_e\left(\mathbf{x}; \Theta_e\right), \hat{\mathbf{x}} = \phi_d\left(\mathbf{z}; \Theta_d\right) \\ \textit{minimize} & \sum_{\mathbf{x} \in X} \mathcal{L}\left(\mathbf{x} - \phi_d\left(\phi_e\left(\mathbf{x}; \Theta_e\right); \Theta_d\right)\right) \end{array}$$

The optimum parameters for the network are

$$\{\Theta_e^*, \Theta_d^*\} = \arg\min_{\Theta_e, \Theta_d} \sum_{\mathbf{x} \in X} \mathcal{L}\left(\mathbf{x} - \phi_d\left(\phi_e\left(\mathbf{x}; \Theta_e\right); \Theta_d\right)\right).$$

If we choose the most commonly used L_2 distance as our loss function we have

$$\left\|\mathbf{x} - \phi_d\left(\phi_e\left(\mathbf{x}; \Theta_e^*\right); \Theta_d^*\right)\right\|^2.$$

Autoencoder networks behave anologous to principal component analysis (PCA) as they both try to learn a lower dimensional representation of the data set. In the case of PCA, the lower dimensional representation is reached using singular value decompostion. In the case of autoencoders, the model is trained to optimize over a set of parameters for the chosen loss function and the lower dimensional representation is learned as the hidden layer of the network.

Compared to PCA, autoencoders are more complicated models that have the advantage of learning non-linear representations of the data. In fact when the decoder is linear and the loss is chosen as L_2, an autoencoder learns to span the same subspace as PCA. [3] One key disadvantage of autoencoders is that one can not choose the number of dimensions for the hidden layer after training.

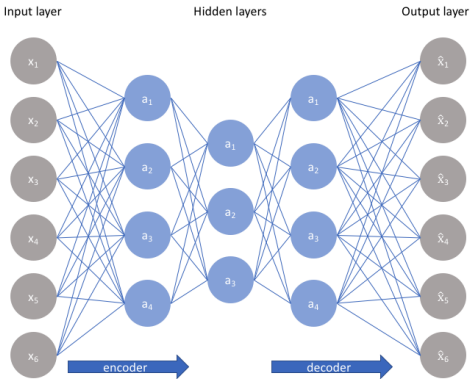


Figure 1: Structure of an autoencoder with three hidden layers.[2]

1.2 Autoencoders in Anomaly Detection

PCA is one of the classical methods used for outlier detection in multivariate data. All methods using PCA rely on outliers having very litle correlation with the principal components. Typically they perform PCA, or robust PCA on the entire data, once data is reduced to a lower number of dimensions, outliers can be identified as data points that do not conform to rest of the data. For

example [9] used a method in which they assigned an anomaly scores to each point based on its distance from the principal components.

Autoencoder networks are among the most popular models for outlier detection. Their ability to learn lower dimensional representations of the data in an unsupervised manner, and providing recovery errors for each data point makes them very attractive for outlier detection. We have already mentioned that autoencoder networks share a good deal of similarities with PCA, and in fact when the encoder is linear they learn the same space as PCA if trained under L_2 loss. Outlier detection method is also similar to PCA, as it depends on the lower dimensional representation to distinguish outliers within the data.

Autoencoders learn a lower dimensional manifold for the data set. After training, the boundaries of this manifold can be charactarized by a threshold value which corresponds to the error values for the data. Points which are not included in the manifold are condidates to be outliers. Thus, a key assumption in autoencoder models is that anomalous data instances will be more difficult to recover compared to normal instances, and have a higher recovery error. In practice, many methods use the recovery error directly as the outlier score. Data points which have a higher recovery error than a certain threshold are considered to be outliers.

We summarise the most elementary algorithmic for autoencoder outlier detection.

Algorithm 1: Outlier Detection with Autoencoders

Data: $\mathcal{X}$, training data

Train the autoencoder using definition 1.1 for optimum parameters Θ_e^* and Θ_d^*

Calculate reconstruction loss for each data point $\mathbf{x} \in \mathcal{X}$:

$$\mathcal{L}(\mathbf{x}) = \left\| \mathbf{x} - \phi_d \left(\phi_e \left(\mathbf{x}; \Theta_e^* \right); \Theta_d^* \right) \right\|^2 .$$

Select a threshold T based on the calculated losses

Observations above T are labelled as outliers

$$\mathcal{O} = \{x \in \mathcal{X} : \|x - \hat{x}\| \geq T\}.$$

1.3 Literature Review

1.4 Shortcomings of Autoencoders

One of the key problems of outlier detection is dealing with high-dimensional data, in which most machine learning models fail to produce satisfactory results due to curse of dimensionality.[6] Autoencoders tend be effective against high-dimensional data due to their complex and non-linear nature. However, there are a few key points which their complexity and non-linearity becomes hindrance.

First, their complexity allows them to generalize to outliers just as well as the rest of the data, effectively memorizing the data set.[5] This is also exacerbated by non-linearity, which makes it difficult to determine when to stop the training process. Moreover, even at the beginning of the training process data points with most errors tend to be outliers, [5] which makes them behave as potential points of leverage due to their disproportionate contribution to the model early on.

Second, most outlier detection methods using autoencoders use the model residuals as outlier scores. Loss function having a double purpose in training, and scoring introduces some bias to the

data. It also makes the model dependent upon the choice of loss function. Another problem with the scoring method is that usually there is no clear cut-off point with which to seperate outliers. As a result most studies either assume as a priori a percentage of contamination, or use another method on top of the autoencoder to determine the cut-off point.

2 Role of Order Statistics

Order statistics is an important tool for order statistics, as they can allow to treat data independent of its distribution. Order statistics have uses in related fields such as extreme value theory. In many applications of autoencoders, quantiles for outliers are assumed to be known as a way to quantify outliers. Here we will go one step further and decide upon a scoring function using order statistics and eliminate the need for such assumptions.

Definition 2.1. *Let $X_1, X_2, \ldots, X_n$ be i.i.d. random variables with cumulative distribution function F. We may further assume that these are absolutely continuous, and call the common p.d.f. f. Let $X_{(1)}, \ldots, X_{(n)}$ be the corresponding (increasing) order statistics. We are to investigate the problem of outliers in a way that the number of κ-outliers is defined via moving blocks as*

$$\zeta_n \coloneqq n - \min\left\{ i : \sum_{j=1}^{i} |X_{(j)}| < \kappa \sum_{j=i+1}^{n} |X_{(j)}| \right\} \tag{1}$$

The definition 2.1 may also be potentially useful for analyzing time series in a nonparametric way (i.e., without the normality or a similarly distributional assumption). Assume that the X_i's are non-negative. We start by defining the following statistics

$$T_{k,n} \equiv T_k = \sum_{i=n-k+1}^{n} X_{(i)}, \quad 1 \le k \le n \tag{2}$$

denoting the sum of the top $k \in \{1, \ldots, n\}$ order statistics from a sample of size n and

$$S_{m,n} \equiv S_m = \sum_{i=1}^{m} X_{(i)}, \quad 1 \le m \le n \tag{3}$$

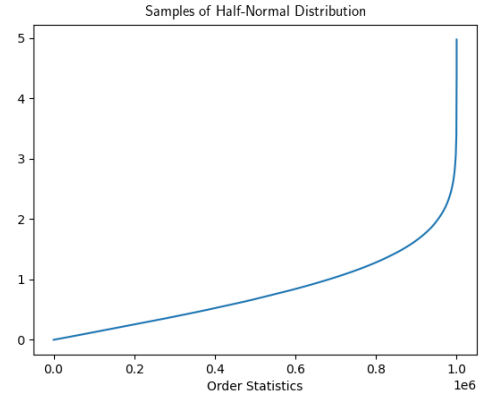
denoting the sum of the first $m \in \{1, \ldots, n\}$ order statistics from the same sample. Subsequently, putting (2) and (3) together, we investigate probabilities of the form $P\left(S_m < \kappa T_{n-m}\right)$ where κ is fixed. We call the random variable

$$R \coloneqq \mathbb{P}\left(\frac{S_m}{T_{n-m}} \le \kappa \right)$$

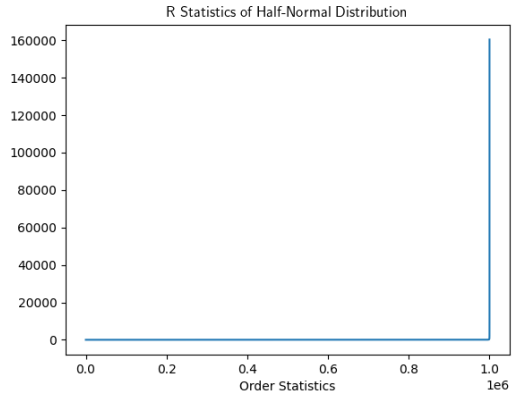
as our outlier statistic.

2.1 How does this statistic look?

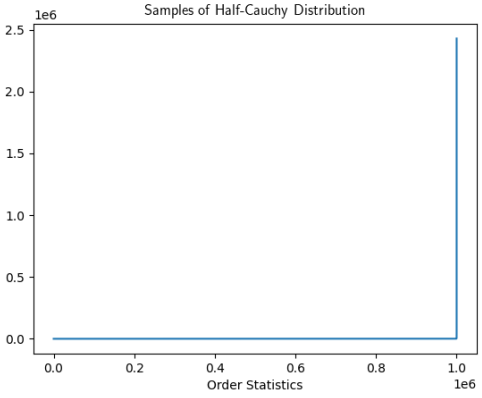
We will look at Half-Normal, Half-Cauchy, and χ^2 distributions. We will take a sample of 1000000 from each distribution and plot the results for each order statistic.



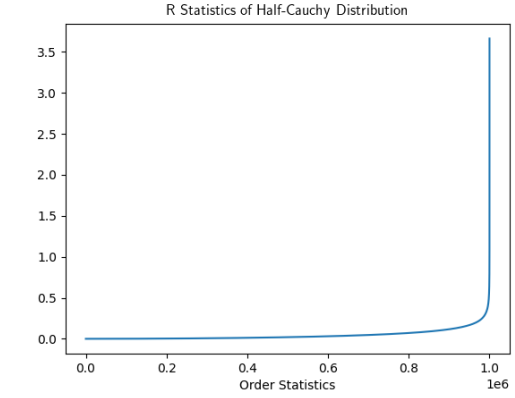
Half-Normal Distribution



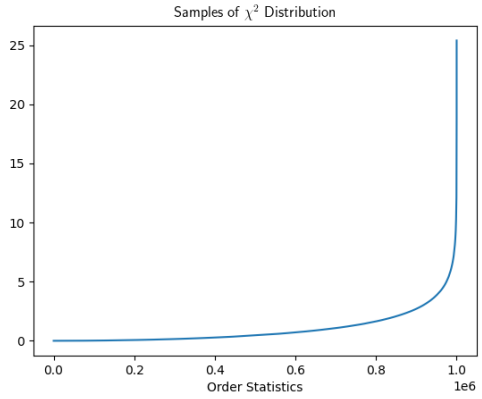
Statistic for Half-Normal Distribution



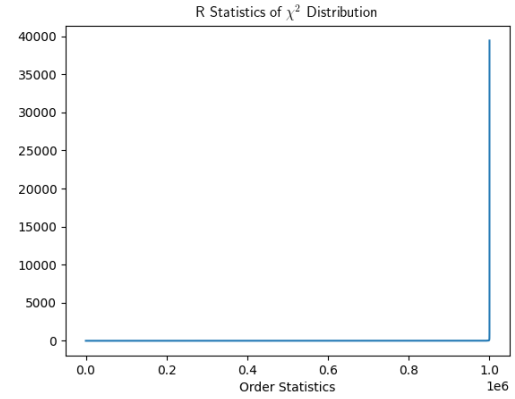
Half-Cauchy Distribution



Statistic for Half-Cauchy Distribution



χ^2 Distribution



Statistic for χ^2 Distribution

Figure 2: Sorted samples of 1000000 taken from half-normal, half-cauchy and χ^2 distributions. We show the order statistic from the samples on the left, and the corresponding R statistics on the right.

3 Selecting Threshold Kappa Value

How to select a cut-off point As with any outlier model, our statistic needs a method for reliably selecting a cut-off point and a corresponding κ threshold point. One option would be to decide on a case-by-case basis what the threshold κ value should be. Alternatively, we may make use of the distribution function of our statistic and pick a value for the κ threshold using an algorithmic approach. One advantage of our statistic is that there is a very clear cut region for threshold compared to the probability distributions. Indeed, as the plots show even in the heavy-tailed cauchy distribution, our statistic produces a definite "elbow-like region".

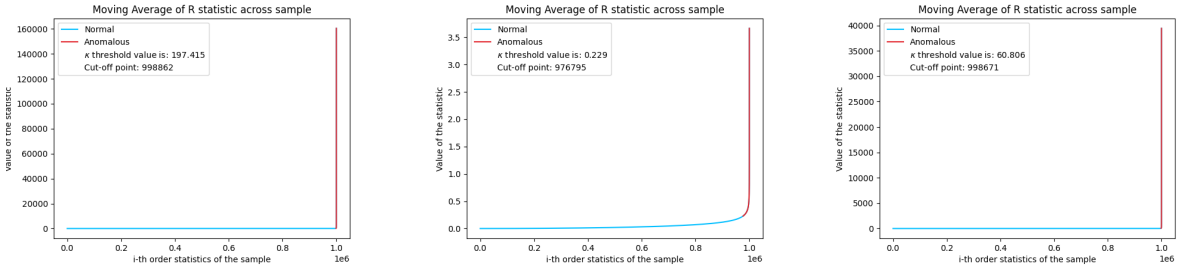
Elbow Detection Ideally, we would like to define the cut-off point with respect to the derivatives of our distribution function, which then can be estimated from a given sample. However, there is no established well-defined notion for the elbow region, which our statistic produces.[1]

Many works in the kneecap or elbow detection literature is based on the following pointwise definition of the curvature of a function. Most works than define the elbow as the point of maximum curvature, and use algorithmic means to calculate it.

Definition 3.1 (Curvature of a function [8])**.** *For any continuous function f, there exists a standard closed-form $K_f(x)$ that defines the curvature of f at any point as a function of its first and second derivative:*

$$K_f(x) = \frac{f''(x)}{(1 + f'(x)^2)^{\frac{3}{2}}}$$

We will use the kneedle detection algorithm for estimating the "elbow point" of our statistic [8]. The kneedle algorithm uses dynamic first derivative threshold, in combination with he IsoData [7] to find the elbows of a discrete data. It can work on discrete datasets and uses a sensitivity parameter to single out potential elbows. We choose the sensitivity parameter as 5.0 in our experiments. You can see the chosen κ thresholds for half-normal, half-cauchy, and chisquare distributions respectively.



Half-Normal Distribution | Half-Cauchy Distribution | χ^2 distribution

Figure 3: Outlier κ-thresholds selected by the kneedle algorithm for the samples in 2.1

4 Order Statistic Augmented Loss

Here we will discuss the ways we have used to incorporate the R-statistic into autoencoder training. We have already mentioned that one particular weaknesses of autoencoders is that due to the high

complexity of the model, it can generalize to outliers just as well as normal observations. Practical remedies such as regularization, early stopping, and dropout layers can be used to ameliorating this weakness. It has also been reported that early in the optimization process the gap between outliers and normal observations is the most noticable.[5]

We can use our statistic throughout the optimization procedure and use the statistic to create a weighted some during the calculation of the loss function. In this way we can regularize the contribution of each observation to the loss by its outlier score, and retain the initial distinction between outliers and the rest of the data while optimizing for a model with greater generalizability.

In order to make use of our statistic, we sample some proportion of the data to use as a scoring system for the entire data. As our statistic is relatively inexpensive to calculate, we may also use the entire data. After every epoch, we calculate our statistic on this entire sample to use as outlier scores and also use kneecap detection to estimate a κ threshold. In the next iteration, after calculation of loss for each batch we calculate the weight of each observation by the outlier score of the nearest loss value in the initial sample. We also experiment with a few methods on how each weight would be calculated.

4.1 Investigating weight functions

First it is possible to use the inverse outlier scores in order to increase the weights of normal observations and decrease the weights of the outlier. However, as the weights would not be normalized this method would tempers with the loss function too much and results in a frail optimization procedure dependent upon the balance in each batch.

Next, the simplest method is to remove the weights of any observation above the κ threshold to restrain the influence of outliers in the training. This has a few important advantages. It tempers very little with the loss function and does incur very little extra cost on calculation. More importanly since it bars data above the threshold from making a contribution, any decrease in their loss is attributed to normal data. Therefore, after optimization this method results in a model with outliers that share very little with the rest of the data.

Finally, as a compromise between two methods we can apply inverse weights to data above the threshold and leave the weights of data below threshold untoched as in the first case. The motivation for this method is to be able to make use of the entire data set while limiting the contribution from any potential outliers.

Algorithm 2: Order Statistic Adjusted Loss function

```
Data: outlier_sample := Sample a portion of the data
Intialize uniform loss weights;
for epoch in Total Epochs do
    for batch in Data do
        Calculate loss; Estimate loss weights using outlier_sample;
    end
    Calculate R-statistic for the outlier_sample;
end
```

5 Kappa Threshold Early Stopping

Deep neural networks are not convex statistical models, as a result their training takes longer and do not necessarily converge to a global minumum. This leads to a search for methods which can determine the time to finish the training when the model is sucessful enough. Early stopping is one such method. It works by monitoring a metric pertinent to model sucess and stopping the training when there is no meaningful improvement on the metric's value.

Autoencoders have high complexity if they are trained until convergence, then they may be able to generalize to normal data and outliers equally well. Early stopping helps to stop autoencoder training before the model starts learning about outliers, leading to a model that can more easily distinguish between outliers and the rest of the data.

Autoencoders are unsupervised methods, which limits the options for choosing a reliable metric for early stopping. One likely candiate is the loss function itself, [5] but the loss function is also used for other algorithms that dynamically change optimizer parameters such as learning rate. The κ threshold value, which is calculated each epoch in order statistic augmented loss, can be a potential metric to use. Since stability in the κ value shows that the model has stopped distinguishing between outliers and the normal data. During training we would expect the κ threshold to decrease as the loss is decreasing and stop when the changes in the loss no longer contribute to distinguishing outliers.

In practice, because it is a point estimate rather than a summary statistic like most metrics, κ threshold is not very consistent. Between consecutive epochs κ threshold value can increase, despite an overall decreasing trend; it also has a tendency to increase early in the optimization. As a result it is difficult find a general set of criteria for κ threshold to be used in early stopping.

The first problem does cause much obstacle, as we are already not interested in stopping training at the very early epoch in which the model has not yet learned enough from the data. We can look for smoothing methods in order to cope with the second problem. One simple idea would be to use moving block averages. This can lead to a much smoother curve with a more prominent decreasing trend.

Algorithm 3: Early Stopping algorithm using κ threshold

Data: Data := Training data
Data: `outlier_sample` := Sample a portion of the Data
`int early_iteration_limit`
`int block_size`
κ`_epochs = []`
Function `check_imporevement(`κ*_epochs, block_size, early_iteration_limit*`):`
  if *size(*κ*_epochs)* $\leq$ *`early_iteration_limit`* **then**
    return *continue*
  end
  `block_`κ = `block_sum(`κ`_epochs, block_size)`
  if *`block_sum`* *not increasing* **then**
    return *stop*
  else
    return *continue*
  end
for *epoch in Total Epochs* **do**
  for *batch in Data* **do**
    Calculate loss
  end
  Calculate R-statistic for the `outlier_sample`
  Calculate κ-threshold for the epoch
  κ_epochs.append(κ-threshold)
  `check_imporevement(`κ`_epochs, block_size, early_iteration_limit)`
end

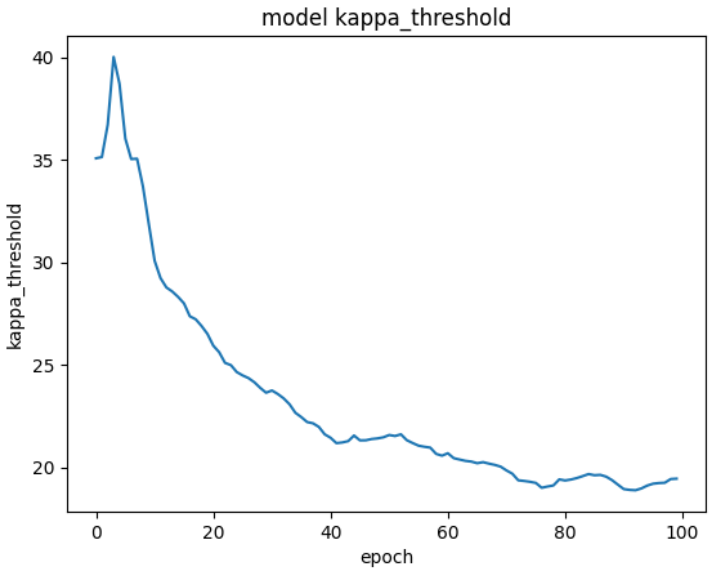


Figure 4: An example behaviour of kappa value throughout training.

6 Experiments

We conduct two kinds of experiments, one with autoencoders on the MNIST [4] data set, and simulation studies using two pareto tails in order to portray the efficacy of our statistic.

6.1 Experiments on MNIST digits

MNIST is an image data set consisting of handwritten digits. We select a digit as to represent normal instances and then undersample from the rest of the data, or from another selected digit to represent anomalies. This method allows us to conduct ablation studies more effectively as we can more easily distinguish the contributions from our algorithms.

For all the results below we use the same autoencoder network with linear layers, the weights are respectively of shapes $(784, 256)$, $(256, 32)$, $(32, 256)$, $(256, 784)$. From the data set we select number 2 as the normal instance and sample from 4. Our final sample has an outlier rate of 2 percent. We first test a simple autoencoder with Adam optimizer with initial learning rate set at 1×10^{-3}, then we add the augmented loss function, and finally we test the full algorithm with augemented loss and kappa early stopping.Batch size in each training is set to 2991 and number of epochs is set to 200 epochs unless it is stopped preemptively with early stopping. We report the confusion matrix and $F1$ scores of the results below, in each case we use order statistics scoring.

Algorithm	$F1$	Confustion Matrix
Simple Autoencoder	0.825	5958 0 36 85
with augmented loss	0.83	5958 0 35 86
with augmented loss and early stopping	0.836	5958 0 34 87

It is important to note that the success of the algorithm increases but little. This may be due to the fact that we have used order statistics based scoring across all methods. One important feature which remains consistent is the lack of false positives. The order statistics may again play a role in this as it generally cuts the very end of a distribution. Changing the sensivity parameter for the kneedle algorithm may help with finding an earlier threshold.

6.2 Distinguishing two Pareto tails

We conduct simulation tests on distinguishing two Pareto tails. Our goal is to find a threshold κ statistic beyond which $> .90$ of the observations belong to the heavy tail. We step our experiment as a follows. For a given two tail indices $\alpha_1 < \alpha_2$. We first determine a κ threshold value for α_1 indexed pareto distribution. Then we sample a $N = 1000000$ observations from both of the distributions, and calculate our statistic across the entire sample. We report the percentage of observations from α_2 in the samples above the predetermined threshold. We repeat this experiment a 1000 times in order to build an confidence interval on our results.

A key problem in our experiments is the selection α_1 and α_2 values, it has been established that as the two indices get closer it becomes numerically difficult to differentiate between the tails, even if one of the tails are outside of the Levy-stable regime.[10] We start our experiments with indices $\alpha_1 = 1.5$ and $\alpha_2 = 2.5$ and move α_2 closer every iteration.

The results of the experiments are given in the table below. When the tail indices are far apart our statistic produces becomes an ideal change point for differentiating between the two tails. It is clear that as the two tail indices get closer our statistic becomes unable to distinguish the difference between the two tails. However, our statistic is still very consistent with variance $\sim 10^{-5}$ across all samples.

α_1	α_2	κ-threshold	Expected percentage of α_2 above κ	Variance
1.5	2.5	2.745	0.95	1.88×10^{-5}
1.5	2.3	2.745	0.924	4.06×10^{-5}
1.5	2.1	2.745	0.85	1×10^{-4}
1.5	1.9	0.78	0.95	7.37×10^{-5}
1.5	1.7	2.745	0.65	3.74×10^{-5}

References

[1] Mário Antunes, Diogo Gomes, and Rui L. Aguiar. "Knee/Elbow Estimation Based on First Derivative Threshold". In: *2018 IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService)*. 2018, pp. 237–240. DOI: 10.1109/BigDataService.2018.00042.

[2] Jordan Jeremy. *Introduction to autoencoders*. [Online; accessed April 3, 2022]. 2018. URL: https://www.jeremyjordan.me/content/images/2018/03/Screen-Shot-2018-03-07-at-8.24.37-AM.png.

[3] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[4] Yann LeCun, Corinna Cortes, and CJ Burges. "MNIST handwritten digit database". In: *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2 (2010).

[5] Nicholas Merrill and Azim Eskandarian. "Modified Autoencoder Training and Scoring for Robust Unsupervised Anomaly Detection in Deep Learning". In: *IEEE Access* 8 (2020), pp. 101824–101833. DOI: 10.1109/ACCESS.2020.2997327.

[6] Guansong Pang et al. "Deep Learning for Anomaly Detection". In: *ACM Computing Surveys* 54.2 (2021), 1–38. ISSN: 1557-7341. DOI: 10.1145/3439950. URL: http://dx.doi.org/10.1145/3439950.

[7] TW Ridler, S Calvard, et al. "Picture thresholding using an iterative selection method". In: *IEEE trans syst Man Cybern* 8.8 (1978), pp. 630–632.

[8] Ville Satopaa et al. "Finding a "Kneedle" in a Haystack: Detecting Knee Points in System Behavior". In: *2011 31st International Conference on Distributed Computing Systems Workshops*. 2011, pp. 166–171. DOI: 10.1109/ICDCSW.2011.20.

[9] Mei-Ling Shyu et al. *A novel anomaly detection scheme based on principal component classifier*. Tech. rep. Miami Univ Coral Gables Fl Dept of Electrical and Computer Engineering, 2003.

[10] Rafał Weron. "Levy-stable distributions revisited: tail index¿ 2 does not exclude the Levy-stable regime". In: *International Journal of Modern Physics C* 12.02 (2001), pp. 209–223.