Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра математических методов прогнозирования

Никишин Евгений Сергеевич

Методы выделения сообществ в социальных графах

курсовая работа

Научный руководитель:
д.ф-м.н., профессор
А.Г. Дьяконов

Образец титульного брал отсюда

Москва, 2016

version 0.04.1

Содержание

1 Введение (неполное)

1.1 Модулярность

$$Q = \frac{1}{2m}\sum_{i,j}\left(A_{ij} - \frac{d_i d_j}{2m}\right)\delta(C_i, C_j)$$

Testtest [2, 5, 3, 4, 1]

2 Разбиение на непересекающиеся сообщества

2.1 Edge betweenness

Для каждой пары вершин связного графа можно вычислить кратчайший путь, их соединяющий. Будем считать, что каждый такой путь имеет вес, равный 1/N, где N — число возможных кратчайших путей между выбранной парой вершин. Если такие веса посчитать для всех пар вершин, то каждому ребру можно поставить в соответствие значение Edge betweenness — сумму весов путей, прошедших через это ребро.

Для ясности приведём следующую иллюстрацию:

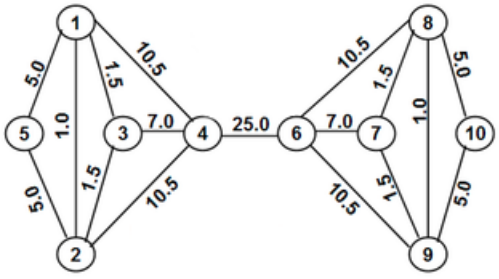

Рис. 1: Граф, для ребёр которого посчитаны значения Edge betweenness

В данном графе хочется выделить два сообщества: с вершинами 1-5 и 6-10. Граница же будет проходить через ребро, имеющее максимальный вес, 25. На этой идее и

основывается алгоритм: поэтапно удаляем ребра с наибольшим весом, а оставшиеся компоненты связности объявляем сообществами.

Собственно, сам алгоритм:

1. Инициализировать веса
2. Удалить ребро с наибольшим весом
3. Пересчитать веса для ребёр
4. Сообществами считаются все компоненты связности
5. Посчитать функционал модулярности (о нём будет сказано ранее)
6. Повторять с шаги 2-6, пока есть рёбра

На каждой итерации процесса получается некое разбиение вершин. Последовательность таких разбиений, имеющая вид дерева, в листьях которого находятся сообщества с одной вершиной, а в корне — большое сообщество, содержащее все вершины, называется дендрограммой. Результатом работы алгоритма является ярус дендрограммы (т.е. разбиение), имеющий максимальную модулярность.

Из необходимости каждый раз пересчитывать веса следует главный минус: вычислительная сложность в худшем случае составляет $O(m^2n)$, где m — количество ребёр, n — количество вершин. Эксперименты показывают, что пересчитывать обычно приходится только веса для рёбер, которые были в одной компоненте связности, что несколько уменьшает сложность, однако зачастую этого оказывается недостаточно.

2.2 Label propagation

Допустим, что большинство соседей какой-либо вершины принадлежат одному сообществу. Тогда, с высокой вероятностью, ему также будет принадлежать выбранная вершина. На этом предположении и строится алгоритм Label propagation: каждая вершина в графе определяется в то сообщество, которому принадлежит большинство его соседей. Если же таких сообществ несколько, то выбирается случайно одно из них. Пример:

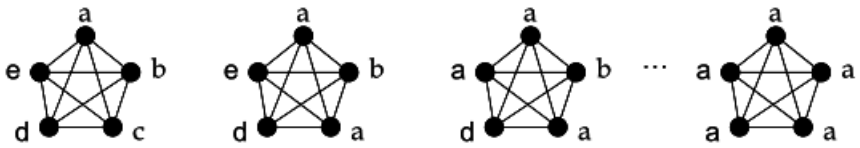

Рис. 2: Демонстрация работы алгоритма для полного графа

В начальный момент времени всем вершинам ставится в соответствие отдельное сообщество. Затем происходят перераспределения сообществ. Из-за случайности важно на каждой итерации изменять порядок обхода вершин. Алгоритм заканчивает работу, когда нечего изменять: все вершины относятся к тем сообществам, что и

большинство их соседей. Авторы также советуют запускать несколько раз алгоритм и выбирать наилучшее из результирующих разбиений, либо пересекать их. Главное достоинство данного алгоритма, в противовес предыдущему, — почти линейная сложность. Однако на зашумленных графах зачастую происходит объединение всех вершин в одно сообщество.

2.3 FastGreedy

Алгоритм заключается в жадной оптимизации модулярности. Как и в прошлом методе, в каждой вершине графа инициализируется отдельное сообщество, а затем объединяются пары сообществ, приводящие к максимальному увеличению модулярности. При этом объединяются только инцидентные пары вершин, так как, в противном случае, модулярность не может увеличиться [во введении необходимо будет объяснить смысл модулярности, чтобы этот факт не вызывал вопросов].

Результатом работы алгоритма будет ярус дендрограммы, на котором модулярность максимальна.

Метод является вычислительно нетрудоёмким ($O(m \log n)$), легко применим к большим графам и, несмотря на жадность, зачастую неплохо справляется с задачей.

2.4 WalkTrap

Допустим, на вершинах графа задана такая метрика, что между двумя вершинами из разных сообществ расстояние велико, а из одного — мало. Тогда выделение сообществ можно рассматривать как задачу кластеризации вершин. Попытаемся ввести такую метрику, используя случайные блуждания. Объект может переместиться из вершины i в вершину j с вероятностью $P_{ij} = \frac{A_{ij}}{d_i}$, где A — матрица смежности, d_i — степень i. То есть на каждом шаге равновероятно выбирается "сосед" вершины i. Таким образом определяется матрица переходов P случайного блуждания. Она примечательна тем, что её степени являются вероятностями перехода из одной вершины в другую за соответствующее число шагов: вероятность перехода из i в j за t шагов равна $(P^t)_{ij}$. Также следует отметить, что $P = D^{-1}A$, где D — матрица со степенями вершин на диагонали. Используя этот аппарат можно ввести желаемую метрику на вершинах:

$$r_{ij} = \sqrt{\sum_{k=1}^{n} \frac{(P^t_{ik} - P^t_{jk})^2}{d(k)}} = \left\| D^{-\frac{1}{2}} P^t_{i\bullet} - D^{-\frac{1}{2}} P^t_{j\bullet} \right\|,$$

где $P^t_{i\bullet}$ — вектор из вероятностей перехода за t шагов из вершины i во все другие. Вообще говоря, метрика зависит от t, авторы советуют брать $3 \leq t \leq 8$.

Естественным образом расстояние между вершинами обобщается на расстояние между сообществами:

$$r_{_1C_2} = \left\| D^{-\frac{1}{2}} P^t_{C_1\bullet} - D^{-\frac{1}{2}} P^t_{C_2\bullet} \right\| = \sqrt{\sum_{k=1}^{n} \frac{(P^t_{C_1k} - P^t_{C_2k})^2}{d(k)}},$$

где

$$P^t_{Cj} = \frac{1}{|C|} \sum_{i \in C} P^t_{ij}$$

Теперь, когда задана метрика, можно попытаться выделить кластеры в графе. Начальное разбиение — по одной вершине в каждом кластере $\mathcal{P}_1 = \{\{v\}, v \in V\}$. Также для всех пар инцидентных вершин считается расстояние. Далее для каждого k:

1. Выбрать C_1 и C_2 из $\mathcal{P}_k$ согласно некоторому метрическому критерию.
2. Объединить два сообщества в новое $C_3 = C_1 \cup C_2$ и обновить разбиение $\mathcal{P}_{k+1} = (\mathcal{P}_k \setminus \{C_1, C_2\}) \cup C_3$.
3. Обновить расстояния между инцидентными сообществами.

После $n-1$ шага получается дендрограмма разбиений, а $\mathcal{P}_n = \{V\}$. Таким образом, остался неясным только критерий выбора пар сообществ на шаге 1. Будем выбирать пару сообществ, минимизирующих приращение среднего квадратов расстояний между каждой вершиной и их сообществом при объединении сообществ. Т.е.

$$\Delta\sigma(C_1, C_2) = \frac{1}{n}\left(\sum_{i \in C_3} r_{iC_3}^2 - \sum_{i \in C_1} r_{iC_1}^2 - \sum_{i \in C_2} r_{iC_2}^2\right) \to \min_{C_1, C_2}$$

Теперь осталось только получить результат, выбрав разбиение, на котором достигает максимума модулярность.

2do: разобраться с русской кодировкой и правильным оформлением списка литературы

Список литературы

[1] Clauset, A. Finding community structure in very large networks / Aaron Clauset, M. E. J. Newman, Cristopher Moore // Physical Review E. — 2004. — http://arxiv.org/abs/cond-mat/0408187.

[2] Girvan, M. Community structure in social and biological networks / Michelle Girvan, M. E. J. Newman // Proceedings of the National Academy of Sciences. — 2001. — http://arxiv.org/abs/cond-mat/0112110.

[3] igraph library. — 2016. — http://igraph.org/python/.

[4] Raghavan, U. N. Near linear time algorithm to detect community structures in large-scale networks / Usha Nandini Raghavan, Reka Albert, Soundar Kumara // Physical Review E. — 2007. — http://arxiv.org/abs/0709.2938.

[5] Slavnov, K. A. Social graph analysis. — 2015. — http://www.machinelearning.ru/wiki/images/6/60/2015_417_SlavnovKA.pdf.