Леонов Алексей
ФКН, 165 группа

Обзор по теме курсового проекта

Поиск сообществ в сложных сетях

Введение

Сложные сети – это графы, обладающие определёнными свойствами, какими обладают большинство графов, встречаемых в реальной жизни: социальные графы, нейронные сети, веб-графы и т.п. Их характерные черты – малый диаметр (как правила порядка $O(\log n)$, где n – количество вершин в графе), большой коэффициент кластеризации, иерархическая структура. Ещё одной характерной чертой является степенной закон распределения различных величин, например степеней вершин, что делает данные сети независимыми от масштаба (scale-free)[1].

Сообществами называются такие подмножества вершин, у которых много рёбер внутри подмножества и мало – с остальными вершинами[2, 3]. Существуют более строгие определения сообщества, но они не общеприняты. Поиск сообществ в сети важен, поскольку с его помощью можно изучить структуру сети, выявив в ней основные части и взаимодействия между ними. Для поиска сообществ существует множество алгоритмов, которые будут рассмотренны в основной части.

Помимо рёбер на основе явно заданных связей между вершинами в сложной сети можно строить рёбра на основе метаданных, ассоциированных с вершинами, соединяя вершины со схожими метаданными. Такой граф называется графом данных (proximity graph). Существует несколько методов построения графа данных[7] (relative neigborhood graph, gabriel graph и т.д.), которые будут рассмотрены в основной части обзора.

Обзор существующих решений

Поскольку сложные сети как правило имеют большое количество вершин и рёбер, масштабируемость используемых алгоритмов критически важна.

Поиск сообществ

Будем рассматривать задачу разбиения вершин графа на непересекающиеся подмножества – сообщества, отметив что помимо неё существует также задача разбиения на пересекающиеся сообщества (soft clustering).

Алгоритм Гирвана-Ньюмана на каждом шаге удаляет ребро с наибольшим значением "промежуточности" (edge betweenness), которая может определяться по разному, до тех пор, пока граф не распадётся на k связных компонент, где k – заранее заданное число сообществ[4]. Его сложность зависит от определения функции промежуточности и алгоритма её вычисления, но в лучшем случае составляет $\Theta(n(n+m))$, что слишком много для больших графов. Ещё одним недостатком является необходимость заранее указывать количество сообществ, которое обычно заранее неизвестно.

Кроме этого существуют методы, в которых количество сообществ автоматически определяется алгоритмом, исходя из данных. Важнейший класс таких методов основан на введении метрики качества разбиения вершин на сообщества и оптимизации введённого функционала. В качестве такого функционала часто берут модулярность, определяемую следующим образом.

$$Q = \frac{1}{m} \sum_{i,j \in V} (A_{ij} - P_{ij})[c_i = c_j]$$

Здесь m – количество ребёр, V – множество вершин, A – матрица смежности графа, c_i – сообщество, к которому отнесена вершина i. P_{ij} – некоторая оценка A_{ij} для случайного графа, например $\frac{k_i k_j}{2m}$, где k_i – степень вершины i.

Задача оптимизации модулярности NP-полна, поэтому используемые на практике алгоритмы не ищут точное решение, а лишь стараются найти как можно лучшее приближение. Один из таких алгоритмов, CNM[5] – работает за $O(n \log^2 n)$ для сложных сетей. Он создаёт по одному сообществу на каждую вершину, а затем на каждом шаге объединяет два таких сообщества, что при этом увеличение модулярности будет на данном шаге

максимальным. Такие итерации продолжаются до тех пор, пока есть пара сообществ, объединение которых увеличит функцию модулярности. Таким образом, он жадно максимизирует функционал модулярности.

Упомянем некоторые другие функционалы помимо функции модулярности. Будем обозначать за A матрицу смежности, за K количество сообществ, за $V_1, \dots V_k$ сами сообщества.

1. Глобальная плотность:

$$Q_{GD}(G,V) = \frac{1}{2}(Q^i_{GD}(G,V) + 1 - Q^e_{GD}(G,V))$$

Здесь два слагаемых отвечают за внутреннюю и внешнюю плотность рёбер:

$$Q^i_{GD}(G,V) = \frac{\sum_{i=1}^{K}\sum_{i,j\in V_k} A_{ij}}{\sum_{i=1}^{K}|V_i|^2}$$

$$Q^e_{GD}(G,V) = \frac{\sum_{i=1}^{K}\sum_{i\in V_k, j\notin V_k} A_{ij}}{\sum_{i=1}^{K}|V_i|\cdot|V\backslash V_i|}$$

2. Локальная взвешенная плотность:

$$Q_{LD}(G,V) = \frac{1}{2|V|}\sum_{i=1}^{K}\frac{1}{|V_k|}(q_i(G,V_k) - q_e(G,V_k)$$

Где q_i и q_e отвечают за внутреннюю и внешнюю плотность ребёр в каждом из сообществ:

$$q_i(G,V_k) = \frac{\sum_{i,j\in V_k} A_{ij}}{|V_k|^2}$$

$$q_e(G,V_k) = \frac{\sum_{i\in V_k, j\notin V_k} A_{ij}}{|V_k|\cdot|V\backslash V_k|}$$

3. Distance based quality function:

$$Q_{DB}(G,V) = \frac{1}{|V|^2}||A - B||$$

Где $B_{ij} = 1$, если i и j лежат в одном сообществе и 0 иначе (то есть матрица смежности, в случае если все сообщества – не связанные друг с другом клики).

4. Node membership quality function:

$$G_{NM}(G,V) = \frac{1}{2|V|}\sum_{i=1}^{|V|}\mu(i,V[i]) + 1 - \mu(i,V\backslash V[i])$$

Здесь $V[i]$ обозначает сообщество, в котором лежит i-ая вершина, а $\mu(i,S) = \frac{\sum_{j\in S} A_{ij}}{|S|}$ – плотность ребёр между вершиной и множеством.

Другой подход к поиску сообществ основан на запуске случайного процесса, в результате работы получается разбиение множества вершин на сообщества. label propagation[6] – один из таких алгоритмов. Вначале каждой вершине присваивается своё сообщество, а затем на каждой итерации для каждой вершины считается число её соседей из каждого сообщества и выбирается сообщество с максимальным числом соседей. В зависимости от модификации либо сообщество вершины обновляется сразу после подсчёта (асинхронный вариант), либо сначала для всех вершин выбираются новые сообщества, а потом они одновременно обновляются после итерации по всем вершинамы (синхронный вариант), либо сообщество вершины обновляется сразу, но вершины перебираются в таком порядке, что вершины из одного сообщества идут подряд (полу-синхронный

вариант). Итерации продолжаются до тех пор, пока количество поменявших сообщество вершин на каждом шаге не станет меньше некоторого числа, выбираемого в зависимости от данных.

Каждая итерация имеет временную сложность $O(n)$, при этом число итераций, необходимых, чтобы процесс сошёлся, невелико даже для больших n. Таким образом, данный алгоритм является масштабируемым.

Отметим также метод consensus clustering, когда на основе нескольких разбиений на сообщества (например, сгенерированных одним алгоритмом, зависящим от случайности) строится новое разбиение. Для каждого ребра считается, в скольки из разбиений оно соединяет вершины из одного сообщества. Далее из графа выбрасываются рёбра, для которых это число меньше определённой заданной границы, и в оставшемся графе находятся связные компоненты, которые и будут сообществами. Таким образом, за счёт увеличения временной сложности улучшается качество разбиения.

Построение графа данных

Рассмотрим несколько видов графов данных. Все они используют представление метаданных вершины как вектора вещественных чисел заданной размерности. Поскольку во всех из них используется функция расстояния, то меняя метрику в пространстве, можно получать различные графы.

1. Каждая вершина соединяется с вершинами, находящимися на расстоянии не более ε от неё.
2. Граф ближайших соседей. Каждая вершина соединяется с k ближайшими соседями.
3. Граф сфер влияния (sphere of influence graph). Для каждой вершины строится сфера с центром в этой вершине и радиусом, равным её расстоянию до ближайшего соседа. Две точки соединяются ребром, если соответствующие им сферы пересекаются (включая или исключая касание)
4. Минимальное остовное дерево. Рассматривается граф, в котором между каждыми двумя вершинами проведено ребро с весом, равным расстоянию между этими вершинами, и в таком графе находится минимальное остовное дерево.
5. Граф относительного соседства (relative neigbourhood graph). Для каждой пары точек рассмотрим пересечение двух сфер с центрами в этих точках и радиусами, равными расстоянию между ними. Если в этом множестве нет других точек, то данная пара точек соединяется ребром.
6. Граф Габриеля. Для каждой пары точек рассмотрим сферу, построеную на отрезке между ними, как диаметре. Если в этом множестве нет других точек, то данная пара точек соединяется ребром.
7. Триангуляция Делоне. В d–мерном пространстве это такое множество точек, что для каждого симплекса на d вершинах в его описанной гиперсфере нет других точек.

Для второго и третьего графа необходимо выполнить множество запросов вида найти "k ближайших соседей данной точки". Такие запросы могут быть оптимизированы с помощью таких структур данных, как k-d дерево и метрическое дерево (ball tree или metric tree). k-d дерево позволяет, используя предподсчёт за $O(n(d + \log n))$ и $O(nd)$ памяти (где d – размерность пространства), отвечать на запрос о k ближайших соседях за время $O(n^{1-1/d} + k)$. При большом d k-d дерево становится неэффективным, и быстрее работает метрическое дерево[8].

Для первого и третьего графа необходимо выполнить множество запросов вида "найти все точки на расстояни не более R от заданной точки". Оптимизировать такие запросы можно с помощью R-дерева[9].

Заметим, что наивный алгоритм вычисления RNG и графа Габриеля занимает $O(n^3)$, что может быть слишком долго, так как, как упомянуто выше, n обычно принимает большие значения. Существует более оптимальный алгоритм построения RNG, использущий $O(n^{2.5})$ времени[10], однако его недостатком является количество требуемой памяти – $O(n^2)$. Также для частного случая, а именно метрики L_∞ существует алгоритм с временной сложностью $O(n^2 \log n)$[11]. Существуют также быстрые алгоритмы для построения графа Габриеля при размерности векторов 2 или 3[12].

Заключение

Таким образом, для поиска сообществ в сложных сетях наиболее подходят алгоритмы CNM и алгоритм распространения меток (label propagation), как наиболее масштабируемые.

Для построения графов данных необходимо использовать специальные алгоритмы и структуры данных, описанные в основной части, с целью ускорения их построения.

Список литературы

[1] M. E. J. Newman, "The structure and function of complex networks", 2003

[2] Santo Fortunato, Darko Hric, "Community detection in networks: A user guide", 2016

[3] Santo Fortunato, "Community detection in graphs", 2009

[4] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks", 2002

[5] Aaron Clauset, M. E. J. Newman,and Cristopher Moore, "Finding community structure in very large networks", 2004.

[6] Usha Nandini Raghavan, Reka Albert, Soundar Kumara , "Near linear time algorithm to detect community structures in large-scale networks", 2009

[7] Matthias Beck, "Computational Discrete Geometry"

[8] "What is a Good Nearest Neighbors Algorithmfor Finding Similar Patches in Images?", 2009

[9] Antomn Guttman, R-trees, "A dynamic index structure for spatial searching", 1984

[10] Jyrki Katajainen, Olli Nevalainen, An almost naive algorithm for finding relativeneighbourhood graphs in L_p metrics, 1987

[11] Joseph O'Rourke, Computing the relative neighborhood graph in L_1 and L_{inf} metrics, 1981.

[12] Guiseppe Liotta, Low degree algorithms for computing and checking Gabriel graphs