GCN

8. Februar 2017

1 Graph Convolutional Networks

- großes wissenschaftliches Problem, willkürliche Graphen in ein neuronales Netz zu füttern
- Bereich bisher dominiert von kernelbasierten Methoden, graphbasierter Regularisierung oder ähnlichem
- Es gibt eine Ansätze, der neuste ist ein *spektraler*
- spektraler Ansatz gilt als langsam
- das ganze zählt zu dem Bereich des *Semi-unsupervised Learning*, das bedeutet, dass wir einen Teil des Graphen gelabelt haben und basierend auf seiner Graphstruktur, den Rest labeln wollen.

1.1 Definitionen

- Wir wollen eine Funktion von Merkmalen auf einem Graphen $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ lernen
- **Eingaben:**
  - Eine Merkmalsbeschreibung x_i für jeden Knoten i dargestellt als eine Merkmalsmatrix $X = N \times D$, wobei N Anzahl der Knoten und D Anzahl der Merkmale
  - Eine repräsentative Beschreibung einer Graphstruktur in Matrixform, normalerweise eine Adjazenzmatrix A
- **Ausgabe**:
  - eine Merkmalsbeschreibung $Z = N \times F$ für jeden Knoten, wobei F Anzahl der Ausgabefeatures für einen Knoten

Jede Schicht des neuronalen Netzes kann über eine nicht-lineare Funktion

$$H^{(l+1)} = f(H^{(l)}, A) \tag{1}$$

beschrieben werden, wobei $H^{(0)} = X$ und $H^{(L)} = Z$ mit L Anzahl der Schichten.
Die Propagationsregel ist dann zum Beispiel:

$$f(H^{(l)}, A) = \sigma(AH^{(l)}W^{(l)}) \tag{2}$$

wobei, σ eine nicht lineare Aktivierungsfunktion ist (z.B. `ReLU`) und $W^{(l)}$ eine Gewichtsmatrix für den $l$ten Layer.

A muss jedoch leicht modifziert werden, denn eine Multiplikation mit A summiert alle Feature Vectors der lokalen Nachbarschaftsknoten auf, jedoch ohne den betrachteten Knoten (falls Knoten keine Kante zu sich selbst). Wir können das fixen, in dem wir für jeden Knoten eine Kante sich zu selbst hinzufügen in dem wir A mit der Identitätsmatrix addieren.

Desweiteren ist A nicht normalisiert, das bedeutet, dass eine Multiplikation mit A die Featurevectors komplett anders skaliert. A kann zum Beispiel normalisiert werden, in dem alle Reihen zu Eins aufsummiert werden.

Der erste Layer würde dann die Features eines Knotens mit denen der direkten Nachbarschaft kombinieren. Ein weiterer Layer würde, diess für alle Wege mit Länge 2 tun und so weiter.

Dies entspricht in etwa einer generaliserten Version des Weisfeiler-Lehman Algorithmus auf Graphen:

Für alle Knoten $v_i \in \mathcal{G}$:

1. Sammle Merkmale $\{h_{v_j}\}$ für alle Nachbarschaftsknoten v_j
2. Update Knotenmerkmal $h_{v_i} \leftarrow \text{hash}(\sum_j h_{v_j})$ Wiederhole k-mal oder bis Konvergenz.

Jedem Knoten wird also ein Merkmal zugeordnet, dass seine Rolle im Graphen beschreibt. Dies funktioniert jedoch nicht gut für z.B. reguläre Graphen, bei dem jeder Knoten gleich viele Kanten besitzt. Der Weisfeiler-Lehman Algorithmus wird oft benutzt, um Graphisomorphismen zu bestimmen.

1.2 Klassifierung eines Graphen

Wie kann dieses Model genutzt werden, um einen Graphen zu klassifizieren? Wir erhalten $N \times F$ Features für einen Graphen nach x vielen Convolutions. F kann natürlich dann auch die Anzahl der Klassen sein.

Weiteres Vorgehen: Wir wollen nicht einzelne Knoten labeln, sondern das gesamte Bild bzw. den gesamten Graphen. Diese Features beschreiben den Knoten sowie seine direkte und indirekte Nachbarschaft. Die Knoten müssten jetzt eigentlich noch geeignet gelabelt werden und können dann in ein Netz gefüttert werden.

Wir können Spatial Informationen ausnutzen, wir wissen wie der Graph in der Lage aussieht. Das sollte man sich ruhig zu nutze machen.

1.3 Offene Fragen

Adjazenzmatrix mit Gewichten? Wie ist das mit der Identität? Gewichte müssten ja eigl umgekehrt werden damit nahe Knoten (niedriges Gewicht) mehr Einfluss haben.

Ein weiteres Problem sind variable Length Convolutions. Die brauchen wir unbedingt.