Softwareprojekt: Rekonstruktion metrischer Graphen

Terese Haimberger, Lea Helmers, Mahmoud Kassem, Daniel Theus, Moritz Walter

Inhaltsverzeichnis

1 Einleitung: Rekonstruktion metrischer Graphen

Motivation und Problemstellung

Im Rahmen des Softwareprojekts „Anwendungen effizienter Algorithmen“ haben wir uns damit befasst, einen Algorithmus umzusetsen, der aus einer Punktmenge den zugrundeliegenden Graphen sowie dessen Metrik rekonstruiert. Dadurch soll Struktur in große Mengen geometrischer Daten gebracht werden, was deren Analyse und Weiterverarbeitung erleichtert. Zu verarbeitende Daten können Netzwerke im weitesten Sinne sein, wie beispielsweise GPS-Daten, Strom- und Nachrichtennetze oder astrologische Daten. Häufig enthalten diese Daten Rauschen oder Ausreißer und sind zudem im Allgemeinen sehr umfangreich. Ziel des Algorithmus ist es, die Datenmenge kompakt durch ihre wichtigsten Verzweigungen darzustellen, wodurch sie auf ihre wichtigen Aspekte reduziert wird. So kann eine einfachere Visualisierung und die weitere Analyse und Verarbeitung der Daten ermöglicht werden.

Der Algorithmus

Als Grundlage für unsere Arbeit diente uns ein Paper [**chenEa2012**], welches einen Algorithmus für die Rekonstruktion metrischer Graphen beschreibt und dessen Richtigkeit beweist. Die Eingabe besteht dabei aus einem metrischen Raum (Y, d_y), der aus den Rohdaten konstruiert wird und dem ein metrischer Graph (X, d_x) zugrundeliegt, den es zu rekonstruieren gilt. Ziel des Algorithmus ist es, diesen zugrundeliegenden Graphen durch einen metrischen Graphen $(\hat{X}, d_{\hat{x}})$ anzunähren und dabei weitestgehend die Abstände von (X, d_x) in $(\hat{X}, d_{\hat{x}})$ beizubehalten. Zusätzlich zu (Y, d_y) wird auch ein Parameter r übergeben, der in einigen Schritten des Algorithmus eine wichtige Rolle spielt. Die Erstellung des Graphen $(\hat{X}, d_{\hat{x}})$ wird nun wie folgt durchgeführt:

1. **Kanten- und Knotenpunkte bestimmen**
   Zunächst wird festgestellt, welche der in (Y, d_y) enthaltenen Punkte in $(\hat{X}, d_{\hat{x}})$ zu einer Kante und welche zu einem Knoten gehören werden. Dafür wird um alle im Eingabegraphen enthaltenen Punkte ein Kreisring gelegt, wobei der innere Kreis des Kreisrings den Radius r und der äußere den Radius $\frac{5}{3}r$ hat. Für die Punktmenge in diesem Kreisring wird anschließend der Rips-Vietoris-Graph mit dem Parameter $\frac{4}{3}r$ erstellt. Das bedeutet, dass alle Punkte, die nicht mehr als $\frac{4}{3}r$ voneinander entfernt sind, zu einer Zusammenhangskomponente gefasst werden. Daraufhin wird der Grad des jeweiligen Knoten bestimmt, indem die Zusammenhangskomponenten des Rips-Vietoris-Graphen gezählt werden. Enthält der Kreisring um einen Knoten zwei Zusammenhangskomponenten, hat er Grad zwei und es handelt sich um einen Punkt, der im rekonstruierten Graphen $(\hat{X}, d_{\hat{x}})$ zu einer Kante gehören wird. Daher erhält er das Label *edge point*. Ist der Grad ungleich zwei, wird der Punkt zu einem Knoten gehören und wird daher als *preliminary branch point* gekennzeichnet. Diese Markierung erhalten diese Punkte lediglich vorläufig (*preliminary*), da im Anschluss alle Punkte innerhalb eines Abstands von $2r$ von einem *preliminary branch point* als *branch point* gekennzeichnet werden. Hierbei werden auch die *preliminary branch points* zu *branch points*, womit sich die Punkte in (Y, d_y) nun in zwei Teilmengen $\mathbb{E}$ und $\mathbb{V}$ aufteilen lassen. In $\mathbb{E}$ sind dabei die *edge points* enthalten und in $\mathbb{V}$ die *branch points*. Der Vorgang wird mit Pseudocode in Algorithm 1 veranschaulicht.

2. **Struktur des Graphen rekonstruieren**
   Anschließend wird für die beiden Mengen $\mathbb{V}$ und $\mathbb{E}$ der Rips-Vietoris-Graph mit dem Parameter $2r$ erstellt. Die dabei entstehenden Zusammenhangskomponenten in $\mathbb{E}$ entsprechen nun den Kanten und jene in $\mathbb{V}$ den Knoten im Graphen $(\hat{X}, d_{\hat{x}})$. Um dessen Struktur zu vervollständigen

Algorithm 1 Kanten- und Knotenpunkte bestimmen

for all $y \in Y$ **do**
  $R \leftarrow C_{\frac{5}{3}r} \setminus C_r$
  $deg\,(y) \leftarrow$ # Zusammenhangskomponenten im Rips-Vietoris-Graphen$_{\frac{4}{3}r}\,(R)$
  if $deg(y) == 2$ **then**
    y erhält das Label edge point
  else
    y erhält das Label branch point
  end if
end for
for all $y \in Y$ **do**
  if y ist nicht weiter als $2r$ von einem preliminary branch point entfernt **then**
    y erhält das Label branch point
  end if
end for

müssen lediglich noch die Verbindungen zwischen Kanten und Knoten rekonstruiert werden. Dies geschieht, indem zwei Knoten aus $\mathbb{V}$ genau dann durch eine Kante $e \in \mathbb{E}$ verbunden werden, wenn sie in ihrer Zusammenhangskomponente Punkte haben, die zu Punkten der Zusammenhangskomponente von e einen kleineren Abstand als $2r$ haben.

3. **Metrik rekonstruieren**
   Schlussendlich müssen noch die Längen der Kanten bestimmt werden. Dafür wird jeder Kante in $\hat{X}$ als Länge der Durchmesser ihrer Zusammenhangskomponente, also der längste kürzeste Weg darin, zuzüglich $4r$ zugewiesen.

2 Organisation und Umsetzung des Algorithmus in Java

Organisation und Kommunikation

Da allen Studenten im Team die objektorientierte Programmiersprache Java geläufig war und sie uns als durchaus geeignet für die Aufgabe schien, haben wir uns entschlossen, unser Programm darin zu schreiben. Für das Arbeiten an unserer Software haben wir uns in drei verschiedene Gruppen aufgeteilt. Die eine Gruppe hat sich mit der Vorverarbeitung der Rohdaten beschäftigt, im Wesentlichen also aus den Rohdaten den metrische Raum (Y, d_y) für die Eingabe des Algorithmus konstruiert. Die zweite Gruppe hat sich mit der Implementierung des Algorithmus, genauer der Rekonstruktion von $(\hat{X}, d_{\hat{x}})$, in Java auseinandergesetzt, während die dritte für die Visualisierung des Graphen $(\hat{X}, d_{\hat{x}})$ zuständig war. Damit wir stets alle über die Arbeit unserer Teammitglieder informiert waren haben wir unsere Ergebnisse auf der Hosting-Plattform GitHub[1] gespeichert und stets aktualisiert. Um anstehende Aufgaben für alle zugänglich zu dokumentieren haben wir zusätzlich das für Projektmanagement bestimmte Anwendung Trello [2] genutzt, bei man ein virtuelles Notizbrett erstellen kann. Hier konnte jeder neue Aufgaben einstellen und sich dafür als Bearbeiter eintragen. Zusätzlich konnten wir dort den Stand der Arbeit an diesen Aufgaben dokumentieren, wodurch stets jeder darüber informiert war, woran die anderen gerade arbeiten.

Zielsetzung

Einschränkung auf GPS und Buchstaben

[1] `github.com`
[2] `trello.com`

3 Arbeit in den einzelnen Gruppen

3.1 Preprocessing

3.2 Reconstruction

3.3 Visualisierung

4 Testen?

5 Schwierigkeiten und Verbesserungsvorschläge

6 Zusammenfassung