AGT Summary

April 24, 2024

Contents

1 Netzwerke und Zentralität

1.1 Charakterisierung der wichtigsten Ecke

Hierfür gibt es mehrere Möglichkeiten

- größter Einfluss
- wichtig für Informationsfluss

Die Wichtigkeit wird mit einem Zentralitätsmaß gemessen.

Definition 1.1.1. Zentralitätsmaße sind sehr unterschiedlich. Es muss nur erfüllt sein, dass bei einem Sterngraphen das Zentrum das größte Zentralitätsmaß erhält. Möglich sind Bewertungen nach

1. dem Maximalgrad (*degree centrality*)
2. der durchschnittlichen Entfernung zu anderen Ecken (*closeness centrality*) (bzw. der Kehrwert davon)
3. der Anzahl der Komponenten, die mit dieser Ecke verbunden sind (*betweenness centrality*). Dafür sei $\sigma_{s,t}$ die Anzahl der kürzesten $s-t$-Wege. $\sigma_{s,t}(v)$ für $v \neq s,t$ ist dann die Anzahl der kürzesten $s-t$-Wege, die durch v gehen. Damit gilt

$$betweenness(v) = \sum_{s,t \in V()G \setminus \{v\}} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$$

1.2 Berechnung der Zentralitätsmaße

Wir führen nur die Berechnung der betweenness ein. Die anderen beide Maße sind sehr einfach.
Der Algorithmus zur Berechnung von $\sigma_{s,t}$ ist an Dijkstra angelehnt. Beginnend mit s wird die Anzahl der Nachbarn von s bestimmt. Anschließend die Anzahl der Knoten mit Abstand 2 usw. Um die Komplexität der Algorithmen zu bestimmen, werden im Folgenden einige Annahmen getroffen:

1. Kotenadjazenz kann in $\mathcal{O}(1)$ bestimmt werden
2. Kanteninzidenz kann in $\mathcal{O}(1)$ bestimmt werden
3. die Nachbarschaft eines Knoten wird in $\mathcal{O}(1)$ pro Knoten bestimmt
4. die zu einem Knoten inzidenten Kanten können in $\mathcal{O}(1)$ pro Kante bestimmt werden

5. alle elementaren Operationen (z.B. Kante löschen) in $\mathcal{O}(1)$.

Auf diese Weise kann man leicht sehen, dass die Laufzeit zur Berechnung von $\sigma_{s,t}$ für alle s, t in $\mathcal{O}(n \cdot m)$ implementiert werden kann. Wir nehmen nun an, $\sigma_{s,t}$ sei bekannt und wir definieren

$$\rho_s(v) = \sum_{t \neq v} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$$

Kennt man nun alle $\rho_s(v)$, dann ist

$$betweenness(v) = \frac{1}{2} \sum_{s \neq v} \rho_s(v)$$

Lemma 1.2.1. *Sei v ein Knoten mit Distanz mindestens $d \geq 1$ zu s und sei L die Menge der Knoten mit Distanz $d+1$ zu s. Dann ist*

$$\rho_s(v) = \sum_{w \in L \cap N(v)} \frac{\sigma_{s,v}}{\sigma_{s,w}} (1 + \rho_s(w)$$

Mit dieser Überlegung lässt sich ein Algorithmus finden, der die *betweenness* jedes Knotens in $\mathcal{O}(mn)$ berechnet.

1.3 Random Walks auf Graphen

Wir wählen zunächst einen Startknoten v_0 bezüglich einer Wahrscheinlichkeitsverteilung $\pi^{(0)}$. Anschließend wird mit Gleichverteilung ein zufälliger Nachbar v_1 von v_0 gezogen usw. Wenn man sich nun die Frage stellt, was die Wahrscheinlichkeit ist, dass der erste gezogenen Knoten (d.h. v_1) gleich dem Knoten u ist, dann entspricht das der Wahrscheinlichkeit, dass u ein Nachbar von v_0 ist mal der Wahrscheinlichkeit, dass anschließend u gezogen wird. Da der zweite Schritt gleichverteilt ist, ergibt sich

$$\pi_u(1) = \sum_{v \in N(u)} \pi_v^{(0)} \cdot \frac{1}{d(v)}$$

Das wird geschrieben als Transition Matrix mit

$$T_{uv} = \begin{cases} \frac{1}{d(v)}, & \text{wenn } uv \in E \\ 0, & \text{sonst} \end{cases}$$

Schreibt man die Wahrscheinlichkeitsverteilung $\pi^{(0)}$ einfach als Vektor, dessen Komponenten zu 1 addieren, ergibt sich

$$\pi^{(n+1)} = T\pi^{(n)}$$

für $n \geq 0$. Um zu überprüfen, ob ein bestimmter Knoten im Random Walk jemals besucht wird, muss die Grenzwertverteilung bestimmt werden

$$\pi^* = \lim_{k\to\infty} T^k \pi^{(0)}$$

Existiert π^*, dann ist π^k eine Cauchy-Folge und man kann leicht sehen, dass $T\pi^* = \pi^*$. Dann ist π^* also ein Eigenvektor zum Eigenwert 1 von T.

Theorem 1.3.1 (Perron-Frobenius)**.** *Sei $A \in \mathbb{R}^{n\times n}$ sodass $\exists k \in \mathbb{N}$ mit $A^k_{ij} > 0$ für alle $i, j \in [n]$. Dann gibt es einen eindeutigen Eigenwert λ^* mit größtem Betrag. Wenn $\lambda^* > 0$ gibt es einen positiven Eigenvektor v^* zu λ^* und alle anderen Eigenvektoren zu λ^* sind Vielfache von v^*. Ist außerdem $\lambda^* = 1$, dann konvergiert $v^{(k+1)} = Av^{(k)}$ gegen ein Vielfaches von v^* für alle positiven Startvektoren $v^{(0)} > 0$.*

Damit kann man sich überzeugen, dass T Eigenwert 1 hat und dass das der größte Eigenwert ist. Außerdem erfüllt T die Eigenschaft aus obigem Theorem, wenn G zusammenhängend und nicht bipartit ist.

1.4 Eigenwert Zentralität

Für einen Knoten v verwenden wir wieder die Matrix T und nehmen $\pi^* > 0$ als den Eigenvektor zum Eigenwert 1 mit $||\pi^*|| = 1$. Der Eintrag π^*_v ist dann die Eigenwert Zentralität von v.
Das Problem dieses Zentralitätsbegriffs ist, dass man den Eigenwert recht leicht erraten kann. Betrachte dazu

$$\overline{\pi}_v = \frac{d(v)}{2\,|E|}\ \forall v \in V$$

Es ist leicht zu sehen, dass dieser Vektor ein Eigenvektor zum Eigenwert 1 ist. Es folgt also, dass wir einen neuen Begriff haben, der aber sehr ähnlich zur *degree centrality* ist. In gerichteten Graphen ist der Begriff ein wenig hilfreicher. Der wichtigste Unterschied ist die modifizierte Matrix T mit

$$T_{vu} = \begin{cases} \frac{1}{d^+(u)}, & \text{wenn es eine Kante von } u \text{ nach } v \text{ gibt} \\ 0, & \text{sonst} \end{cases}$$

Das größere Problem sind Senken (d.h. Knoten v mit ausgehendem Grad $d^+(v) = 0$). Das kann gelöst werden, indem der Prozess neugestartet wird (d.h. eine Kante zu jedem anderen Knoten eingeführt wird).

1.5 PageRank

PageRank ist der Suchalgorithmus von Google. Er funktioniert in den folgenden Schritte, die sehr ähnlich zum Eigenwertzentralität sind

1. Wähle unter Gleichverteilung einen Startknoten
2. Mit Wahrscheinlichkeit $1-\alpha$ (α konstant) wähle einen Nachbarn und gehe dorthin.
3. Mit Wahrscheinlichkeit α wähle einen neuen Startknoten.

Die Transitionsmatrix definiert nun eine andere Matrix

$$P = (1-\alpha)T + \frac{\alpha}{n}J$$

wobei J die Matrix mit nur 1 Einträgen ist. Es ergibt sich der Prozess

$$\pi^{(k+1)} = P\pi^{(k)}$$

Da P positiv ist, ergibt Theorem 1.3.1 die Existenz der Grenzwertverteilung p_v^*. Es gilt $\text{PageRank}(v) = \pi_v^*$. Da der erste Teil von P *sparse* ist, kann die Iteration relativ effizient durchgeführt werden.

2 Clustering

Wie führen zunächst den Begriff des Clustering-Koeffizienten ein. Sei $v \in V$. dann ist

$$C(v) = \frac{|E_G[N(v)]|}{\binom{|N(v)|}{2}}$$

Der durchschnittliche Clustering-Koeffizient ist dann

$$C(G) = \frac{1}{|V|}\sum_{v\in V} C(v)$$

Ein Zufallsgraph mit Kantenwahrscheinlichkeit p hat im Erwartungswert eine Kantendichte $\frac{|E|}{\binom{n}{2}}$ von ungefähr p. Der Clustering-Koeffizient ist ebenso ungefähr p.

2.1 Berechnung des Clustering Koeffizienten

Es ist leicht zu sehen, dass man den Clustering Koeffizienten eines einzelnen Knotens v in $\mathcal{O}(d(v)^2)$ berechnen kann. Um den durchschnittlichen Wert zu bestimmen genügt daher eine Laufzeit von $\mathcal{O}(\sum_{v \in V(G)} d(v)^2)$. Die Summe lässt sich nach oben abschätzen durch $2mn$ wodurch die Laufzeit bei $\mathcal{O}(2mn)$ liegt.
Ist $d(v)$ klein, so ist der Algorithmus sehr effizient, aber ist $d(v) >> \sqrt{m}$ so lässt sich eine Verbesserung erzielen, indem für jede Kante uw überprüft wird, ob u, v, w ein Dreieck bilden. Kombiniert man diese beiden Überlegungen zu einem Algorithmus mittels einer Fallunterscheidung, so erhält man einen Algorithmus zur Berechnung des durchschnittlichen Clustering Koeffizienten mit einer Laufzeit von $\mathcal{O}(m^{\frac{3}{2}})$.
Es gibt außerdem einen randomisierten Ansatz für die Schätzung des durchschnittlichen Clustering Koeffizienten auf Graphen mit Minimalgrad mindestens 2. Hierfür wird zunächst eine Konstante $k \in \mathbb{N}$ festgelegt. Anschließend werden nacheinander k Knoten $v_1, ..., v_k$ zufällig gezogen und aus $N(v_i)$ werden jeweils zwei Nachbarn u_i, w_i zufällig gezogen. Es wird gezählt, wie viele dieser Nachbarn der k Knoten mit v_i ein Dreieck aufspannen und diese Anzahl anschließend durch k geteilt.

Theorem 2.1.1. *Sei $\varepsilon > 0, \delta > 0$ und $k = \lceil \ln\left(\frac{2}{\delta}\right) / (2\varepsilon^2) \rceil$. Dann hat der Algorithmus eine Laufzeit von $\mathcal{O}(\ln\left(\frac{1}{\delta}\right) / \varepsilon^2 \cdot \ln n)$ und mit Wahrscheinlichkeit mindestens $1 - \delta$ unterscheidet sich der berechnete Wert um maximal ε vom tatsächlichen Wert.*