ulm university universität ulm

Temporal Graph Embedding

Justin Mücke | 18.02.21 | Universität Ulm, Seminar Data-Science

Inhaltsverzeichnis

Temporale Graphen

- Graph verändert sich über Zeit
- Darstellung durch $G = (V, E)$
- Evolution gegeben durch: $G_T = \{G_1, \ldots, G_T\}$

Einbettung statischer Graphen

- Graphen als mathematische Struktur sehr komplex
- überführung in Menge von Vektoren
- beibehalten der Graphtopologie, Knoten-Knoten Beziehung
- 2 Kategorien:
  - Node Embedding
  - Graph Embedding

word2vec

- Ansatz aus der Sprachverarbeitung
- Input: Text $T = (w_1, w_2, \ldots, w_n)$
- Berechnung der Wahrscheinlichkeit, dass zwei Wörter benachbart sind
- Nachbarn: $\{w_i | i \in (c - \epsilon, c + \epsilon)\}$
- Output: Menge von Vektoren

DeepWalk

- Weiterführung des word2vec Ansatz
- 3 Steps:
  - Sampling: Ausführen von RandomWalks
  - berechnung des nächsten Knotens:

$$P(v_i = x | v_{i-1} = y) = \begin{cases} \frac{\pi_{xy}}{Z} & \text{if}(x, y) \in E \\ 0 & \text{otherwise} \end{cases}$$

  - Erzeugen textliche Struktur auf welcher word2vec angewandt werden kann.

node2vec

- weiterführung des DeepWalk Ansatz
- Sampling über RandomWalks
- Nodes erhalten bias α
- $\alpha_{pq}(xy) = \begin{cases} \frac{1}{p} & \text{if } d_{xy} = 0 \\ 1 & \text{if } d_{xy} = 1 \\ \frac{1}{q} & \text{if } d_{xy} = 2 \end{cases}$
- $q < 1 \rightarrow$ eher zu fernen Knoten
- $q > 1 \rightarrow$ eher zu nahen Knoten

tNodeEmbed

- Node Level Algorithmus
- Kann in drei Teile eingeteilt werden
  - Initialization
  - Node alignment
  - Finalization

Initialization

- Erstellen eines repräsentativen Vektors für alle Knoten
- Nutzung des node2vec Algorithmus

Node Alignment

- Keine Garantie, dass bei der Einbettung die Achsen der Graphen überdecken
- Orthogonale Transformation zum Ausrichten
- Ausrichten der Graphen Q_t und $Q_t + 1$ mit $Q'_t = RQ_t$
- Annäherung durch: $R_{t+1} = \underset{Rs.t.R^T R=I}{\arg\min} \|RQ_{t+1} - Q_t\|$
- Iterativer Vorgang über ganze Evolution

Finalization

- Algorithmus kann sowohl für Node klassifizierung als auch für Link prediction genutzt werden
- Einziger unterschied ist die genutzte Verlustfunktion
- Node classification:
  $L_{task} = -\sum_{v \in V} \log Pr(class(v)|f_T(v))$
- Link prediction:
  $L_{task} = -\sum_{v_1, v_2 \in V} \log Pr((v_1, v_2) \in E|g(f_T(v_1), f_T(v_2)))$
- f_T rekursiv definiert als $f_{t+1} = \sigma(Af_t(v) + BQ_tR_tv)$
- Embedding nun definiert als
  $L = \min_{A,B,Q_1,\ldots Q_T,R2,\ldots,RT} L_{task}$

tdGraphEmbed

- Graph Level Algorithmus
- Ziel ist es, Graph in einen d-dimensionalen Raum $\mathbb{R}^d$ einzubetten.
- Erstellung einer Mappingfunktion
- Kann ebenfalls in drei Teile gegliedert werden
  - Random Walks
  - Context Nodes
  - Optimization

Random Walks

- Algorithmus nutzt Random Walks
- Berechnung der Wahrscheinlichkeit, dass Graph durch die random Walks beschrieben widespread
- Wahrscheinlichkeit definiert als: $Pr(G_t | W_{v_1}, W_{v_2}, \ldots, W_{v_k}))$
- Random Walks werden konkateniert um ein Dokument zu formen

Context Nodes

- Defined as: $N_s(v_i^t) = \{v_{i-\omega}^t, \ldots, v_{i+\omega}^t\}$.
- Mit $\log p(v_i^t | N_s(v_i^t), G_t)$ können die nächsten Nodes in random Walks vorhergesagt werden
- Ziel: lernen einer repräsentation ϕ
  - $\phi(v_i^t)$: Mapping einer Node
  - $\phi(G_t)$: Mapping des Graphen
- Berechnung von $p(v_i^t | N_s(v_i^t), G_t)$ mit Mapping:

$$p(V_i^t | N_s(v_i^t), G_t) = \frac{e^{\phi(v_i^t) \cdot h}}{\sum_{j \in V_t} e^{\phi(v_j^t) \cdot h}}$$

Optimization

- Output berechnung kann vereinfacht werden zu:
  $$\max_{\phi} \sum_{t \in T} \sum_{W \in G_t} \sum_{v_i^t} -log \sum_{j \in V_t} e^{\phi(v_j^t) \cdot h} \quad + \phi(v_i^t \cdot h)$$
- Zur Berechnung kann ein neuronales Netz genutzt werden
- Da dies sehr resourcen kostig werden kann wird negative sampling genutzt
- Zur weiteren effizientsteigerung werden immer nur Knoten betrachtet, bei denen sich etwa verändert hat.

Anwendungen

- Node Classification
- Link prediction
- Clustering
- Similarity
- Anomaly detection