Глубинное обучение в графовых данных

2. Доглубинные подходы

Высшая Школа Экономики 2022/2023

в предыдущих сериях…

Графы как инструмент анализа данных

+ в виде графа можно представить очень много реальных систем
+ помогают в работе с непоследовательными данными

- очень сложная структура
- требуют сильно подумать о том как представлять данные

Типы задач на графах

1. Node Level
2. Edge Level
3. Graph Level

Создание признаков вручную

До глубинного обучения

Прежде чем переходить к основным методам глубинного обучения следует ознакомиться с существующими подходами для создания так называемых hand-crafted признаков

По аналогии с типами существующими задач для анализа графовых данных, будем разбирать признаки для вершин, ребер и графов

ВАЖНО: Предполагаем, что все графы, разбираемые в лекции будут неориентированными

Постановка задачи машинного обучения на графах

Задача: сделать предсказание для ряда объектов

Признаки:

Объекты:

Функция потерь:

Постановка задачи

Задача: сделать предсказание для ряда объектов

Признаки: *n*-мерные векторы

Объекты: вершины, ребра, графы

Функция потерь: Зависит от задачи

Node-level statistics

- степень вершины
- центральность
- коэффициент кластеризации
- графлеты

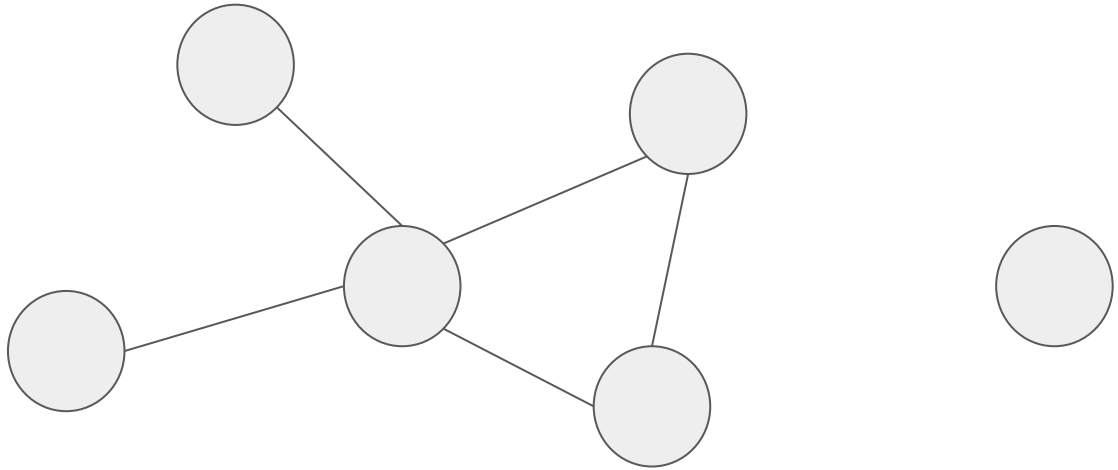
Степень вершины

Определение: степенью вершины называется число ребер, инцидентных этой вершине

В случае неориентированных графов без петлей можно считать что степенью вершины называется количество соседей этой вершины

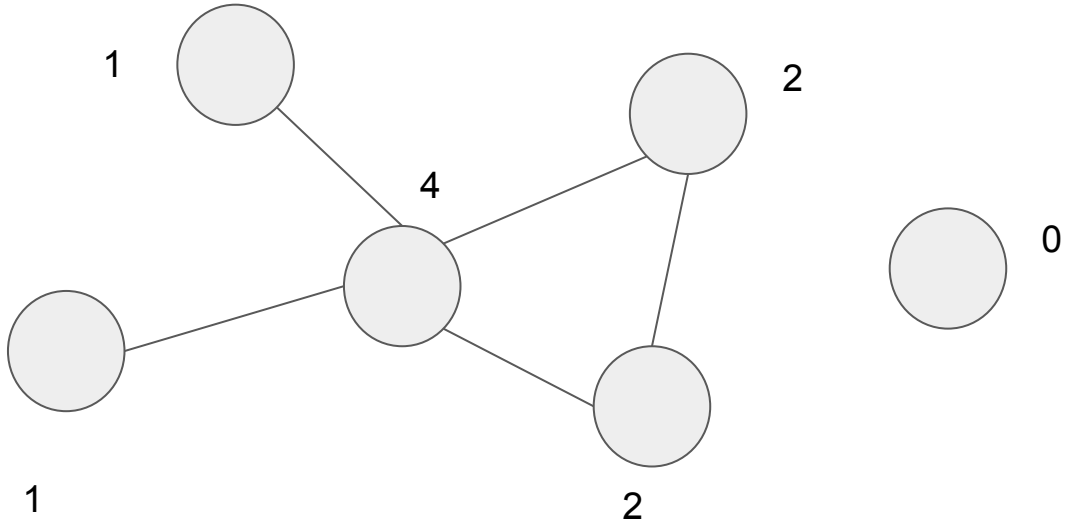
$$d_u = \sum_{v \in V} \mathbf{A}[u, v]$$

Степень вершины



Степень вершины

Основное свойство этой статистики - все соседние вершины котируются одинаково



Центральность

Степень вершины не позволяет понять **значимость** вершины в контексте графа

Для более точного понимания значения вершины в графе можно посмотреть на **меру центральности**

Существую различные способы посмотреть на эту статистику:

- с точки зрения собственных векторов
- с точки зрения соседства
- с точки зрения близости

Центральность с помощью собственных значений

Определение: центральностью называется сумма центральностей всех соседей

Определение рекурсивное - не очень удобное, нужно аналитическое

$$e_u = \frac{1}{\lambda} \sum_{v \in V} \mathbf{A}[u, v] e_v \; \forall u \in \mathcal{V},$$

Центральностью с помощью собственных значений

Если определить **e** как вектор центральностей вершин, то можно перейти к стандартному уравнению для собственных значений матрицы смежности

Получается **e** - собственный вектор матрицы смежности **A**, для

$$\lambda \mathbf{e} = \mathbf{A}\mathbf{e}$$

Центральность с помощью соседства

Можно сделать предположение, что вершина **v** важная если она лежит на большом количестве кратчайших путей между другими вершинами, которые содержат эту вершину

$$c_v = \sum_{s \neq v \neq t} \frac{\text{количество кратчайших путей между s и t содержащих v}}{\text{количество кратчайших путей между s и t}}$$

Центральность с помощью близости

Можно предположить что вершина важная, если она находится очень близко ко всем вершинам

$$c_v = \frac{1}{\sum_{u \neq v}}$$

длина кратчайшего пути между u и v

Коэффициент кластеризации

Для определения сообществ можно посчитать статистику для вершины, показывающую соединенность для всех соседей вершины

$$c_u = \frac{|(v_1, v_2) \in \mathcal{E} \ : \ v_1, v_2 \in \mathcal{N}(u)|}{\binom{d_u}{2}}.$$

От коэффициента кластеризации к графлетам

Если заметить, то коэффициент кластеризации считает число “треугольников”

Возникает закономерный вопрос - а что будет если считать не только треугольники?

Графлеты

Графлетами называется подграфы, описывающие структуру сети соседей вершины

По сути, можно сказать что это обобщение для предыдущих способов описания признаков:

- степень графа это количество ребер, которые связаны с вершиной
- коэффициент кластеризации это количество треугольников, которые связаны с вершиной

Графлеты

Различных графлетов размера 2 - 5 существует 73 (в то время как неизоморфных графов 30), соответственно вектор графлетов размером это 73 числа показывающее, сколько графлектом под определенным номером касается наша вершина (размер, естественно, не ограничен пятью)

Смысл графлетной степени вершины - это дополнительная информация о локальной топологии

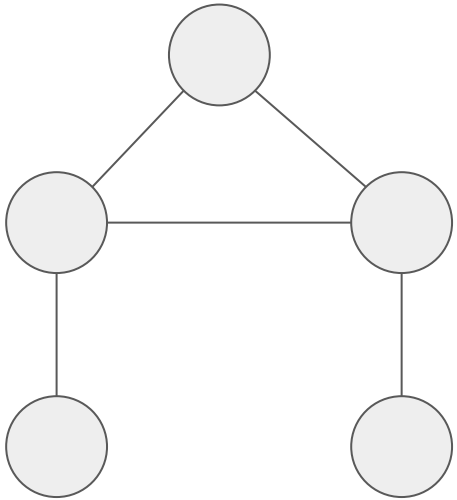
Графлеты

Определение: порожденный подграф - подмножество вершин и ребер графа, где вершины соединены если они соединены в оригинальном графе

Определение: изоморфные графы - 2 графа называются изоморфными если у них одинаковое число вершин соединенных одинаково

Определение: графлеты - соединенные неизоморфные порожденные подграфы

Графлеты. Пример



Предсказание пропущенных связей

Дано: набор ребер

Выход: новые ребра

Как придумать признаки для пары вершин?

Предсказание пропущенных связей. 2 типа задачи

- **Случайное удаление ребер** :

  Часть вершин удаляются из графа и ставится цель предсказать их появления

- **Ребра с течением времени**

  Имея момент времени t_0 и t’_0 для графа вывести отранжированный список вершин для моментов времени t_1 и t’_1

Предсказание пропущенных связей

Основная идея состоит в следующем подходе

- считаем для каждой вершин в графе пары score
- ранжируем
- лучшие n предсказываем как новые связи
- проверяем, какие на самом деле появились в итоге состояний графа в t_1 и t’_2

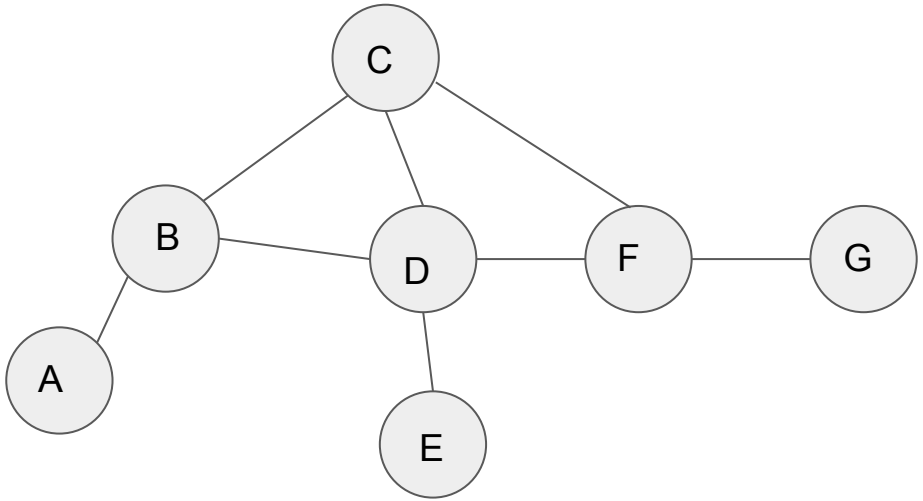
Предсказание пропущенных связей : признаки

Рассмотрим следующие виды признаков

- дистанция
- локальное пересечение
- глобальное пересечение

Дистанция

Дистанцией между парой вершин можно считать длину кратчайшего пути между ними



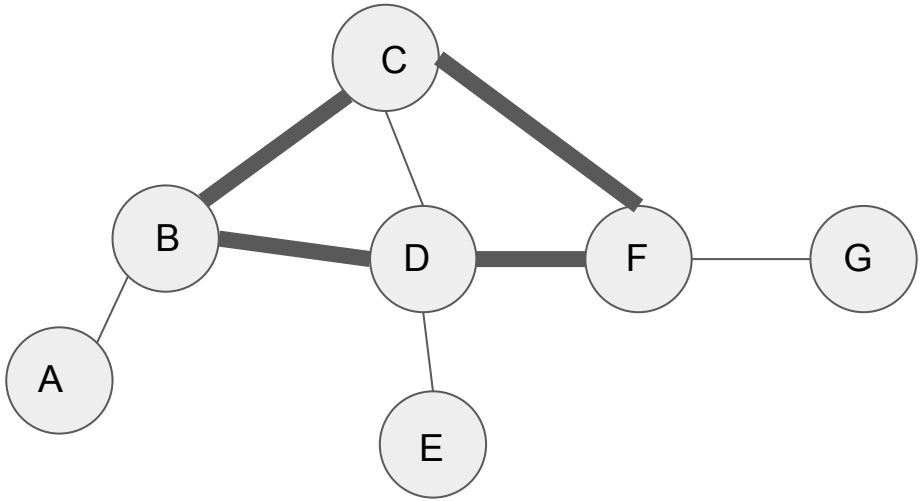
AC = CE = 2

BF = 2

BG = 3

Дистанция

Кратчайшие пути BE и BF равны, но из B в F можно существует два кратчайших пути. Из этого можно извлечь больше информации.



Локальное пересечение

Имеет смысл смотреть на множества соседей двух вершин.

- **Общие соседи**

$$\mathbf{S}[u, v] = |\mathcal{N}(u) \cap \mathcal{N}(v)|$$

- **Коэффициент Жаккара**

$$\mathbf{S}_{\text{Jaccard}}[u, v] = \frac{|\mathcal{N}(u) \cap \mathcal{N}(v)|}{|\mathcal{N}(u) \cup \mathcal{N}(v)|}$$

- **Индекс Адамика-Адара**

$$\mathbf{S}_{\text{AA}}[v_1, v_2] = \sum_{u \in \mathcal{N}(v_1) \cap \mathcal{N}(v_2)} \frac{1}{\log(d_u)}$$

Проблемы локального пересечения

В случае если вершины не пересекаются, то значения метрики выбранной для локального пересечения будут равны нулю.

Это имеет смысл с точки зрения текущего состояния, но в случае изменения графа вершины могут иметь связь через некоторое время. Для большего количества информации следует смотреть на граф целиком.

Глобальное пересечение

На что имеет смысл смотреть для произвольного графа, если мы хотим посмотреть на связь двух далеких друг от друга вершин?

Напрашивается - посчитать количество всех путей между двумя вершинами. **(Индекс Каца)**

$$\mathbf{S}_{\mathrm{Katz}}[u, v] = \sum_{i=1}^{\infty} \beta^i \mathbf{A}^i[u, v],$$

На уровне графа

Отдельно разобрав признаки, которые характеризуют вершины и связи, пришло время разобраться, как можно охарактеризовать граф

Один из вариантов - взять просто агрегировать статистики вершин (в теории неплохо для старта)

На помощь приходят ядра. После его задания можно вернуться к классическому машинному обучению (kernel SVM).

- Графлетовые ядра
- Weisfeiler - Lehman kernel

Графлетовые ядра

Основная идея - позаимствовать идею мешка слов, заменив его на мешок вершин (bag of nodes).

Для создания графлетового ядра примем следующие уточнения:

- вершины в графлетах не должны быть соединены
- нет корневой вершины

Графлетовые ядра

Вектор количества графлетов - вектор где указаны количества графлетов в сети

$$K(G, G') = \boldsymbol{f}_G{}^{\mathrm{T}} \boldsymbol{f}_{G'}$$

Графлетовые ядра

При разных G, G’ - проблема с результатом

Нужна нормализация

$$\boldsymbol{h}_G = \frac{\boldsymbol{f}_G}{\mathrm{Sum}(\boldsymbol{f}_G)} \qquad K(G, G') = \boldsymbol{h}_G{}^{\mathrm{T}} \boldsymbol{h}_{G'}$$

Графлетовые ядра. Проблемы

Очень дорого.

Для размера k графа размера n цена будет n^k

Почему? По причине того что изоморфизм подграфов это NP-hard

Если степень вершины сверху ограничена d то nd^k

Weisfeiler-Lehman Kernel

Идея алгоритма - итеративная аггрегация по соседям

Алгоритм:

1. Назначим каждой вершине метку
2. $l^{(i)}(v) = \text{HASH}(\{\{l^{(i-1)}(u)\ \forall u \in \mathcal{N}(v)\}\})$
3. после k шагов этот вектор суммаризует структуру о k-hop соседстве

WL Kernel

Хороший способ - вычислительно эффективен

Вычисляя ядро требуется следить только за цветами, появляющимися в двух графах

Считать цвета - линия

В общем получается линия по ребрам