Hochschule Esslingen
University of Applied Sciences

Forschungsprojekt

Ausreißer-Erkennung in Zeitreihen mittels Graphen-basierter Algorithmen

im Studiengang Angewandte Informatik
der Fakultät Informationstechnik
Wintersemester 2020/2021

Bahar Uzun
764647
Jeremy Kielman
764097
Marcus Erz
762294

Abgabedatum: 28. Februar 2021
Prüferin: Prof. Dr. rer. nat. Gabriele Gühring

Kurzfassung

todo: Kurzfassung erstellen

Schlagwörter: Anomalie-Erkennung, Ausreißer-Erkennung, Netsimile, MIDAS, Random Walk, Graphen-basierte Algorithmen, Zeitreihen

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1 Einleitung

Im Rahmen der Forschungsprojekt werden verschiedene Algorithmen zur Ausreißer-Erkennung in Graphen erforscht und getestet. Nachfolgend soll die Motivation hinter dieser Thematik erläutert werden.

1.1 Hintergrund

todo: formulieren

1.2 Problemstellung

todo: Ziele definieren Das Ziel dieser Forschungsprojekt ist es verschiedene Algorithmen anzuwenden und erste Erkenntnisse aus ihnen zu gewinnen. Dieses Hauptziel, im Zuge des ersten Semesters des Forschungsprojekts, kann wie folgt in drei Teilziele unterteilt werden:

1. Verschaffen eines Überblicks über die existierenden Algorithmen zur Erkennung von Ausreißern in Graphen
2. Die Entwicklung eines Ausreißer-Scores für die zugrundeliegenden Algorithmen
3. Erste Anwendung der verwendeten Graphen-basierten Algorithmen auf Zeitreihen

1.3 Verwandte Arbeiten

todo: related work einfügen

2 Transformation Zeitreihe zu Netzwerk

Ziel unseres Forschungsprojektes war es verschiedene Algorithmen zur dynamischen AusreiÃer Erkennung in Netzwerken, auf Zeitreihen anzuwenden. Um dies zu ermÃ¶glichen mÃ¼ssen die Zeitreihen hierzu zunÃ¤chst in Netzwerke umgewandelt werden. In diesem Kapitel wird dieser Schritt erlÃ¤utert. Je nach Algorithmus der verwendet werden soll, wird die Umwandlung unterschiedlich durchgefÃ¼hrt. Aus diesem Grund ist dieses Kapitel in weitere Unterkapitel aufgegliedert um auf die Besonderheiten bei der Umwandlung fÃ¼r unterschiedliche Algorithmen einzugehen.

2.1 Netsimile

Der Netsimile Algorithmus erwartet fÃ¼r jeden Zeitabschnitt ein Netzwerk. Sodass spÃ¤ter die einzelnen Netzwerke miteinander verglichen werden kÃ¶nnen um auffÃ¤llige Netzwerke zu identifizieren. Hierzu muss die Zeitreihe zunÃ¤chst in unterschiedliche Zeitabschnitte aufgeteilt werden. Je nach Datensatz kann hierbei ein unterschiedliches Intervall zum aufsplitten der Zeitreihe gewÃ¤hlt werden. Insofern die Zeitreihe eine PeriodizitÃ¤t aufweist, kann diese bestimmt und als Intervall festgelegt werden.
AnschlieÃend muss jeder Zeitabschnitt in ein Netzwerk umgewandelt werden. Dies wird getan indem die Ähnlichkeit zwischen allen Elementen des Zeitabschnitts berechnet wird. Die Ähnlichkeiten bilden anschlieÃend die Gewichte zwischen den Elementen im Netzwerk.

$$D_{ij} = \left(\sum_k \left| v_k^i - v_k^j \right|^p \right)^{1/p}$$

Das Ergebnis hiervon bildet eine Adjazenzmatrix. Diese Adjazensmatrix muss anschlieÃend in ein Textfile geschrieben werden, sodass die Daten vom Netismile Algorithmus genutzt werden kÃ¶nnen. Dazu wird fÃ¼r jede Kante eine neue Zeile im File generiert, in dieser wird zunÃ¤chst der Ursprungsknoten, dann der Zielknoten und anschlieÃend die Gewichtung aufgefÃ¼hrt. FÃ¼r jeden Zeitabschnitt muss ein einzelner File angelegt werden.

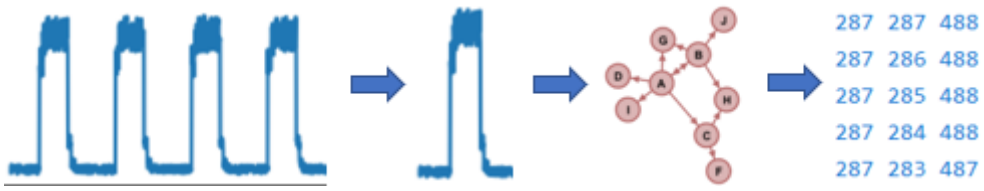

Abb. 2.1: Umwandlung einer Zeitreihe in Netzwerk

todo: Das Netzwerk aus der Grafik noch abÃ¤ndern

2.2 MIDAS

Die Umwandlung in eine Zeitreihe funktioniert Ã¤hnlich wie in Kap. 2.1. Lediglich das Ãbergabeformat an den Algorithmus unterscheidet sich. Hierbei muss nicht fÃ¼r jeden Zeitabschnitt eine neue Datei angelegt werden. Anstatt dessen muss bei jeder Kante angegeben werden in welchem Abschnitt sie stattfindet. AuÃerdem kann an den Algorithmus nicht direkt eine Gewichtung der Kante Ã¼bergeben werden. Es muss anstatt eine Gewichtung zu Ã¼bergeben einfach 10 mal die gleiche Kante Ã¼bergeben werden. Je nachdem wie hoch die Gewichtung der Kante ist.

```
284 174 13
284 174 13
284 174 13
284 175 13
284 175 13
284 175 13
284 175 13
284 175 13
284 175 13
284 175 13
284 175 13
284 176 13
284 176 13
```

Abb. 2.2: Datensatz Midas

todo: Vielleicht kleineren Auszug aus Datensatz verwenden

2.3 MIDAS-R

todo: Hab hier das mit der Hauptkomponentenzerlegung gemacht. Wenn es Ergebnisse hierfÃ¼r gibt. Kann ich das hier noch erklÃ¤ren

3 Netsimile

todo: In diesem Kapitel werden grundlegende Themen behandelt, die im Rahmen des Forschungsprojekts zum Verständnis der Ausreißer-Erkennung in Graphen gedient haben.

3.1 Grundlagen

todo: Einführung in den Algorithmus

3.1.1 Canberra Distance

Einführung

todo: Stichworte sammeln

4 MIDAS

todo: In diesem Kapitel werden grundlegende Themen behandelt, die im Rahmen des Forschungsprojekts zum Verständnis der Ausreißer-Erkennung in Graphen gedient haben.

Erst erklären wie der MIDAS funktioniert. Und zum Laufen gebracht mit Graphen über die Zeit ENRON & DARPA. Im Anschluss auf Zeitreihendaten angewendet.

4.1 Grundlagen

todo: Einführung in den Algorithmus

4.1.1 Count-min Sketch

Einführung

todo: Stichworte sammeln

5 Statische Algorithmen

todo: Labels für die einzelnen Texte umbenennen Es ist mit dieser Art von Algorithmen möglich Ausreißer in einer vollständigen und abgeschlossenen Zeitreihe zu identifizieren. Es werden zwei Algorithmen vorgestellt, ein auf Percolation basierender Algorithmus und ein auf IsoMap basierender Algorithmus. Beide Algorithmen wurden dazu entwickelt Ausreißer in unterschiedlichen Typen von Datensätzen zu erkennen (z.B. Videos, Bilder, Netzwerke). Voraussetzung hierfür ist lediglich, dass eine Distanz zwischen unterschiedlichen Elementen des Datensatzes berechnet werden kann (vgl. Amil et al. 2019, S. 2). Im Folgenden werden die Algorithmen, hinsichtlich ihrer Fähigkeit Ausreißer in Zeitreihen zu identifizieren evaluiert.

Für beide Algorithmen gilt, dass die Zeitreihe zunächst in ein Netzwerk umgewandelt werden muss. Hierzu die Formel, welche hierzu verwendet wird:

$$a = b \tag{5.1}$$

$$D_{ij} = \left(\sum_k \left| v_k^i - v_k^j \right|^p \right)^{1/p}$$

Formel 1 gibt an, wie ein Element ij der Distanzmatrix berechnet wird. Insofern für p gleich zwei eingesetzt wird, wird zwischen den Elementen die euklidische Distanz berechnet. Ein Element der Zeitreihe kann aus mehreren Werten bestehen z.B. bei multivariaten Zeitreihen. Die berechnete Distanzmatrix bildet ein Netzwerk, dabei bilden die Zeitreihen Elemente die Knoten und die Distanzen zwischen ihnen, die Gewichte (vgl. Amil et al. 2019, S. 2-3).

5.1 IsoMap Basierter Algorithmus

Der Grundgedanke hinter diesem Ansatz ist, dass Informationen über Ausreißer bei der Reduzierung der Dimensionalität mit dem IsoMap Algorithmus verloren gehen. Insofern versucht wird, die Informationen zu rekonstruieren und mit der ursprünglichen Matrix vergleicht, können große Abweichungen bei Ausreißer Elementen festgestellt werden (vgl. Amil et al. 2019, S. 3).

5.1.1 IsoMap

Beim IsoMap handelt es sich um einen Algorithmus zur nichtlinearen Dimensionsreduktion. Zunächst werden beim IsoMap die Nachbarn eines jeden Punktes über K-Nearest Neighbor bestimmt. Anschließend wird jeder Punkt mit den gefundenen Nachbarn verknüpft, wodurch ein neuer Körper entsteht. Daraufhin wird eine neue Distanzmatrix auf dem entstandenen Körper berechnet. Diese Matrix kann auch als geodätische Distanzmatrix bezeichnet werden und wird im weiteren Verlauf des Algorithmus benötigt. Der Zweck des Ablaufs ist es das nichtlineare

Zusammenhänge in der anschließenden Dimensionsreduktion erhalten bleiben. Die Dimensionsreduktion erfolgt anschließend über Eigenvektor ? (vgl. Tenenbaum et al. 2000, S. 3). todo: Noch nach Seite für Quelle suchen

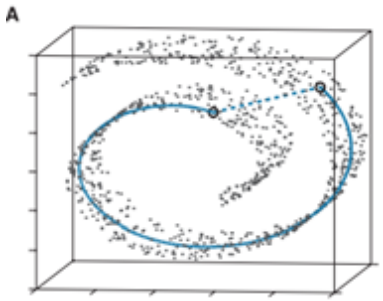

Abb. 5.1: Funktionsweiße IsoMap

5.1.2 IsoMap Basierter Algorithmus

Mithilfe des IsoMap Algorithmus wurden neue Features berechnet. Im nächsten Schritt wird versucht aus diesen Features die ursprüngliche Distanzmatrix zu rekonstruieren. Nun kann die Distanzmatrix aus Kap. 5.1.1 mit dieser Distanzmatrix verglichen werden. Dazu wird die Pearson Korrelation zwischen den jeweiligen Vektoren der Matrizen berechnet. Für Ausreißer wird erwartet, dass die Ähnlichkeit sehr niedrig ist, da die Informationen über sie bei der Reduktion verloren gehen (vgl. Amil et al. 2019, S. 3).

5.1.3 Implementierung

Da für die Implementierung des Algorithmus viele Berechnungen mit Matrizen durchgeführt werden müssen, wurde hierzu auf Python/Numpy zurückgegriffen. Für den IsoMap Algorithmus existierte eine sehr gute Implementierung in SciKitLearn, deshalb wurde auf diese zurückgegriffen. An den Algorithmus können verschieden Parameter übergeben werden, es handelt sich hierbei um dieselben Parameter, welche auch an den IsoMap Algorithmus übergeben werden können.

5.1.4 Ergebnisse

uqbcqwiu

5.2 Perculation

Bei diesem Algorithmus werden schrittweiße die Kanten mit den höchsten Gewichten aus der Distanzmatrix entfernt. Ziel dieses Prozesses ist es Ausreißer vom Hauptcluster zu trennen. Dabei kann davon ausgegangen werden, dass Ausreißer höhere Kantengewichte zu ihren Nachbarn aufweisen und deshalb schneller separiert werden. Sobald ein Knoten komplett separiert ist, wird ihm ein Ausreißer Score zugeordnet. Der Wert des Ausreißer Scores wird über die zuletzt entfernte Kante des Knoten definiert (vgl. Amil et al. 2019, S. 3).

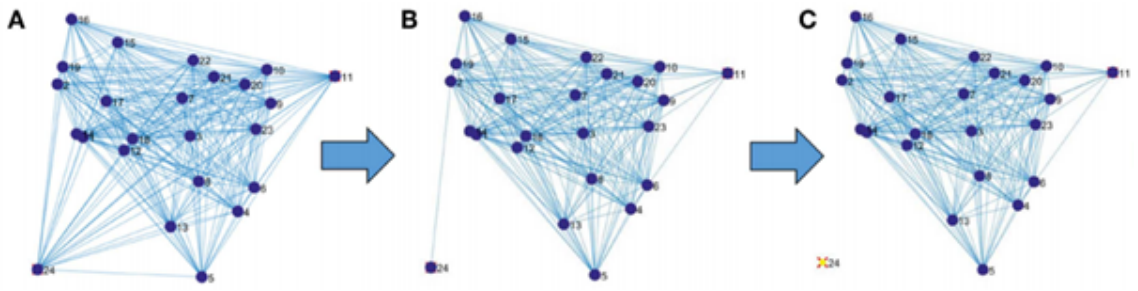

Abb. 5.2: Ablauf Perculation basierter Algorithmus

5.2.1 Implementierung

Aus denselben Gründen wie in Kap. 5.1.3 erläutert, wurde für die Implementierung auf Python/Numpy zurückgegriffen. Da die Distanzmatrix sehr umfangreich werden kann wurden einige Veränderungen an dem Algorithmus vorgenommen, um ihn Performanter zu machen. Eine Modifikation, die vorgenommen wurde, ist das die Kanten nicht einzeln, sondern in Gruppen entfernt werden. Dadurch muss seltener überprüft werden, ob ein Knoten mittlerweile komplett isoliert ist. Außerdem wurde ein Abbruchkriterium implementiert, bei welchem der Algorithmus angehalten wird sobald eine bestimmte Prozentzahl an Kanten entfernt wurde. Dies hat keine Auswirkungen auf die Qualität der Ausreißer Erkennung, da Ausreiser üblicherweise bereits zu beginn des Algorithmus isoliert werden. Der Algorithmus berechnet für jedes Element der Zeitreihe einen Ausreißer Score. Allerdings können die Ausreißer Scores sehr stark schwanken. Deshalb ist es schwierig Ausreißer zu identifizieren, welche sich über mehrere Zeitschritte erstrecken, da kein kontinuierliches Ansteigen des Scores beobachtet werden kann. Eine Möglichkeit, um diese Art der Ausreißer trotzdem zu identifizieren, ist es ein Sliding Window Verfahren einzusetzen. Dabei wird der Ausreißer Score für jedes Element neu berechnet, indem ein Mittelwert über die Zeitpunkte vor einem und nach einem Element gebildet wird. Dadurch werden die Schwankungen im Ausreißer Score abgemildert. Prinzipiell ist der Algorithmus parameterfrei, durch die Veränderungen kann jedoch die Größe des Sliding Window als Parameter übergeben werden.

5.2.2 Ergebnisse

todo: Die Bilder vielleicht noch überarbeiten, sodass sie schöner aussehen. Vielleicht auch noch die Tabelle mit den Sternen rein machen. Vielleicht die beiden Graphiken zu einer Zusammenführen.

todo: Fragestellung: Inwieweit können vielleicht auch andere Datensätze in Graphen umgewandelt werden, sodass z.B. der Netismile darauf angewendet werden kann.

6 Kapitelname

6.1 Unterkapitel

6.1.1 Unterunterkapitel

Man kann mit den labels im Text bezug nehmen mit Kap. 6.1.1. Wenn man zitieren möchte, dann verwendet man am besten (vgl. Mihajlovic & Petkovic 2001, S. 1). Ich denke die Logik wird klar, wenn man es hier betrachtet. Anführungszeichen *im Fließtext* erfolgen mittels "Wort" manchmal benötigt man das space um ein Leerzeichen zu generieren.

Fette Überschrift

Für Bilder könnt ihr folgendes copy pasten. Die Bilder, die ihr einfügen möchtet, sollten im Ordner fig eingefügt werden. Hier sollte bei includegraphics der Name eingesetzt werden, caption steht für Bildunterschrift und das label muss individuell sein damit man darauf referenzieren kann.

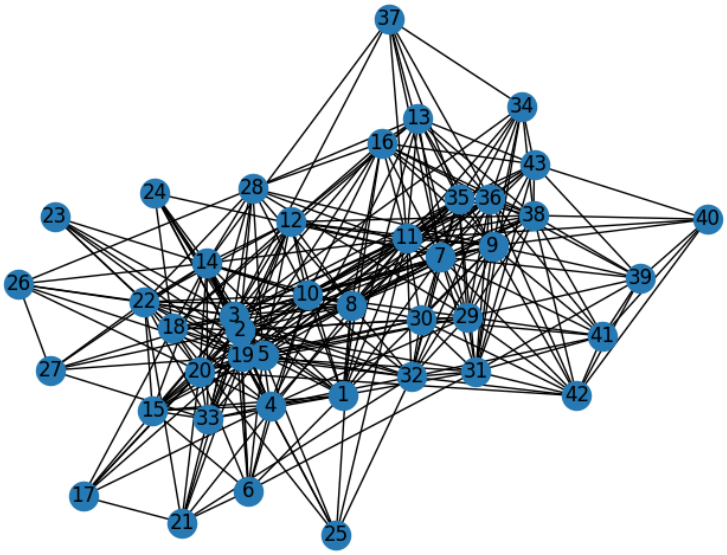

Abb. 6.1: Ego-Netzwerk des Datensatzes

Bei mehreren Bildern nebeneinander könnt ihr folgendes verwenden:

Wenn ihr to dos vermerken wollt, nutzt hierfür: todo: hier ein to do einfügen

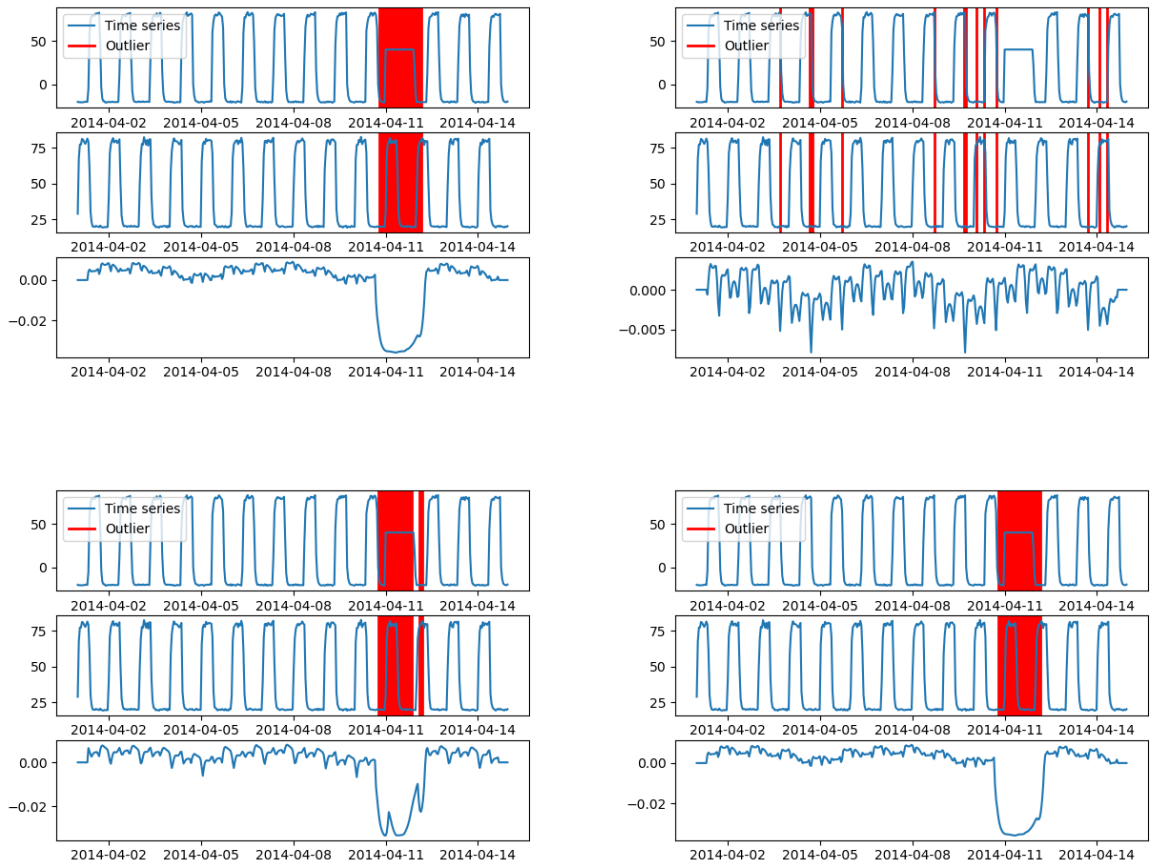

Abb. 6.2: Im Uhrzeigersinn: an erster Variablen ausgerichtet (Variante 1), an zweiter Variablen ausgerichtet (Variante 2), nicht ausgerichtet (Variante 3) und an allen Variablen ausgerichtet (Variante 4).

Wenn ihr etwas aufzählen wollt:

1. Verschaffen eines Überblicks über die existierenden Algorithmen zur Erkennung von Ausreißern in Graphen
2. Die Entwicklung eines Ausreißer-Scores für die zugrundeliegenden Algorithmen
3. Erste Anwendung der verwendeten Graphen-basierten Algorithmen auf Zeitreihen

Literaturverzeichnis

Amil, P., Almeira, N. & Masoller, C. (2019), 'Outlier mining methods based on graph structure analysis', *Frontiers in Physics* **7**, 194.
URL: *https://www.frontiersin.org/article/10.3389/fphy.2019.00194*

Mihajlovic, V. & Petkovic, M. (2001), *BEISPIELDynamic Bayesian Networks: A State of the Art*, Vol. TR-CTIT-34 of *CTIT Technical Report Series*, University of Twente, Netherlands. Imported from CTIT.

Tenenbaum, J. B., Silva, V. d. & Langford, J. C. (2000), 'A global geometric framework for nonlinear dimensionality reduction', *Science* **290**(5500), 2319–2323.
URL: *https://science.sciencemag.org/content/290/5500/2319*