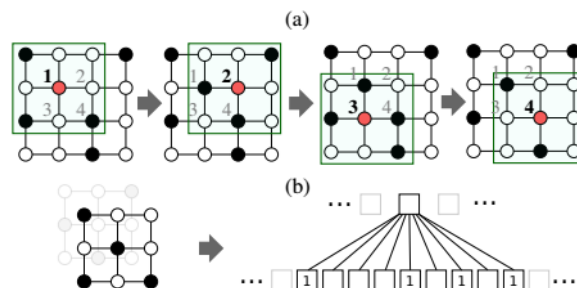


1 Convolutional neural networks (CNN) für Graphen

1.1 Einleitung

- Anwendungsfälle:
 1. Aus einer Menge von Graphen soll eine Funktion für Klassifizierungs- oder Regressionsprobleme gelernt werden, die auf nicht bekannte Graphen angewendet werden kann
 2. lerne Graph-Repräsentationen, um auf Graph-Eigenschaften (fehlende Kanten, Knodeigenschaften) unbekannter Graphen zu schließen
- Graphrepräsentation:
 - Graphen können gerichtet oder ungerichtet sein
 - Graphen können zyklisch sein
 - Graphen können mehrere unterschiedliche Kantentypen besitzen (mehrere Perceptive-Field-Layer)
 - Graphen können mehrere diskrete oder kontinuierliche Werte an ihren Knoten haben
- Methode berechnet lokal verbundene Nachbarschaften der Graphen und benutzt sie als die *Receptive Fields* des CNN
- die Methode kann für Graphen mit gewichteten Kanten erweitert werden



- Idee: repräsentiere Bilder als Graph
 - ein Bild kann als Graph repräsentiert werden, indem die Knoten jeweils einen Pixel repräsentieren und es eine Kante zwischen zwei Knoten gibt, wenn deren Pixel benachbart sind
 - die lokale Nachbarschaft eines Pixels wird repräsentiert als ein Quadrat um den Punkt (hier 3×3)
 - Aus der Nachbarschaft kann ein Merkmal ermittelt werden
- üblicherweise gibt es keine räumliche Anordnung einer Graph-Repräsentation
- Probleme:

1. Welche Nachbarschaften um welche Knoten und in welcher Reihenfolge bilden die Receptive Fields?
 2. Wie können die einzelnen Nachbarschafts-Graphen in einem Vektor repräsentiert werden (Normalisierung)?
- Verfahren:
 1. bestimme eine Knoten-Auswahl inklusive Reihenfolge
 2. bestimme den Nachbarschafts-Graphen um diesen Knoten mit genau k Knoten
 3. normalisiere die Nachbarschafts-Graphen
 4. füttere sie in ein CNN

1.2 Grundlagen

- Graph $G = (V, E)$ mit $V = \{v_1, \dots, v_n\}$ und $E \subseteq V \times V$, wobei n Anzahl der Knoten und m Anzahl der Kanten
- Adjazenzmatrix A mit Größe $n \times n$, wobei $A_{i,j} = 1$, falls eine Kante von v_i nach v_j existiert (sonst 0) $\Rightarrow v_i$ und v_j sind adjazent
- ein Weg ist eine Sequenz von Knoten, bei der benachbarte Knoten adjazent sind
- $d(u, v)$ beschreibt die minimale Distanz zwischen von u nach v
- $N_1(v)$ beschreibt die 1-Nachbarschaft um einen Knoten, d.h. alle Knoten die adjazent sind zu v

1.2.1 Beschriftung und Partitionierung

- eine Graph-Beschriftung $l : V \rightarrow S$ bildet einen Knoten auf eine sortierbare Einheit ab
- induziert ein *Ranking* $r : V \rightarrow \{1, \dots, |V|\}$ mit $r(u) < r(v)$ genau dann, wenn $l(u) > l(v)$
- falls l injektiv, dann gibt es eine totale Ordnung der Knoten in G und eine eindeutige Adjazenzmatrix A^l , bei der die Knoten die Position $r(v)$ haben
- eine Graph-Beschriftung induziert eine Partitionierung $\{V_1, \dots, V_k\}$ mit $u, v \in V_i$ falls $l(u) = l(v)$
- Metriken:
 - Anteil an kürzesten Wegen von u zu v (*Betweenness centrality*)
 - Grad der Knoten (Anzahl adjazenter Knoten)
 - ...

1.3 Lernen von Graphen


1.3.1 Knotenauswahl

- Auswahl an Knoten, für die ein Receptive Field erstellt werden soll
 - Gegeben: Graph-Beschreibung l , Abstand s , Anzahl w an Receptive Fields
1. sortiere die Knoten auf Basis von l
 2. iteriere über die sortierte Knotenmenge mit Abständen s , bis w Knoten ausgewählt wurden

1.3.2 Nachbarschaftssuche

- Gegeben: Knoten v , Größe k des Receptive Fields
1. setze initiale Knotenmenge N auf v
 2. wiederhole bis $|N| > k$:
 - a) berechne für alle Knoten i in N die Nachbarschaften $N_1(i)$ und füge sie zu N hinzu
- Bemerkung: im Allgemein gilt $|N| \neq k$

1.3.3 Normalisierung

- Gegeben: Menge von Graphen \mathcal{G} mit k Knoten, Distanzmetriken für Matrizen d_A und Graphen d_G
- Optimierungsproblem: $\min_l \sum_{G \in \mathcal{G}} \sum_{G' \in \mathcal{G}} (d_A(A^l(G), A^l(G')) - d_g(G, G'))$
- 

1.4 Auswertung

- CNNs mit Bildern können identisch über CNNs mit Graphen dargestellt werden
- Methode funktioniert teilweise deutlich besser als State-of-the-Art Graph-Kerne (z.B. bei Klasifizierungsproblemen)

1.5 Zukünftige Arbeiten

- gewichtete Kanten (oder allgemeiner Graphen mit Kanteneigenschaften)
- Graphen auf andere Netze übertragen, z.B. RNNs
- kombiniere unterschiedliche Receptive Field-Größen

2 Superpixel

- *Superpixel*: Menge von n Pixeln $S_i = \{t_1, \dots, t_n\}$, wobei $t_i \in \{1, \dots, N\}$ einen Pixel beschreibt
- Menge von Superpixeln: $S = \{S_1, \dots, S_m\}$, sodass $S_i \cap S_j = \emptyset$ für alle i, j und $\cup_i S_i = \cup_j t_j$
- Nachbarschaft: $(S_i, S_j) \in \mathcal{N}$, wenn S_i und S_j räumlich verbunden sind
- \Rightarrow Superpixel bieten eine Möglichkeit, die Größe des Problems zu minimieren
- \Rightarrow Superpixel haben aber folglich einen bestimmten Fehlergrad
- \Rightarrow finde den besten Ausgleich zwischen Größe und Fehlergrad

2.1 CNNs auf Superpixeln

-

wie kann
man su-
perpi-
xel in
ein cnn
füttern?

3 Graph-Kerne

-