Lernen mit Graphen: Kern- und neuronale Methoden[1]

Christopher Morris[2]

Abstract: Die vorliegende Arbeit befasst sich mit (überwachter) Graphklassifikation, d. h. mithilfe einer Menge von bereits klassifizierten Beispielgraphen wird ein Modell so trainiert, dass es die Klassen von bisher ungesehenen Graphen vorhersagen kann. Im ersten Teil dieser Arbeit stellen wir Kernmethoden für Graphen vor. Insbesondere stellen wir skalierbare Kerne vor, die mit kontinuierlichen Knoten- und Kantenbeschriftungen umgehen können. Ferner stellen wir einen Graphkern vor, der globale Grapheigenschaften berücksichtigen kann, die von anderen Graphkernen nicht erfasst werden. Zu diesem Zweck schlagen wir eine lokale Version des k-dimensionalen Weisfeiler-Leman-Algorithmus vor, der eine bekannte Heuristik für das Graph-Isomorphie-Problem ist. Wir zeigen, dass unser lokaler Algorithmus mindestens die gleiche Mächtigkeit wie der ursprüngliche Algorithmus hat, wobei wir gleichzeitig die Spärlichkeit des zugrundeliegenden Graphen berücksichtigen und Overfitting verhindern. Anschließend stellen wir ein theoretisches Framework für die Analyse von Graphkernen vor, und zeigen, dass die meisten Kerne nicht in der Lage sind einfache graphentheoretische Eigenschaften zu unterscheiden. Der zweite Teil beschäftigt sich mit neuronalen Ansätzen zur Graphklassifikation und deren Verbindung zu Kern-Methoden. Wir zeigen, dass die Expressivität sogenannter Graph-Neural-Networks durch den 1-dimensionalen Weisfeiler-Leman-Algorithmus nach oben beschränkt werden kann.

1 Einleitung und Übersicht

Relationale Daten kommen heutzutage in verschiedenen Bereichen der Chemie- und Bioinformatik, in der Analyse sozialer Netzwerke und der Mustererkennung vor. Graphen können solche Daten auf natürliche Weise modellieren. Diese Arbeit befasst sich mit der Entwicklung von Methoden zur (überwachten) Graphklassifikation.

Im ersten Teil konzentrieren wir uns auf Kern-Methoden für Graph, sogenannte *Graphkerne*. Intuitiv gesprochen, messen Graphkerne die Ähnlichkeit eines Paars von Graphen. Genauer gesagt ist ein Graphkern eine positiv-semidefinite Funktion, die ein Paar von Graphen auf eine reelle Zahl abbildet. Daher können Graphkerne zusammen mit etablierten Lernalgorithmen, wie z. B. Support-Vektor-Maschinen, verwendet werden. Alternativ kann jeder Graph auf einen (endlichdimensionalen) Merkmalsvektor abgebildet werden. Der Graphkern ergibt sich dann durch das paarweise innere Produkt zwischen solchen Vektoren. Wir sprechen dann von einem *expliziten* Graphkern. In Anwendungen, z. B. in der Chemieinformatik, sind die Knoten und Kanten eines Graphen üblicherweise mit kontinuierlichen Merkmalen annotiert. Die in der Vergangenheit entwickelten Graphkerne, die mit solchen Eingaben umgehen konnten, skalieren nicht, da sie die Kernfunktion für jedes Paar von Graphen berechnen, was zu einem quadratischen Mehraufwand in der Laufzeit führt. Wir umgehen dieses Problem, indem wir jeden Graphen auf einen endlichdimensionalen, reellen Vektor abbilden. Wir zeigen, dass das innere Produkt zwischen solchen Vektoren bekannte Graphkerne

[1] Orginaltitel: „Learning with Graphs: Kernel and Neural Approaches“

[2] Technische Universität Dortmund, Fakultät für Informatik, christopher.morris@tu-dortmund.de

approximiert, die mit kontinuierlichen Eingaben umgehen können. Wir evaluieren die von uns vorgeschlagene Methode auf zahlreichen Benchmark-Datensätzen, und zeigen, dass unsere Methode vielfach schneller ist und trotzdem vergleichbar hohe Klassifikationsgüten liefert. Der nächste Abschnitt befasst sich mit Erweiterungen von Kernfunktionen, die auf dem 1-dimensionalen Weisfeiler-Leman-Algorithmus (1-WL) basieren. Dazu leiten wir einen Kern her, der auf dem k-dimensionalen Weisfeiler-Leman-Algorithmus (k-WL) basiert, der in Hinblick auf die Unterscheidung nicht-isomorpher Graphen mächtiger ist als der 1-WL. Um die Anwendbarkeit für das maschinelle Lernen zu gewährleisten, leiten wir eine lokale Variante her, die Overfitting verhindert und die Spärlichkeit des zugrundeliegenden Graphen berücksichtigt. Darüber hinaus zeigen wir, dass die lokale Variante mindestens die gleiche Mächtigkeit wie der ursprüngliche Algorithmus hat. Um den Algorithmus auf große Datenmengen anzuwenden, zeigen wir, wie man ihn approximieren kann. Schließlich evaluieren wir die von uns vorgeschlagenen Kerne empirisch, und zeigen, dass sie auf einer Vielzahl von Benchmark-Datensätzen anderen Methoden bezüglich Klassifikationsgüte überlegen sind. Abschließend entwickeln wir ein theoretisches Framework für die Analyse der Expressivität von Graphkernen, welches sich Konzepten aus dem Graph-Property-Testing bedient. Hier unterscheidet ein Graphkern eine Grapheigenschaft, wenn er einen konstanten Winkel (unabhängig von der Größe des Graphen) zwischen den Merkmalsvektoren zweier beliebiger Graphen garantiert, von denen einer die Eigenschaft hat und der andere weit davon entfernt ist. Wir untersuchen bekannte Graphkerne und zeigen, dass sie nicht in der Lage sind grundlegenden Eigenschaften zu unterscheiden.

Der zweite Teil der Arbeit befasst sich mit neuronalen Methoden zur Graphklassifikation. Insbesondere betrachten wir Graph-Neural-Networks (GNNs), iund zeigen, dass diese nicht mächtiger sein können als der 1-WL, in Bezug auf die Unterscheidung von nicht-isomorphen (Sub-)Graphen. Auf der anderen Seite zeigen wir, dass GNNs bei richtiger Parameterinitialisierung die gleiche Mächtigkeit wie der 1-WL haben. Darauf aufbauend schlagen wir, basierend auf dem k-WL, eine Verallgemeinerung von GNNs vor, sogenannte k-GNNs.

2 Kern-Methoden

Im Folgenden beschreiben wir die Arbeiten im Bereich der Graphkerne, die auf den Veröffentlichungen [Mo16], [MKM17], [MM19] und [Kr18] basieren.

2.1 Skalierbare Graphkerne für Graphen mit kontinuierlichen Merkmalen

Die Grundidee unseres sogenannten Hash-Graph-Kernel-Frameworks ist, dass man kontinuierliche Merkmale mittels einer Familie von Hashfunktionen auf diskrete Label abbildet. Darauffolgend wird ein skalierender Graphkern für Graphen mit diskreten Labeln auf dem so transformierten Graphen angewandt.

Sei $\mathscr{H} = \{h\colon \mathbb{R}^d \to \mathbb{N}\}$ eine Familie von Hashfunktionen und G ein Graph mit kontinuierlichen Knotenmerkmalen $a\colon V(G) \to \mathbb{R}^d$. Wir transformieren den Graphen (G,a) zu einem Graphen

mit diskreten Labeln, indem wir jedes Merkmal $a(v)$ mit einer Hashfunktion h aus $\mathscr{H}$ nach $h(a(v))$ abbilden. Wir schreiben $h(G)$ für den durch diese Prozedur erhaltenen Graphen. Die Funktion h wird hier zufällig und gleichverteilt aus der Familie von Hashfunktionen $\mathscr{H}$ gezogen. Um die Varianz zu kontrollieren, wird die obige Prozedur mehrmals wiederholt. Wir erhalten also eine Sequenz von diskret-gelabelten Graphen $(h_i(G))_{i=1}^I$, hier ist I die Anzahl der Iterationen. Diese Sequenzen von gelabelten Graphen wird nun mit einem beliebigen Graphkern für Graphen mit diskreten Labeln, dem sogenannten diskreten Basiskern, verglichen. Formal können wir den Hash-Graphkern folgendermaßen definieren.

Definition 2.1. Sei $\mathscr{H}$ eine Familie von Hashfunktionen und sei k_b ein diskreter Basiskern. Dann ist der Hash-Graphkern für zwei Graphen G und H mit kontinuierlichen Knotenmerkmalen definiert als

$$k_\mathrm{HGK}(G,H) = \frac{1}{I}\sum_{i=1}^{I} k_\mathrm{b}(h_i(G),h_i(H)),$$

hier ist h_i zufällig und gleichverteilt aus $\mathscr{H}$ gezogen.

Um die Skalierbarkeit des Hash-Graphkerns zu gewährleisten, benötigen wir explizite Merkmalsvektoren für den Kernel k_HGK. Das ist genau dann möglich, wenn wir diese für den diskreten Basiskern effizient berechnen können. Die Merkmalsvektoren können dann über alle Iterationen konkateniert werden und mit dem Faktor $\sqrt{1/I}$ skaliert werden.

Um zu zeigen, dass das obige Verfahren (langsame) Graphkerne für Graphen mit kontinuierlichen Merkmalen approximieren kann, müssen wir einige Anforderungen an die verwendete Familie von Hashfunktionen stellen. Die grundlegende Idee hierbei ist, dass man die Familie so auswählt, dass für zwei zufällige Hashfunktionen h_1 und h_2 die Kollisionswahrscheinlichkeit

$$\Pr[h_1(x) = h_2(y)]$$

einem Kern k gleicht, der die Ähnlichkeit von x und y aus $\mathbb{R}^d$ sinnvoll angibt. Für den Fall $h_1 = h_2$, wurden solche Familie bereits in [RR08] untersucht. Allerdings können diese hier nicht angewendet werden, da viele Graphkerne mittels Multiplikation aus anderen Kernfunktionen zusammengesetzt werden. Daher tritt ein Problem von Abhängigkeit auf. Daher führen wir das Konzept der *unabhängigen k-Hash-Familien* ein.

Definition 2.2. Sei $k\colon \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ eine Kernfunktion und sei $\mathscr{H} = \{h\colon \mathscr{X} \to S\}$ eine Familie von Hashfunktionen, die auf die Menge S abbilden. Dann ist $\mathscr{H}$ eine *unabhängige k-Hash-Familie*, wenn

$$\Pr[h_1(x) = h_2(y)] = k(x,y).$$

Hier sind h_1 und h_2 unabhängig und zufällig aus $\mathscr{H}$ gezogen.

Mithilfe der obigen Hash-Familien konnten wir zeigen, dass das Hash-Graph-Kernel-Framework langsame Graphkerne für Graphen mit kontinuierlichen Merkmalen mit hoher Wahrscheinlichkeit bis auf einen beliebig kleinen, additiven Fehler approximieren kann.

Das obige Verfahren wurde auf zahlreichen, praxisnahen Benchmark-Datensätzen evaluiert. Hier konnte gezeigt werden, dass Instanzen des Hash-Graph-Kernel-Frameworks deutlich schneller sind, dabei aber trotzdem vergleichbar hohe Klassifikationsgüten erzielen.

2.2 Expressive Graphkerne basierend auf dem k-dimensionalen Weisfeiler-Leman-Algorithmus

Die meisten aktuellen Graphkerne berücksichtigen nur *lokale* Grapheigenschaften, d. h. sie werden auf der Basis von Eigenschaften der Nachbarschaft von Knoten oder anderer kleiner Substrukturen berechnet, z. B. der Weisfeiler-Lehman-Subtree-Kern [Sh09], der auf dem 1-WL basiert. Obwohl der 1-WL bereits recht leistungsfähig ist und in anderen Bereichen erfolgreich angewendet wurde, ist er beispielsweise nicht in der Lage unterschiedlich lange Zyklen zu unterscheiden, was ein wichtiges Merkmal für die Analyse sozialer Netzwerke und in der Chemieinformatik ist. Eine mächtigere Erweiterung des 1-WL ist der k-WL.

Formal können wir diesen folgendermaßen beschreiben. Sei G ein Graph und sei $k \geq 2$. In jeder Iteration $i \geq 0$ berechnet der Algorithmus eine *Färbung* $C_i^k \colon V(G)^k \to \mathbb{S}$, wobei $\mathbb{S}$ eine beliebige Kodomäne ist. In der ersten Iteration ($i = 0$) erhalten zwei Tupel $\mathbf{v}$ und $\mathbf{w}$ in $V(G)^k$ die gleiche Farbe, falls die Abbildung $v_i \mapsto w_i$ einen Isomorphismus zwischen den induzierten Graphen $G[\mathbf{v}]$ and $G[\mathbf{w}]$ beschreibt. Für $i \geq 0$ ist C_{i+1}^k folgendermaßen definiert

$$C_{i+1}^k(\mathbf{v}) = (C_i^k(\mathbf{v}), M_i(\mathbf{v})),$$

hierbei ist

$$M_i(\mathbf{v}) = \big(\{\!\{C_i^k(\phi_1(\mathbf{v},w)) \mid w \in V(G)\}\!\}, \ldots, \{\!\{C_i^k(\phi_k(\mathbf{v},w)) \mid w \in V(G)\}\!\}\big). \tag{1}$$

Hier bezeichnet $\{\!\{\ldots\}\!\}$ eine Multimenge. Ferner ist

$$\phi_j(\mathbf{v},w) = (v_1, \ldots, v_{j-1}, w, v_{j+1}, \ldots, v_k).$$

Hier tauscht $\phi_j(\mathbf{v},w)$ also die j-te Komponente des Tupels $\mathbf{v}$ durch den Knoten w aus. Der k-WL berücksichtigt zwar mehr globale Eigenschaften, aber keine lokalen Eigenschaften. Ferner zeigen Experimente, dass er nicht skaliert und zu Overfitting führt, daher schlagen wir eine *lokale Variante* vor. Diese färbt ebenfalls alle k-Tupel, um nun aber lokale Merkmale zu extrahieren, definieren wir die lokale Nachbarschaft eines k-Tupel. Anstatt Gleichung 1 verwenden wir

$$M_i^\delta(\mathbf{v}) = \{\!\{s_\delta(\mathbf{v},w) \mid w \in V(G)\}\!\},$$

wobei

$$s_\delta(\mathbf{v},w) = \{(C_i^{k,\delta}(\phi_j(\mathbf{v},w)), j) \mid j \in [k] \text{ und } w \in \delta(v_j)\},$$

und $\delta(v)$ die Menge der Nachbarn des Konten v bezeichnet. Daher ist die Färbungsfunktion definiert als

$$C_{i+1}^{k,\delta}(\mathbf{v}) = (C_i^{k,\delta}(\mathbf{v}), M_i^\delta(\mathbf{v})).$$

Wir zeigen, dass eine Variante des lokalen Algorithmus mindestens die gleiche Mächtigkeit, in Bezug auf die Unterscheidung von nicht-isomorphen Graphen, wie der k-WL hat. Für spärliche Graphen kann die Laufzeit für jede Iteration des lokalen Algorithmus nach oben durch $|V(G)^k| \cdot kd$ begrenzt werden, wobei d den maximalen oder durchschnittlichen Grad des Graphen bezeichnet. Daher berücksichtigt der lokale Algorithmus die Spärlichkeit des zugrundeliegenden Graphen, was zu (iterationsweise) verbesserten Berechnungszeiten im

Vergleich zu dem nicht-lokalen k-WL führt. Ferner können wir experimentell zeigen, dass er Overfitting verhindert.

Wir berechnen nun einen Kern basieren auf dem lokalen Algorithmus, indem wir ihn für für $h \geq 0$ Iterationen ausführen. Nach jeder Iteration i berechnen wir einen Merkmalsvektor $\phi_{\mathrm{L}}^{i}(G)$ aus $\mathbb{R}^{|\mathbb{S}_i|}$ für jeden Graphen G, wobei $\mathbb{S}_i \subseteq \mathbb{S}$ das Bild von $C_i^{k,\delta}$ bezeichnet. Jede Komponente $\phi_{\mathrm{L}}^{i}(G)_s$ zählt die Anzahl der Vorkommen von k-Tupeln, die mit s in $\mathbb{S}_i$ gefärbt sind. Der finale Merkmalsvektor $\phi_{\mathrm{L}}(G)$ ist dann definiert als die Konkatenation der Merkmalsvektoren aller h Iterationen, d. h

$$\left[\phi_{\mathrm{L}}^{0}(G), \ldots, \phi_{\mathrm{L}}^{h}(G)\right].$$

Seien G und H zwei Graphen dann ist der entsprechende Kernel $k_{\mathrm{L}}(G,H) = \langle \phi_{\mathrm{L}}(G), \phi_{\mathrm{L}}(H) \rangle$. Da der Algorithmus alle möglichen k-Tupel betrachtet skaliert er nicht für sehr große Graphen. Daher haben wir einen Approximationsalgorithmus entwickelt, der den exakten Merkmalsvektor bis auf einen additive Fehler bezüglich der ℓ_1-Norm annähern kann. Für Graphen mit beschränkten Grad konnten wir zeigen, dass die Laufzeit konstant in der Graphgröße ist.

In einer experimentellen Studie konnten wir zeigen, dass der obige Kernel auf einer Vielzahl von Benchmark-Datensätzen aktuelle Graphkerne schlägt, und deutlich schneller als der ursprüngliche k-WL ist.

2.3 Ein theoretisches Framework zur Untersuchung der Expressivität von Graphkernen

In den vergangenen zwei Jahrzehnten wurde eine große Anzahl von Graphkernen vorgeschlagen, siehe z. B. [KJM20]. In Anbetracht der großen Anzahl verfügbarer Graphkerne und der Fülle von Benchmark-Datensätzen wird es zunehmend schwierig einen fairen experimentellen Vergleich der Kerne durchzuführen und ihre Vor- und Nachteile für bestimmte Datensätze und Anwendungsszenarien zu spezifizieren. Tatsächlich können die derzeitigen experimentellen Vergleiche kein vollständiges Bild vermitteln und sind von begrenzter Nützlichkeit für einen Praktiker, der einen Kernel für eine bestimmte Anwendung auswählen muss. Daher führen wir ein theoretisches Framework für die Analyse der Expressivität von Graphkernen ein, welches durch Konzepte aus dem Graph-Property-Testing motiviert ist.

Im Folgenden betrachten wir Graphen mit beschränkten Grad, d. h. der Knotengrad ist unabhängig von der Anzahl der Knoten. Seien G und H zwei Graphen mit n Knoten mit maximalem Knotengrad d. Dann ist die *Editierdistanz* $\Delta(G,H)$ zwischen G und H definiert als die minimale Anzahl von Kantenmodifikation, d. h. das Hinzufügen oder das Entfernen von Kanten, die benötigt werden, um G in eine isomorphe Kopie von H abzuändern. Eine *Grapheigenschaft* ist nun eine Menge $\mathscr{P}$ von Graphen, die unter Isomorphie abgeschlossen ist. Die Menge von Graphen aus $\mathscr{P}$ mit n Knoten bezeichnen wir mit $\mathscr{P}_n$. Ein Graph G auf n Knoten mit maximalem Knotengrad d ist ε-*weit* von $\mathscr{P}_n$ entfernt, falls für alle Graphen H mit maximalem Knotengrad d aus $\mathscr{P}_n$ gilt, dass

$$\Delta(G,H) > \varepsilon dn$$

für $\varepsilon > 0$. Ferner sei $\hat{k}$ der normalisierte Kern eines Kerns k and sei $\hat{\phi}$ der normalisierte Merkmalsvektor, d. h.

$$\hat{k}(x,y) = \langle \hat{\phi}(x), \hat{\phi}(y) \rangle = \left\langle \frac{\phi(x)}{\|\phi(x)\|_2}, \frac{\phi(y)}{\|\phi(y)\|_2} \right\rangle$$
$$= \frac{k(x,y)}{\sqrt{k(x,x) \cdot k(y,y)}} \in [-1,1].$$

Der normalisierte Kern $\hat{k}(x,y)$ entspricht dem Kosinus des Winkels zwischen $\phi(x)$ und $\phi(y)$ im Merkmalsraum. Ein Graphkern *identifiziert* eine Eigenschaft, falls keine zwei Graphen auf den gleichen Merkmalsvektor abgebildet werden, außer beide haben die Eigenschaft oder beide haben sie nicht.

Definition 2.3. Sei $\mathscr{P}$ eine Grapheigenschaft. Falls ein Graphkern $k \colon \mathscr{G} \times \mathscr{G} \to \mathbb{R}$ für jedes n aus $\mathbb{N}$, und jeden G aus $\mathscr{P}_n$ und H nicht in $\mathscr{P}_n$

$$\frac{k(G,H)}{\sqrt{k(G,G) \cdot k(H,H)}} < 1$$

erfüllt, *identifiziert* k die Eigenschaft $\mathscr{P}$.

Damit ein Graphkern eine Grapheigenschaft unterscheiden kann, ist es wünschenswert, dass ein konstanter Winkel, unabhängig von n, zwischen den Merkmalsvektoren existiert. Dies kann aber oft nicht erreicht werden, wie das folgende Resultat für den Shortest-Path-Kern [BK05] zeigt.

Proposition 2.4. Für den Shortest-Path-Kern k und jede Konstante c, $0 < c < 1$, existieren n aus $\mathbb{N}$ und zwei Graphen G und H aus $\mathscr{G}_n$, wobei G zusammenhängend, und H nicht zusammenhängend ist, so dass

$$\frac{k(G,H)}{\sqrt{k(G,G) \cdot k(H,H)}} > 1 - c.$$

Um einen Winkel unabhänging von der Graphgröße zu erhalten, führen wir den Begriff von *Unterscheidbarkeit* ein.

Definition 2.5. Eine Grapheigenschaft $\mathscr{P}$ ist durch den Graphkern $k \colon \mathscr{G} \times \mathscr{G} \to \mathbb{R}$ *unterscheidbar*, falls für jedes $\varepsilon > 0$, und d aus $\mathbb{N}$, ein $\delta = \delta(\varepsilon, d) > 0$ existiert, so dass für jedes n aus $\mathbb{N}$, jeden G in $\mathscr{P}_n$, und jeden H, der ε-weit von $\mathscr{P}_n$ entfernt ist, gilt, dass

$$\frac{k(G,H)}{\sqrt{k(G,G) \cdot k(H,H)}} \leq 1 - \delta.$$

Im Folgenden untersuchen wir Identifizierbarkeit und Unterscheidbarkeit des Random-Walk- [GFW03], des Weisfeiler-Lehman-Subtree- [Sh11], des Shortest-Path- [BK05], und des Graphlet-Kerns [Sh09]. Tabelle 1 fasst die Resultate im Vergleich zum k-Disc-Kern, siehe unten, zusammen.

Eigenschaft	Graphkern				
	WL	RW	SP	GRAPHLET	k-DISC
Zusammenhang	✗	✗	✓	✗	✓
Planarität	✗	✗	✗	✗	✓
Bipartitheit	✗	✗	✗	✗	
Dreiecksfreiheit	✗	✗	✗	•	✓

Tabelle 1: Unterscheidbarkeit von Grapheigenschaften für den Weisfeiler-Lehman-Subtree- (WL), Random-Walk- (RW), Graphlet-, Shortest-Path- (SP), und k-Disc-Kern, Key: ✓Unterscheidbar, •Identifizierbar (aber nicht unterscheidbar), and ✗Nicht identifizierbar.

Die Merkmalsvektoren des Random-Walk- und des Weisfeiler-Lehman-Subtree-Kerns können reguläre Graphen nicht unterscheiden. Wenn also für eine Grapheigenschaft $\mathscr{P}$ and n aus $\mathbb{N}$ ein regulärer Graph mit der Eigenschaft existiert und ein anderer regulärer Graph, der ε-weit von der Eigenschaft entfernt ist, existiert, dann können beide Kerne die Eigenschaft nicht identifizieren. Daher gilt das folgende Resultat.

Theorem 2.6. Der Random-Walk- und der Weisfeiler-Lehman-Subtree-Kern können Zusammenhang, Planarität, Bipartitheit und Dreiecksfreiheit nicht identifizieren.

Für den Shortest-Path-Kern konnten wir zeigen, dass er nur Zusammenhang identifizieren kann. Ferner konnten wir für den Graphlet-Kernel zeigen, dass er nur Dreiecksfreiheit identifizieren kann.

Im Folgenden leiten wir einen Graphkern her, der obige Eigenschaften unterscheiden kann und effizient berechenbar ist. Wir definieren hierzu das Histogramm $\text{hist}_G(k)$, das die Anzahl an unterschiedlichen (d. h. nicht-isomorphen) k-Discs um einen Knoten in G zählt. Hierbei ist eine k-Disk um einen Konten v aus $V(G)$ der induzierte Subgraph aus allen Knoten, die von v auf einem Pfad der Länge kleiner als k erreichbar sind. Ferner benötigen wir das normalisierte Histogramm $\text{freq}_G(k) = \text{hist}_G(k)/n$ für G aus $\mathscr{G}_n$. Wir betrachten den folgenden Graphkern. Seien G and H Graphen, dann ist der *k-Disc-Kern* definiert als

$$k_{\text{KD}}(G,H) = \langle \text{freq}_G(k), \text{freq}_H(k) \rangle.$$

Ein entscheidener Unterscheid zum Graphlet-Kernel ist hier, dass der k-Disc-Kern nur zusammenhängende Subgraphen betrachtet, wohingegen der Graphlet-Kernel alle möglichen Subgraphen betrachet. Für Graphen mit beschränkten Grad kann der k-Disc-Kernel in linearer Zeit berechnet. Das folgende Resultat stellt nun das Hauptresultat dar.

Theorem 2.7. Für den k-Disc-Kernel gilt, dass er
(1) Zusammenhang für $k \geq 4/\varepsilon d$ unterscheiden kann,
(2) Dreieckfreiheit fuer $k \geq 1$ unterscheiden kann, und
(3) für jedes $\varepsilon > 0$, d aus $\mathbb{N}$, existiert ein k aus $\mathbb{N}_{>0}$, sodass Planarität unterschieden werden kann.

Ferner haben wir Lerngarantieren für den Kern-Nächste-Nachbar-Klassifizier hergeleitet. Wir zeigen, dass der Vorhersagefehler für die Klassifikation einer Grapheigenschaft unter der Annahme begrenzt werden kann, dass der verwendete Kernel die Eigenschaft unterscheiden kann. Die Anzahl der benötigen Trainingsbeispiele hängt hier nur von der Dimension des Merkmalsraums ab.

3 Neuronale Methoden

In diesem Abschnitt stellen wir, basierend auf [Mo19], neuronale Methoden für die Graphklassifikation vor. Hier haben sich GNNs in den letzten Jahren als Standardarchitektur herausgebildet. Im Folgenden zeigen wir die Grenzen dieser Architektur auf. Insbesondere zeigen wir, dass jede mögliche GNN-Architektur hinsichtlich der Unterscheidung nicht-isomorpher Graphen nicht mächtiger als der 1-WL sein kann. Des Weiteren schlagen wir, basierend auf dem k-WL, eine beweisbar mächtigere Verallgemeinerung von GNNs vor.

GNNs können als eine neuronale Variante des 1-WL angesehen werden, bei dem Farben durch kontinuierliche Merkmalsvektoren ersetzt werden und neuronale Netze zur Aggregation der Nachbarschaft eines Knotens verwendet werden. Tatsächlich kann das GNN-Framework als Implementierung einer kontinuierlichen Form von graphbasiertem „Message Passing“ betrachtet werden, bei dem lokale Nachbarschafts-Informationen aggregiert und an die Nachbarn weitergegeben werden. Durch den Einsatz eines trainierbaren neuronalen Netzes zur Aggregation von Informationen aus den Knoten-Nachbarschaften können GNNs zusammen mit den Parametern des Klassifikations- oder Regressionsalgorithmus Ende-zu-Ende trainiert werden.

Im Folgenden beschreiben wir GNNs formal. Sei (G,l) ein gelabelter Graph mit initialer Knotenfärbung $f^{(0)}\colon V(G) \to \mathbb{R}^{1\times d}$, die *konsistent* mit l ist. Das heißt, dass jeder Knoten mit einem Merkmalsvektor $f^{(0)}(v)$ aus $\mathbb{R}^{1\times d}$ annotiert, sodass $f^{(0)}(u) = f^{(0)}(v)$ genau dann wenn $l(u) = l(v)$. Alternativ können in $f^{(0)}(v)$ anwendungsspezifische Informationen kodiert werden, z. B. kontinuierliche Eigenschaften von chemischen Molekülen oder Vektorrepräsentation von Text in sozialen Netzwerken. Ein GNN-Modell besteht nun aus einer Komposition von neuronalen Schichten, wobei jede Schicht Informationen aus der lokalen Nachbarschaft eines Knotens aggregiert. Ein einfaches GNN-Modell kann folgendermaßen beschrieben werden. In jeder Schicht $t > 0$, berechnen wir ein neues Merkmal

$$f^{(t)}(v) = \sigma\Big(f^{(t-1)}(v)\cdot W_1^{(t)} + \sum_{w\in N(v)} f^{(t-1)}(w)\cdot W_2^{(t)}\Big) \tag{2}$$

aus $\mathbb{R}^{1\times e}$ für v aus $V(G)$. Hierbei sind $W_1^{(t)}$ and $W_2^{(t)}$ Parametermatrizen aus $\mathbb{R}^{d\times e}$, und σ ist eine punktweise, nicht-lineare Funktion, z. B. eine Sigmoid- oder eine Rectifier-Funktion.

Die innere Summe über die Nachbarschaft kann durch eine Permutations-invariante, differenzierbare Funktion ersetzt werden, und die äußere Summe durch spaltenweise Konkatenation oder LSTM-artige Funktion ersetzt werden. Daher kann eine allgemeine GNN-Schicht als

$$f_{\text{merge}}^{W_2}\Big(f^{(t-1)}(v), f_{\text{aggr}}^{W_1}\big(\{\!\{f^{(t-1)}(w) \mid w\in N(v)\}\!\}\big)\Big), \tag{3}$$

geschrieben werden. Hier aggregiert $f_{\text{aggr}}^{W_1}$ über die Multimenge der Merkmale der Nachbarn und $f_{\text{merge}}^{W_2}$ vereinigt die Knotenrepräsentation des vorherigen Schritts mit den berechneten Merkmal der Nachbarn. Hierbei sind $f_{\text{aggr}}^{W_1}$ und $f_{\text{merge}}^{W_2}$ beliebige differenzierbare, Permutation-invariante Funktionen, z. B. neuronale Netzwerke. Im Folgenden bezeichnen wir obige Architekturen als 1-*dimensionale GNNs* (1-GNNs).

Im Folgendem untersuchen wir die Beziehung zwischen dem 1-WL und 1-GNNs. Sei (G,l) ein gelabelter Graph, und seien $\mathscr{W}^{(t)} = \left(\mathbf{W_1^{(t')}}, \mathbf{W_2^{(t')}}\right)_{t' \leq t}$ die Parameter eines 1-GNNs bis Iteration t. Hier kodieren wie initialen Label $l(v)$ als Vektoren $f^{(0)}(v)$ aus $\mathbb{R}^{1\times d}$, z. B. als 1-Hot-Kodierung. Unser erstes theoretisches Ergebnis zeigt, dass jede 1-GNN-Architektur, in Bezug auf die Unterscheidung von nicht-isomorphen Subgraphen, nicht mächtiger sein kann als der 1-WL. Formal, seien $f_{\text{aggr}}^{W_1}$ und $f_{\text{merge}}^{W_2}$ beliebige differenzierbare Funktionen und $\mathscr{W}^{(t)}$ beliebig gewählt, dann gilt, dass die Färbung C_t^1 vom 1-WL immer die Färbung $f^{(t)}$ des 1-GNN $\mathscr{W}^{(t)}$ verfeinert.

Theorem 3.1. Sei (G,l) ein gelabelter Graph. Dann gilt für alle $t \geq 0$, und allen initialen Färbungen $f^{(0)}$, die konsistent mit l sind, und alle Parameter $\mathscr{W}^{(t)}$, dass

$$C_t^1 \sqsubseteq f^{(t)}.$$

Unser zweites Resultat zeigt, dass eine Sequenz von Parametermatrizen $\mathscr{W}^{(t)}$ existiert, sodass das korrespondierende 1-GNN genau die gleiche Mächtigkeit wie der 1-WL hat.

Theorem 3.2. Sei (G,l) ein gelabelter Graph. Sei $t \geq 0$, dann existiert eine Sequenz von Parametermatrizen $\mathscr{W}^{(t)}$ und eine 1-GNN-Architektur, sodass

$$C_t^1 \equiv f^{(t)}.$$

Ferner, wurde, basierend auf dem k-WL, die k-GNN-Architektur entwickelt, und das obige Resultat für allgemeine k hergeleitet. In einer experimentellen Studie wurde diese auf zahlreichen Benchmark-Datensätzen untersucht. Insbesondere konnten für k-GNNs hervorragende Resultate für die Regression von quantenchemischen Eigenschaften gezeigt werden.

4 Zusammenfassung

Die vorliegende Arbeit beschäftige sich mit Methoden für die (überwachte) Graphklassifikation. Im Bereich der Kernmethoden haben wir skalierende Graphkerne für Graphen mit kontinuierlichen Knotenmerkmalen vorgeschlagen. Ferner wurde ein Graphkern, der globale Eigenschaften von Graphen einfangen kann, entwickelt. Zusätzlich wurde ein Framework für die Untersuchung der theoretischen Expressivität von Graphkernen vorgestellt. Hier wurde gezeigt, dass aktuelle Kerne nicht in der Lage sind einfach Grapheigenschaften einzufangen und ein beweisbar mächtiger Graphkern vorgestellt. Im Bereich der neuronalen Methoden wurden die Grenzen von GNNs aufgezeigt und eine beweisbar mächtigere Architektur hergeleitet.

Literaturverzeichnis

[BK05] Borgwardt, K. M.; Kriegel, H.-P.: Shortest-path kernels on graphs. In: IEEE International Conference on Data Mining. IEEE, S. 74–81, 2005.

[GFW03] Gärtner, T.; Flach, P.; Wrobel, S.: On Graph Kernels: Hardness Results and Efficient Alternatives. In: Learning Theory and Kernel Machines, Jgg. 2777 in Lecture Notes in Computer Science, S. 129–143. 2003.

[KJM20] Kriege, N. M.; Johansson, F. D.; Morris, C.: A survey on graph kernels. Applied Network Science, 5(1), Jan 2020.

[Kr18] Kriege, N. M.; Morris, C.; Rey, A.; Sohler, C.: A Property Testing Framework for the Theoretical Expressivity of Graph Kernels. In: International Joint Conference on Artificial Intelligence. IJCAI, S. 2348–2354, 2018.

[MKM17] Morris, C.; Kersting, K.; Mutzel, P.: Glocalized Weisfeiler-Lehman Kernels: Global-Local Feature Maps of Graphs. In: IEEE International Conference on Data Mining. IEEE, S. 327–336, 2017.

[MM19] Morris, C.; Mutzel, P.: Towards a practical *k*-dimensional Weisfeiler-Leman algorithm. CoRR, abs/1904.01543, 2019.

[Mo16] Morris, C.; Kriege, N. M.; Kersting, K.; Mutzel, P.: Faster Kernel for Graphs with Continuous Attributes via Hashing. In: IEEE International Conference on Data Mining. IEEE, S. 1095–1100, 2016.

[Mo19] Morris, C.; Ritzert, M.; Fey, M.; Hamilton, W. L.; Lenssen, Jan Eric; Rattan, G.; Grohe, M.: Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks. In: AAAI Conference on Artificial Intelligence. S. 4602–4609, 2019. AAAI.

[Mo20] Morris, Christopher: Learning with Graphs: Kernel and Neural Approaches. Dissertation, TU Dortmund University, 2020.

[RR08] Rahimi, A.; Recht, B.: Random features for large-scale kernel machines. In: Advances in Neural Information Processing Systems. NIPS, S. 1177–1184, 2008.

[Sh09] Shervashidze, N.; Vishwanathan, S. V. N.; Petri, T. H.; Mehlhorn, K.; Borgwardt, K. M.: Efficient Graphlet Kernels for Large Graph Comparison. In: International Conference on Artificial Intelligence and Statistics. PMLR, S. 488–495, 2009.

[Sh11] Shervashidze, N.; Schweitzer, P.; van Leeuwen, E. J.; Mehlhorn, K.; Borgwardt, K. M.: Weisfeiler-Lehman Graph Kernels. Journal of Machine Learning Research, 12:2539–2561, 2011. JMLR.

Christopher Morris studierte Informatik an der Technischen Universität Dortmund. Seit Juli 2015 war er wissenschaftlicher Mitarbeit in der Arbeitsgruppe von Prof. Dr. Petra Mutzel und Mitglied des Sonderforschungsbereichs 876 „Verfügbarkeit von Information durch Analyse unter Ressourcenbeschränkung“. Im Jahre 2019 hatte er einen dreimonatigen Gastaufenthalt an der Universität Stanford. Im Jahre 2020 schloss er mit Auszeichnung („summa cum laude“) seine Promotion zu einem Thema im Bereich des maschinellen Lernens mit Graphen ab. Seit März 2020 ist er wissenschaftlicher Mitarbeiter in der Arbeitsgruppe von Prof. Dr. Andrea Lodi an der Universität Montreal, Kanada.