

Машинное обучение (Machine Learning)

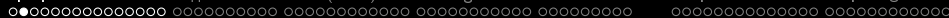
Графовая нейронная сеть (Graph Neural Network)

Уткин Л.В.

Санкт-Петербургский политехнический университет Петра Великого



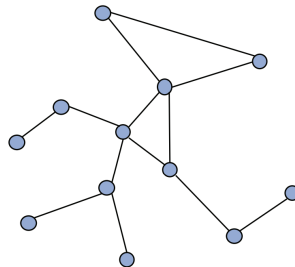
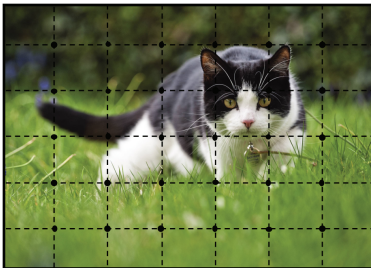
ПОЛИТЕХ
Санкт-Петербургский
политехнический университет
Петра Великого



Графы

- Граф - это структура данных, состоящая из двух компонентов: вершин и ребер. Граф G описывается набором вершин V и ребер E , которые он содержит:
 $G = (V, E)$.
- Ребра могут быть как направленными, так и ненаправленными, в зависимости от того, существуют ли направленные зависимости между вершинами.
- Вершины часто называют узлами.

Картинки как графы (1)



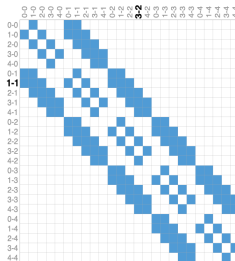
Zhiyuan Liu, Jie Zhou. Introduction to Graph Neural Networks.

DOI:10.2200/s00980ed1v01y202001aim045

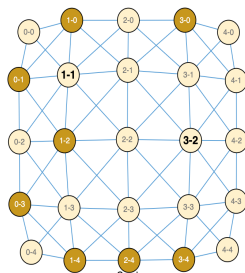
Картинки как графы (3)

0-0	1-0	2-0	3-0	4-0
0-1	1-1	2-1	3-1	4-1
0-2	1-2	2-2	3-2	4-2
0-3	1-3	2-3	3-3	4-3
0-4	1-4	2-4	3-4	4-4

Image Pixels



Adjacency Matrix



Graph

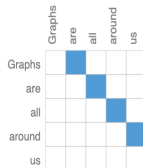
<https://distill.pub/2021/gnn-intro/>

Картинки как графы (4)

- Упорядочиваем узлы, в данном случае каждый из 25 пикселей в простом изображении смайлика 5x5, и заполняем матрицу $n_{nodes} \times n_{nodes}$, если два узла имеют общее ребро. Каждое из этих трех представлений есть разные представления одного и того же фрагмента данных.

Текст как граф

- Можно оцифровывать текст, связывая индексы с каждым символом, словом или токеном и представляя текст как последовательность этих индексов.
- Это создает простой ориентированный граф, где каждый символ или индекс является узлом и соединен ребром с узлом, следующим за ним.



<https://distill.pub/2021/gnn-intro/>

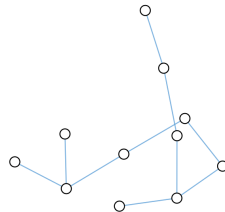
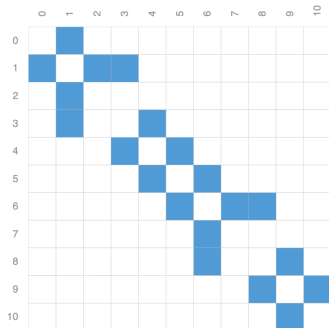
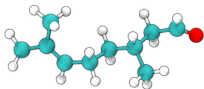
А что на практике

На практике обычно текст и изображения кодируются не так: эти графические представления избыточны, поскольку все изображения и весь текст будут иметь очень регулярную структуру. Например, изображения имеют полосчатую структуру в своей матрице смежности, потому что все узлы (пиксели) связаны в сетке. Матрица смежности для текста — это просто диагональная линия, потому что каждое слово соединяется только с предыдущим словом и со следующим.

Молекулы как графы (1)

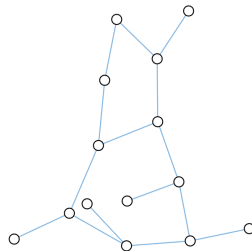
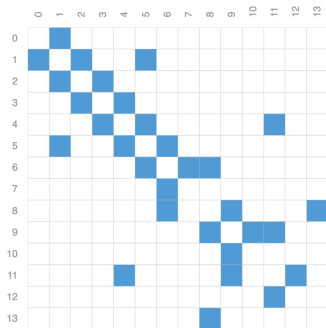
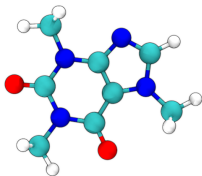
- Молекулы состоят из атомов и электронов в трехмерном пространстве.
- Частицы молекулы взаимодействуют, но когда пара атомов застревает на стабильном расстоянии друг от друга, мы говорим, что они связаны ковалентной связью.
- Разные пары атомов и связей имеют разное расстояние (например, одинарные связи, двойные связи).
- Это очень удобная и распространенная абстракция для описания трехмерного объекта в виде графа, где узлы — это атомы, а ребра - ковалентные связи.

Молекулы как графы (2)



<https://distill.pub/2021/gnn-intro/>

Молекулы как графы (3)

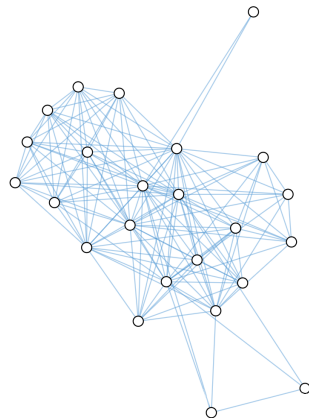
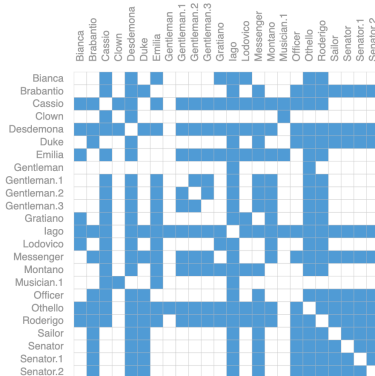


<https://distill.pub/2021/gnn-intro/>

Социальные сети как графы (1)

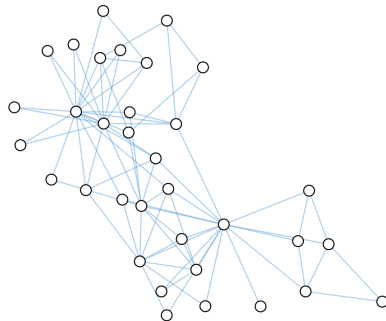
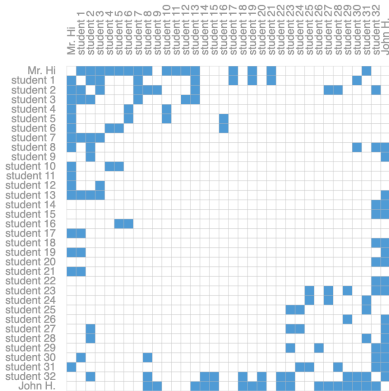
- Можно построить граф, представляющий группы людей, моделируя отдельных людей как узлы, а их отношения — как ребра.
- В отличие от изображений и текстовых данных, социальные сети не имеют одинаковых матриц смежности.

Социальные сети как графы (2)



Сеть сцен из пьесы “Отелло” (<https://distill.pub/2021/gnn-intro/>)

Социальные сети как графы (3)



Взаимодействие между людьми в клубе карате

(<https://distill.pub/2021/gnn-intro/>)

Сеть цитирований и другое как графы

- Можно визуализировать сети цитирований статей в виде графа, где каждая статья является узлом, а каждое направленное ребро — цитированием одной статьей другой. Можно добавить информацию о каждой статье в узел, например реферат.
- В компьютерном зрении размечают объекты в визуальных сценах. Можем строить графы, рассматривая эти объекты как узлы, а их отношения — как ребра.
- Модели машинного обучения, программный код и математические уравнения также можно представить в виде графов, где переменные - узлы, а ребра - операции.

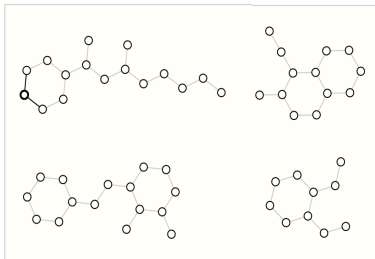
Задачи

Задачи для графовых данных

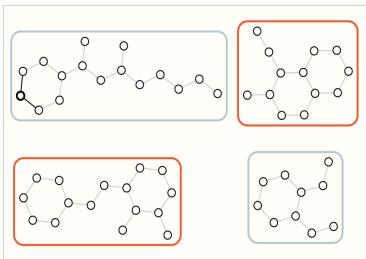
Три типа задач для графовых данных

- Существует три основных типа задач предсказания:
на уровне графа, на уровне узла и на уровне ребра.
 - 1 На уровне графа предсказываем одно свойство для всего графа.
 - 2 На уровне узла - некоторое свойство для каждого узла в графе.
 - 3 На уровне ребер - свойство или наличие ребер в графе.
- Для всех трех уровней задачи могут быть решены с помощью GNN.

Задача на уровне графа (2)



Input: графы

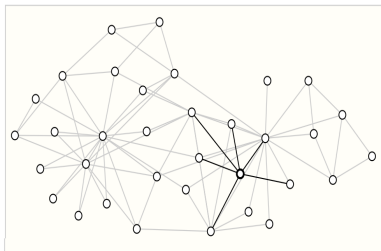


Output: метки для каждого графа, например "содержит ли граф 2 кольца"

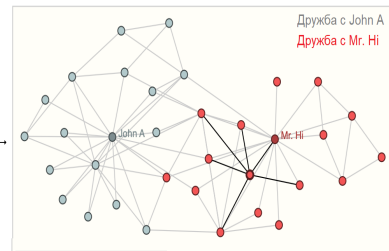
Задача на уровне узлов (1)

- Уровень узлов связан с предсказанием личности или роли каждого узла в графе.
- Пример задачи прогнозирования на уровне узла - клуб. Набор данных представляет собой единый граф социальной сети, состоящий из людей, которые находятся в дружбе или вражде с одним из двух клубов после раскола.
- Задача прогнозирования состоит в том, чтобы определить, состоит ли данный член в дружбе с Mr. Hi либо с John H. после раскола.
- В этом случае расстояние между узлом до Mr. Hi или John H. сильно коррелирует с этой меткой.
- В изображении, проблемы прогнозирования на уровне узлов аналогичны сегментации изображения, где пытаемся обозначить роль каждого пикселя в изображении.

Задача на уровне узлов (2)



Input: Граф с узлами без меток

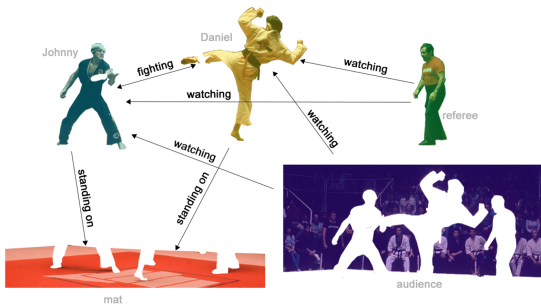


Output: Метки узлов

Задача на уровне ребер (1)

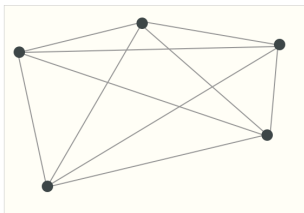
- Пример - понимание сцены изображения.
- Помимо идентификации объектов на изображении, модели глубокого обучения можно использовать для прогнозирования взаимосвязи между ними.
- Можно сформулировать это как классификацию на уровне ребер: узлы - объекты на изображении, нужно предсказать, какие из этих узлов имеют общее ребро или каково значение этого ребра.

Задача на уровне ребер (2)

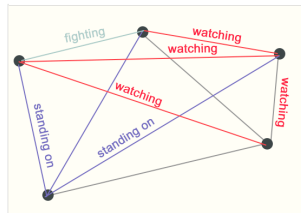


<https://distill.pub/2021/gnn-intro/>

Задача на уровне ребер (3)



Input: полный граф,
неразмеченные ребра



Output: метки ребер

<https://distill.pub/2021/gnn-intro/>

Графовая нейронная сеть (Graph Neural Network)

- Графовая нейронная сеть (GNN) - тип нейронной сети, который работает со структурой графа.
- Типичное применение GNN - классификация узлов.
- По сути, каждый узел в графе связан с меткой, и мы хотим предсказать метку узлов с некоторой вероятностью.
- В постановке задачи классификации узлов каждый узел v характеризуется своим признаком \mathbf{x}_v и связан с меткой t_v .

GNN

Графовая нейронная сеть - основные понятия

Графовая нейронная сеть (Graph Neural Network)

- Дан частично размеченный граф G , цель - использовать размеченные узлы для предсказания меток неразмеченных.
- Он учится представлять каждый узел v с помощью d -мерного вектора (эмбединга) \mathbf{h}_v , который содержит информацию о его окрестности.

Состояние встраивания (1)

- Пусть f - параметрическая функция, называемая локальной переходной функцией, которая является общей для всех узлов и обновляет состояние узла в соответствии с входной окрестностью.
- Пусть g - локальная выходная функция, которая описывает, как выход был произведен.
- Тогда \mathbf{h}_v и o_v (метка узла) определяются следующим образом:

$$\mathbf{h}_v = f(\mathbf{x}_v, \mathbf{x}_{\text{co}[v]}, \mathbf{h}_{\text{ne}[v]}, \mathbf{x}_{\text{ne}[v]})$$

$$o_v = g(\mathbf{h}_v, \mathbf{x}_v)$$

Иллюстрация эмбединга узла

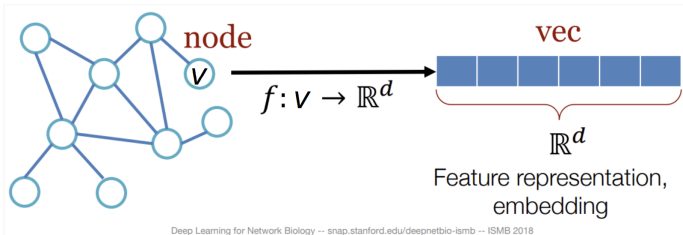
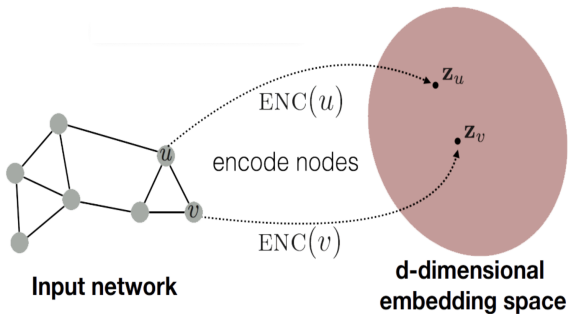


Иллюстрация цели



Уравнения для глобальных функций

- Пусть \mathbf{H} , \mathbf{O} , \mathbf{X} и \mathbf{X}_N - это векторы, построенные путем конкатенации всех эмбедингов \mathbf{h}_v , всех выходных данных o_v , всех признаков \mathbf{x}_v и всех признаков узлов $\mathbf{x}_{ne[v]}$ соответственно. Тогда можно представить уравнения в более компактной форме:

$$\mathbf{H} = F(\mathbf{H}, \mathbf{X})$$

$$\mathbf{O} = G(\mathbf{H}, \mathbf{X}_N)$$

- F - глобальная функция перехода, а G - глобальная выходная функция, которые являются конкатенацией f и g для всех узлов в графе соответственно.
- Значение \mathbf{H} является фиксированным и однозначно определяется в предположении, что F - это карта сжатия.

Итерационный процесс обновления (1)

- Т.к. ищем уникальное решение для \mathbf{h}_v , можем применить теорему Банаха о неподвижной точке и переписать приведенное выше уравнение как итерационный процесс обновления

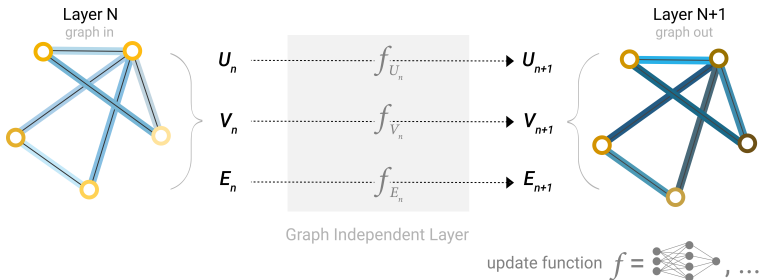
$$\mathbf{H}^{t+1} = F(\mathbf{H}^t, \mathbf{X})$$

- Такая операция часто называется передачей сообщений или объединением соседей.
- \mathbf{H}^t обозначает t -ую итерацию \mathbf{H} . Динамическая система из этого уравнения сходится экспоненциально быстро к решению уравнения $\mathbf{H} = F(\mathbf{H}, \mathbf{X})$ для любого начального значения \mathbf{H}^0 .

GNN в общем

- GNN использует отдельный многослойный перцептрон (MLP) для каждого компонента графа: называем это слоем GNN.
- Для каждого вектора узла применяем MLP и возвращаем обученный вектор узла.
- Делаем то же самое для каждого ребра, обучая эмбединги для каждого ребра,
- а также для вектора глобального контекста, обучая один эмбединг для всего графа.

GNN в общем

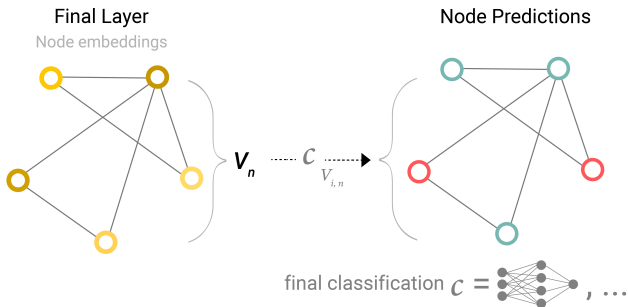


Один слой простой GNN. Граф является входом, и каждый компонент (V , E , U) обновляется MLP для создания нового графа.

Pooling

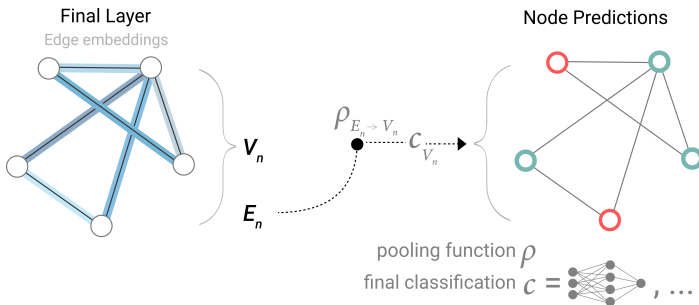
Pooling

GNN с pooling (1)



GNN с pooling и признаками ребер

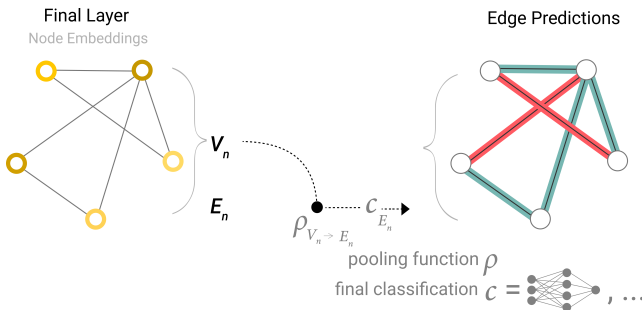
Если у нас есть только признаки уровня ребер и надо предсказать информацию о двоичном узле, можно также использовать pooling для направления информации туда, куда она должна идти. Модель выглядит так



GNN с pooling и признаками узлов

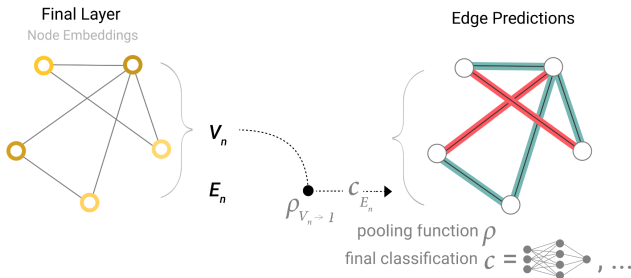
Если у нас есть только признаки уровня узлов и нужно предсказать бинарную информацию на уровне ребер, модель выглядет следующим образом.

Узлы могут быть распознаны как объекты изображения, и мы пытаемся предсказать, имеют ли объекты общие отношения (бинарные ребра).



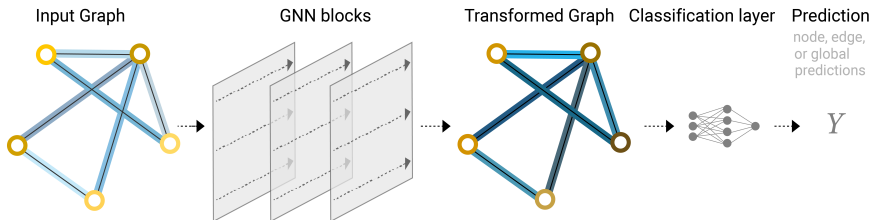
GNN с pooling и признаками узлов

Если у нас есть только признаки уровня узлов и нужно предсказать бинарное глобальное свойство, нам нужно собрать всю доступную информацию об узле и агрегировать ее. Это похоже на слой Global Average Pooling в CNN. То же самое можно сделать и с ребрами.



GNN с pooling

В примерах классификационная модель сможет быть легко заменена любой дифференцируемой моделью или адаптирована к многоклассовой классификации с использованием обобщенной линейной модели.



GNN и pooling

- Метод объединения (pooling) послужит строительным блоком для построения более сложных моделей GNN.
- Если есть новые атрибуты графа, просто нужно определить, как передавать информацию из одного атрибута в другой.
- В простейшей формулировке GNN не использовалась связность графа внутри слоя GNN.
- Каждый узел обрабатывается независимо, как и каждое ребро, а также глобальный контекст.

GNN: message passing между частями графа (1)

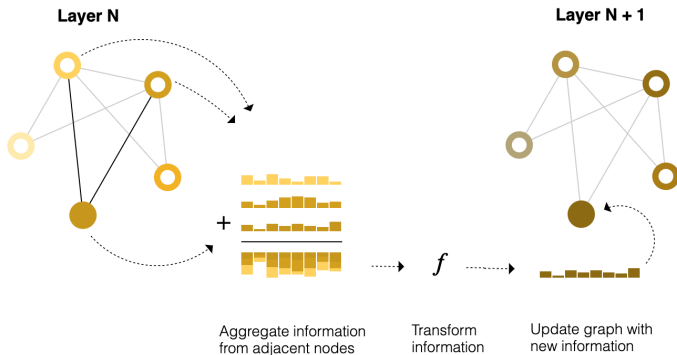
- Идея message passing проста - вершины графа посылают сообщения, и новое представление каждой вершины получается как функция Φ от:
 - Предыдущего представления вершины
 - Сообщений соседей
- $\mathbf{h}_v^{(k)} = \Phi(\mathbf{h}_v^{(k-1)}, \mathbf{X}_{\mathcal{N}(v)}^{(k)})$, где $\mathbf{h}_v^{(k-1)}$ – предыдущее представление вершины,
 $\mathbf{X}_{\mathcal{N}(v)}^{(k)} = \text{AGG}^{(k)}(\mathbf{h}_u^{(k-1)}, u \in \mathcal{N}(v))$ - представления соседей
- Message passing - процесс итеративной агрегации соседей (neighborhood aggregation).

GNN: message passing между частями графа (2)

- Message passing работает в три этапа:
 - 1 Каждый узел в графе вычисляет **message** для каждого из своих соседей. Сообщения являются функцией узла, соседа и ребра между ними.
 - 2 Сообщения отправляются, и каждый узел **агрегирует** полученные сообщения, используя инвариантную к перестановке функцию (т.е. не имеет значения, в каком порядке получены сообщения). Эта функция обычно является суммой или средним значением.
 - 3 После получения сообщений каждый узел **обновляет (updates)** свои атрибуты в зависимости от своих текущих атрибутов и агрегированных сообщений.

Эта процедура происходит синхронно для всех узлов в графе, так что на каждом шаге передачи сообщения все узлы обновляются.

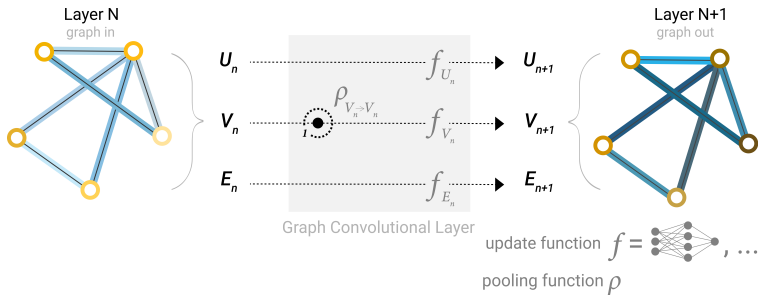
GNN: message passing (1)



GNN: message passing (3)

- Эта послед-сть операций, примененная один раз, есть простейший тип message passing GNN.
- Это как стандартная свертка: message passing и свертка - это операции по агрегированию и обработке информации о соседях элемента для обновления значения элемента. В графах элемент - это узел, а в изображениях - пиксель. Однако количество соседних узлов в графе может быть переменным, в отличие от изображения, где каждый пиксель имеет заданное количество соседей.
- Путем объединения слоев GNN, передающих сообщения, узел может включать информацию со всего графа: после трех слоев узел получает информацию об узлах, находящихся на расстоянии трех шагов от него.

GNN: message passing (4)



Алгоритм DeepWalk (2)

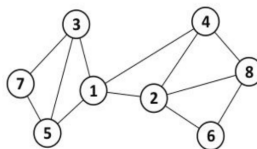
- Начиная с целевого узла (корня), случайным образом выбираем узел, соседний данному и достраиваем до него путь.
- Далее случайным образом выбираем узел, соседний тому, что нашли на предыдущем шаге, достраиваем до него путь и так далее, пока не будет сделано заданное количество шагов.

Алгоритм DeepWalk - более формально

- Генератор случайных блужданий в графе G отбирает равномерно случайную вершину v_i как корень случайного блуждания \mathcal{W}_{v_i} .
- На каждом шаге случайного блуждания следующий узел равномерно выбирается из соседей предыдущего узла. Затем каждая последовательность разрезается на подпоследовательности длиной $2|w| + 1$, где w - размер окна в SkipGram.
- Т.о. SkipGram перебирает все возможные словосочетания в случайном блуждании, которые появляются в окне w . Для каждого мы сопоставляем каждую вершину v_j ее текущему вектору представления $\Phi(v_j) \in \mathbb{R}^d$. Имея представление v_j , максимизируем вероятность его соседей в блуждании.

DeepWalk - случайные блуждания

случайные блуждания по графу + word2vec



(1) Random walk

Corpus

#seq.	traveling path
seq.1	1,3,7,3,5,7,5,1
seq.2	4,8,4,2,8,4,1,4
seq.3	5,7,5,1,5,1,2,1
seq.4	8,6,2,6,2,6,8,6
...	...
seq.32	1,3,1,5,7,5,1,5

(2) Sliding window

Training set

target	context
3	1, 7
7	3
3	7, 5
5	3, 7
7	5
...	...

$$\frac{1}{|V|} \sum_{i=1}^{|V|} \sum_{-c \leq j \leq c} \log p(v_{i+j} | Y_i) \rightarrow \max,$$

$$p(v_{i+j} | Y_i) = \frac{\exp(Y_{i+j}^T Y_i)}{\sum_{k=1}^{|V|} \exp(Y_k^T Y_i)}$$

Алгоритм DeepWalk - softmax

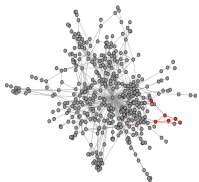
- Иерархический softmax применяется для вычислений softmax из-за огромного количества узлов. Чтобы вычислить значение softmax для каждого отдельного выходного элемента, мы должны вычислить все $\exp(x_k)$ для всех элементов k :

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{k=1}^K \exp(x_k)}$$

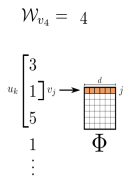
Иерархический softmax (1)

- Иерархический softmax использует бинарное дерево. Все листья v_1, v_2, \dots, v_8 являются вершинами графа.
- В каждом внутреннем узле есть двоичный классификатор, чтобы решить, какой путь выбрать. Чтобы вычислить вероятность вершины v_k , нужно вычислить вероятность каждого подпути на пути от корня до листа v_k .
- Т.к. вероятность дочерних элементов каждого узла равна 1, свойство суммы вероятностей всех вершин, равное 1, сохраняется в иерархическом softmax.
- Время вычисления элемента уменьшено до $O(\log |V|)$, так как самый длинный путь для бинарного дерева ограничен $O(\log(n))$, где n - количество листьев.

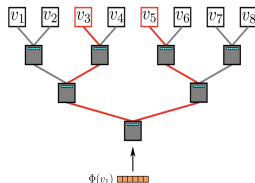
Иерархический softmax (2)



(a) Random walk generation.



(b) Representation mapping.



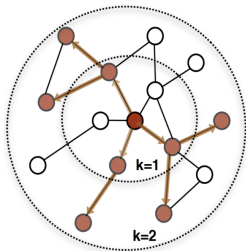
(c) Hierarchical Softmax.

Окно длины $2w + 1$ скользит по случайному блужданию \mathcal{W}_{v_4} , отображая центральную вершину v_1 в ее представление $\Phi(v_1)$. Иерархический Softmax факторизует $\Pr(v_3|\Phi(v_1))$ и $\Pr(v_5|\Phi(v_1))$, соответствующих путям, начинающимся в корне и заканчивающимся в v_3 и v_5 . Представление Φ обновляется, чтобы максимизировать вероятность совпадения v_1 с его контекстом $\{v_3, v_5\}$.

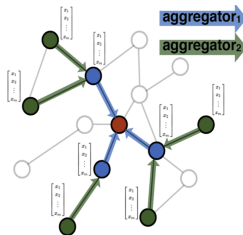
GraphSage и соседи узла

- Вместо обучения отдельного эмбединга для каждого узла мы обучаем набор функций-агрегаторов, которые учатся агрегировать информацию об объектах из локального окружения узла.
- Каждая функция агрегатора собирает информацию с разного количества узлов в зависимости от глубины поиска от заданного узла.
- При тестировании используется обученная система для получения эмбедингов для полностью невидимых узлов.

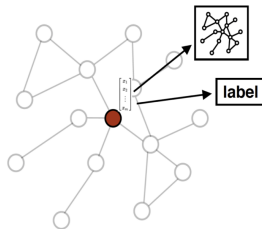
GraphSage (пример)



1. Sample neighborhood



2. Aggregate feature information from neighbors



3. Predict graph context and label using aggregated information

GraphSage: Pooling aggregator (1)

- Выполняет поэлементный pooling на множестве соседей. Пример max-pooling:

$$AGGREGATE_k^{pool} = \max (\{ \sigma (\mathbf{W}_{pool} \mathbf{h}_{u_i}^{k-1} + \mathbf{b}), \forall u_i \in \mathcal{N}(v) \})$$

- которую можно заменить функцией mean-pooling или любой другой симметричной функцией pooling. Это указывает на то, что pooling агрегатор работает лучше всего, в то время как агрегатор mean-pooling и max-pooling имеют одинаковую производительность.

GraphSage: Pooling aggregator (2)

- Функция потерь определяется как:

$$J_G(\mathbf{z}_u) = -\log(\sigma(\mathbf{z}_u^T \mathbf{z}_u)) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log(\sigma(-\mathbf{z}_u^T \mathbf{z}_u))$$

- где u и v встречаются одновременно в случайном блуждании фиксированной длины, а v_n - “негативные” примеры, которые не встречаются одновременно с u , Q и $P_n(v)$ - число “негативных” примеров и их распределение вероятностей, \mathbf{z}_u - выходные векторы

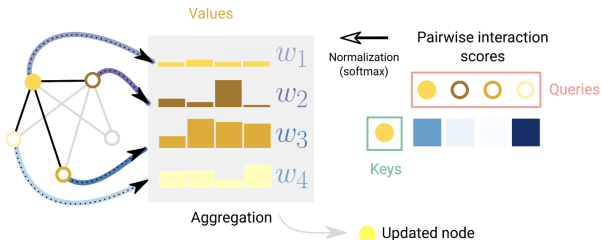
Graph Attention Networks (1)

- Другой способ передачи информации между атрибутами графа - это модель внимание. Например, когда рассматриваем суммирование (агрегацию) узла и его соседних узлов на расстоянии 1-го ребра, можно также рассмотреть возможность использования взвешенной суммы. Тогда задача состоит в том, чтобы связать веса инвариантным к перестановкам способом.
- Один из подходов - использование скалярной скоринговой функции, которая присваивает веса на основе пар узлов $f(node_i, node_j)$.
- В этом случае оценочную функцию можно интерпретировать как функцию, которая измеряет, насколько релевантен соседний узел по отношению к центральному узлу.

Graph Attention Networks (2)

- Веса можно нормализовать, например, с помощью функции softmax, чтобы сосредоточить большую часть веса на соседе, наиболее важном для узла по отношению к задаче.
- Эта концепция лежит в основе сетей Graph Attention Networks (GAT) и Set Transformers.
- Инвариантность перестановок сохраняется, потому что скоринговая функция анализирует пары узлов.

Graph Attention Networks (3)



Модель внимания для одного узла по отношению к соседним узлам. Для каждого ребра вычисляется оценка взаимодействия, нормализуется и используется для взвешивания вложений узлов.

Graph Attention Networks (4)

- Трансформеры можно рассматривать как GNN с механизмом внимания.
- Согласно этому представлению, трансформер моделирует несколько элементов (например, токены) как узлы в полностью связанном графе, а механизм внимания назначает эмбединги ребер каждой паре узлов, которые используются для вычисления весов внимания.
- Разница заключается в предполагаемом шаблоне связи между объектами, GNN предполагает разреженный шаблон, а трансформер моделирует все соединения.

