

Обработка графов

Виктор Китов
victorkitov.github.io



Содержание

1 Введение

2 Матричные критерии для эмбеддингов

3 Эмбеддинги из обхода графа

4 Эмбеддинги через автокодировщик

5 Графовые нейросети

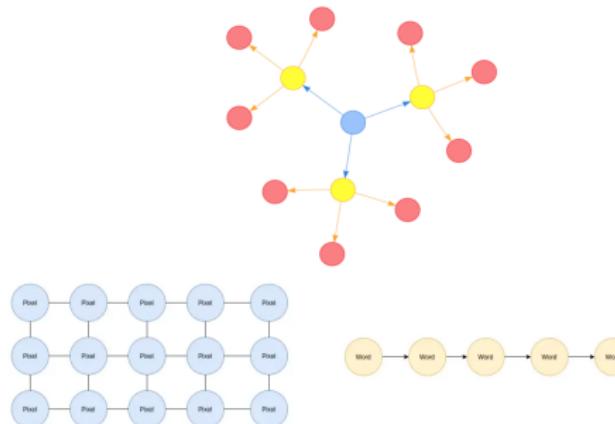
Граф и др. типы данных

Граф: (V, E) .

- рёбра направленные / ненаправленные
- могут задаваться веса связей W (обычно ≥ 0)

Нет понятий левый/правый/верхний/нижний сосед

- как в изображениях и текстах



Социальные сети



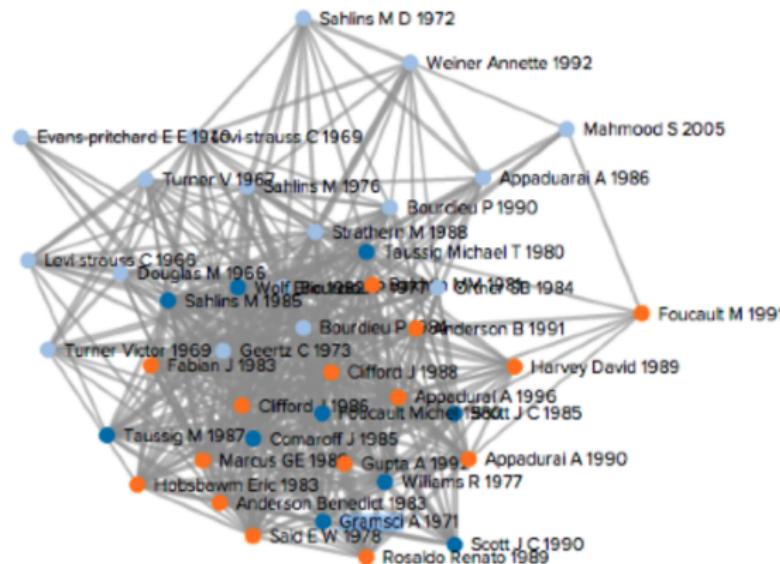
Социальная сеть:

- узлы: люди
- связи: состояние дружбы, сообщения, репосты.

Задачи на социальных сетях

- Предсказание свойств узла (node prediction):
 - не указанные атрибуты: возраст, пол, образование, профессия
 - интересы
 - является ли аккаунт ботом?
- Предсказание связей (link prediction):
 - рекомендации друзей
- Обнаружение сообществ (community prediction):
обнаружение сильно связанных подграфов, отвечающих
 - коллегам по работе, выпускникам одного ВУЗа и т.д.

Граф цитирований научных работ



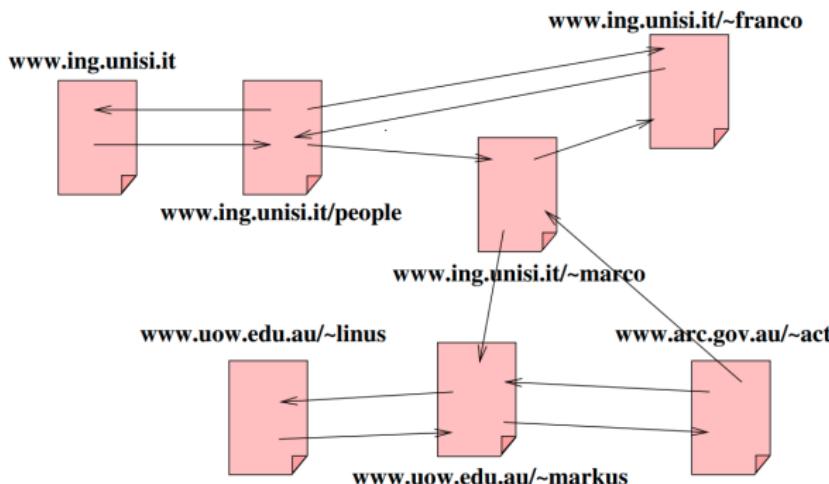
Граф цитирований:

- узлы: статьи
- связи: цитирования др. работ.

Задачи на графе цитирований

- Предсказание свойств узла:
 - тематика работы
 - ключевые слова
 - авторитетность работы
- Предсказание связей:
 - рекомендации для полного обзора литературы
- Обнаружение сообществ:
 - направления исследований, научные школы

Граф веб-страниц



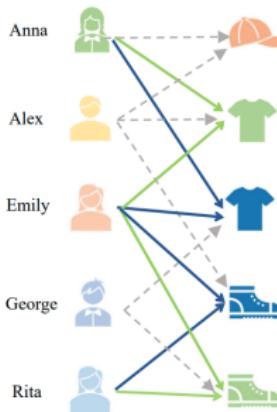
Граф веб-страниц:

- узлы: веб-страницы
- связи: гиперсылки, факт просмотра одним пользователем.

Задачи на графе веб-страниц

- Предсказание свойств узла:
 - важность страницы при ранжировании
 - тематика страницы
- Предсказание связей:
 - рекомендация страниц похожей тематики
- Обнаружение сообществ:
 - построение семантической карты интернета
 - выявление рубрик для автоматической рубрикации страниц

Взаимодействие пользователей и товаров



Взаимодействия пользователей и товаров (user-item graph)¹

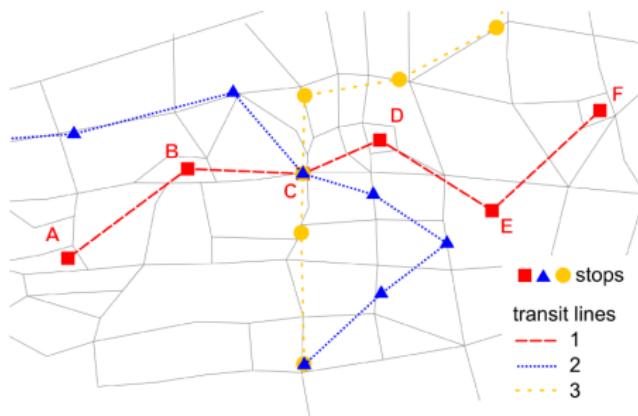
- узлы: пользователи, товары
- связи: их взаимодействие (купил товар, просмотрел видео, лайкнул описание)

¹Stacked Mixed-Order Graph Convolutional Networks for Collaborative Filtering (2020).

Граф взаимодействия пользователей и товаров

- Предсказание свойств узла:
 - категоризация пользователей и товаров
- Предсказание связей:
 - рекомендательная система

Транспортная сеть



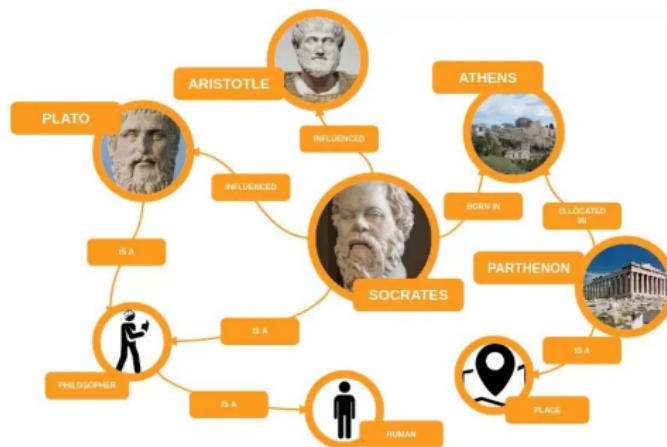
Транспортная сеть:

- узлы: локации
- связи: переезд из одной локации в другую.

Задачи на транспортной сети

- Предсказание свойств узла:
 - определение типов застройки (офисы/жилые дома/торговые центры)
- Предсказание связей:
 - рекомендация новых прямых маршрутов, дорог, авиарейсов.
- Обнаружение сообществ:
 - выявление самодостаточных регионов (например для расположения рекламы)

Граф знаний



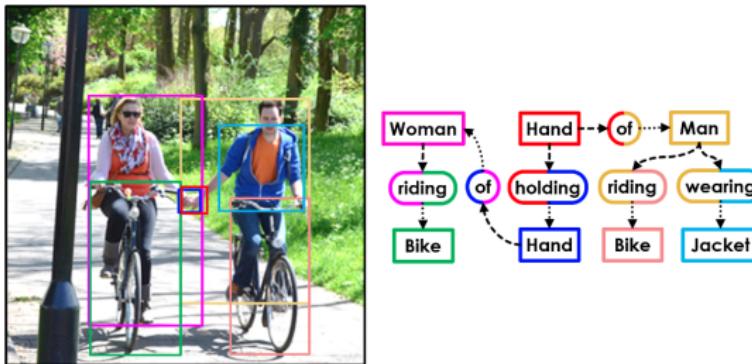
Граф знаний:

- узлы: объекты мира, сущности, понятия.
- связи: отношения между ними

Граф знаний

- Предсказание свойств узла:
 - человек: время жизни, должность, ...
 - город: уровень жизни, развитие культуры, ...
- Предсказание связей:
 - выявление взаимосвязей между сущностями, новые знания (knowledge discovery)

Граф сцены (scene graph)²



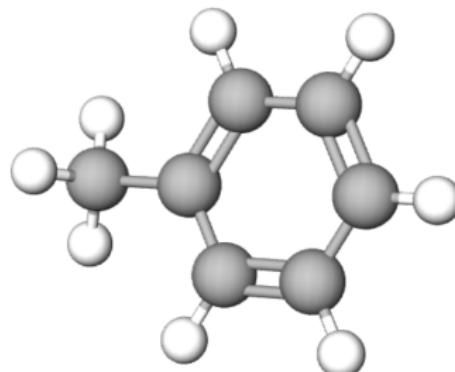
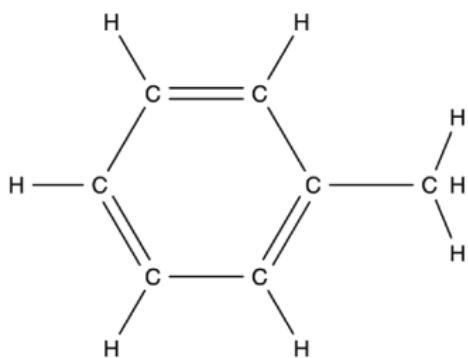
Граф сцены (scene graph):

- узлы: объекты
- связи: типы взаимосвязей объектов

Классификация сцены по графу (семейная, работа, панорама).

²<https://www.arxiv-vanity.com/papers/2001.02314/>

Молекула как граф



Молекула как граф:

- узлы: атомы
- связи: химические связи

Молекула как граф

Классификация графа:

- предсказывать химические свойства молекулы
- если лекарство: лечебный эффект

Генерация графа:

- придумывание вещества с заданными свойствами
- придумывание лекарства с желаемым эффектом

Инструменты

- Коллекция графовых датасетов - Open Graph Benchmark

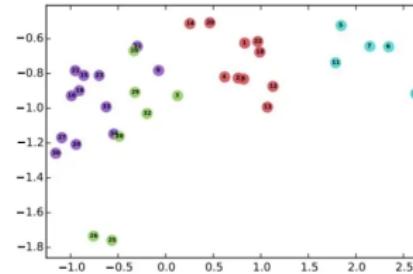
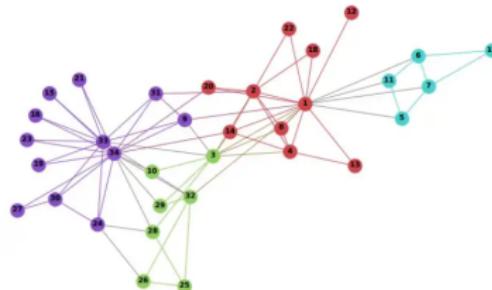


- Библиотеки:
 - PyTorch Geometric
 - Deep Graph Library
 - Spektral



Эмбеддинги для вершин³

- Задача - закодировать каждую вершину графа вектором фикс. размерности.
 - англ. node embedding (или graph representation learning)
 - размерность $\approx 200 - 300$
 - близким вершинам должны соотв. близкие представления



³A Survey on Network Embedding (2017).

Содержание

- 1 Введение
- 2 Матричные критерии для эмбеддингов
- 3 Эмбеддинги из обхода графа
- 4 Эмбеддинги через автокодировщик
- 5 Графовые нейросети

Locally linear embedding⁴

- Поиск эмбеддингов можно ставить как глобальную задачу (для всех вершин сразу).
 - и находить аналитическое решение (собств. вектора опр. матриц)
- Locally linear embedding: эмбеддинги должны быть линейной комбинацией эмбеддингов близких вершин.
- Пусть $C(i)$ - множество вершин, связанных с вершиной i .

$$\begin{cases} \sum_{i=1}^N \left(y_i - \sum_{j \in C(i)} w_{ij} y_j \right)^2 \rightarrow \min_{\{y_i\}} \\ \frac{1}{N} \sum_{i=1}^N y_i = 0, \quad \frac{1}{N} \sum_{i=1}^N y_i y_i^T = I \end{cases}$$

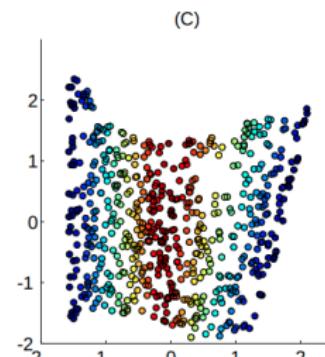
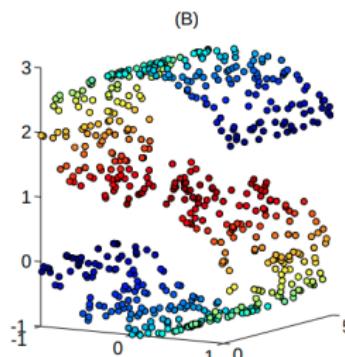
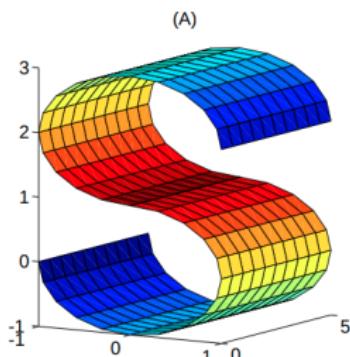
- Решение - СВ опр. матрицы.

⁴<https://cs.nyu.edu/~roweis/lle/papers/lleintro.pdf>

Locally linear embedding

Если по многомерным представлениям построить граф связей (через K-NN), то можно найти веса, \downarrow размерности: $x \rightarrow y$;

$$\begin{cases} \sum_{i=1}^N \left(x_i - \sum_{j \in C(i)} w_{ij} x_j \right)^2 \rightarrow \min \{w_{ij}\}_{ij} \\ \sum_{j=1}^N w_{ij} = 1, \quad i = 1, 2, \dots, N. \end{cases}$$



Laplacian Eigenmaps⁵

Laplacian Eigenmaps: эмбеддинги близких вершин должны быть близки:

$$\begin{cases} \sum_{i,j=1}^N w_{ij} \|y_i - y_j\|^2 \rightarrow \min_{\{y_i\}} \\ Y^T D Y = I \end{cases}$$

Граф (V, E, W) :

- $V = \{1, 2, \dots, N\}$ - вершины
- $E = \{e_{ij}\}_{i,j}$ - рёбра
- $W = \{w_{ij}\}_{ij}$ - веса рёбер, $w_{ij} \geq 0$, $w_{ij} > 0 \iff \exists e_{ij} \in E$

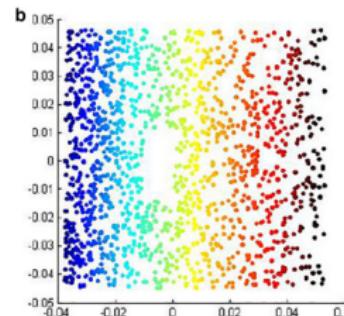
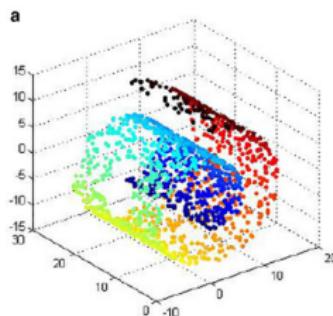
$$d_i = \sum_j w_{ij}, \quad D = \text{diag} \{d_1, d_2, \dots, d_N\}$$

Решение Laplacian Eigenmaps: СВ опр. матрицы.

⁵<https://www2.imm.dtu.dk/projects/manifold/Papers/Laplacian.pdf>

Laplacian Eigenmaps для ↓ размерности

- Laplacian Eigenmaps может использоваться для ↓ размерности.



- По x_1, \dots, x_N строим граф
 - узлы - точки, веса:
 - 0/1 в зависимости от K-NN близости или
 - $\|x_i - x_j\| \leq threshold$
 - $w_{ij} = e^{-\|x_i - x_j\|^2 / \sigma^2}$
 - ↓ размерности: $x_i \rightarrow y_i$

Cauchi Graph Embedding⁶

В Laplacian Eigenmaps $\sum_{i,j=1}^N w_{ij} \|y_i - y_j\|^2 \rightarrow \min_{\{y_i\}}$,

- $\|y_i - y_j\|^2$ недостаточно учитывает i, j с высоким w_{ij} .
 - из-за квадрата, который медленно растет в районе 0.
- i, j с низким w_{ij} влияют сильнее, чем должны из-за $\|y_i - y_j\|^2$.

Поэтому предлагается Cauchi Graph Embedding ($1 = [1, 1, \dots, 1]$):

$$\begin{cases} \sum_{i,j=1}^N \frac{w_{ij}}{\|y_i - y_j\|^2 + \sigma^2} \rightarrow \max_{\{y_i\}}, \\ Y^T Y = I, \quad Y^T 1 = 0 \end{cases}$$

- из-за деления на $\|y_i - y_j\|^2$:
 - i, j с высоким w_{ij} учитываются лучше
 - i, j с низким w_{ij} всё ещё влияют (при низком σ^2).

⁶Статья.

Cauchi Graph Embedding - лучше

Снижение размерности 2D->1D работает лучше для Cauchi Graph Embedding:

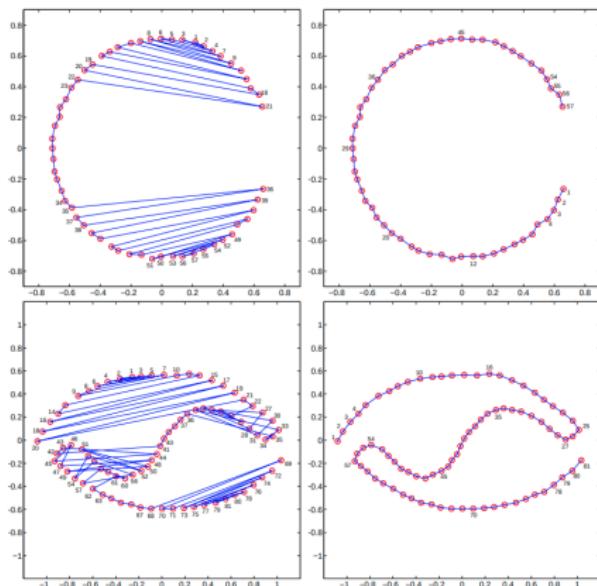
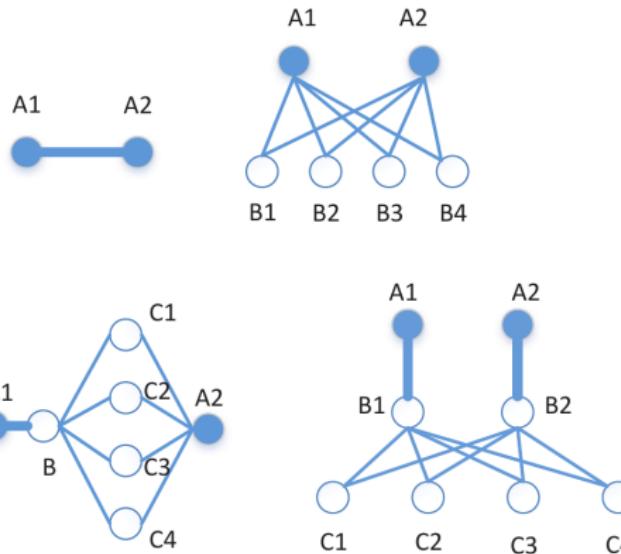


Figure. Embedding results comparison. A C-shape and S-shape manifold dataset are visualized in both figures. After performing Laplacian embedding (left) and Cauchy embedding (right), we use the numbers $1, 2, \dots$ to indicate the ordering of data points on the flattened manifold. For the visualization purpose, we use the blue lines to connect the data points which are neighboring in the embedded space.

GraRep⁷

Важность учёта $K = 1, 2, 3, 4$:

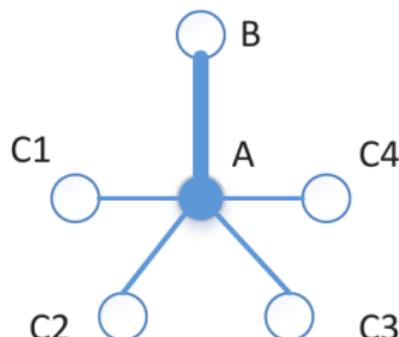
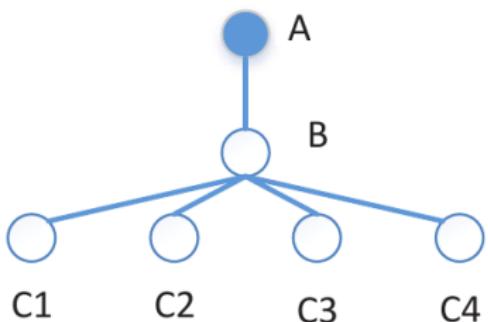


- GraRep: эмбеддинги вершин низкоранговым разложением матрицы переходов на $K = 1, 2, 3$ шагов вперёд.

- результаты для разных K конкatenируются.

GraRep

Важно не смешивать эмбеддинги для разных K , иначе будем путать, например, такие случаи:



Поэтому эмбеддинги для разных K конкатенируем.

GraRep

- Для графа (V, E) : $W = \{w_{ij}\}_{ij}$, $w_{ij} \geq 0$ - связи вершин.
 - например $w_{ij} = \mathbb{I}[e_{ij} \in E]$.
- $d_i = \sum_j w_{ij}$, $D = \text{diag}\{d_1, d_2, \dots, d_N\}$, $N = |V|$.
- Матрица вероятностей переходов за 1 шаг:

$$A = D^{-1}W, \quad A = \{p(j|i)\}_{ij}$$

- Матрица вероятностей переходов за k шагов:

$$\underbrace{A * A * \dots * A}_{k \text{ раз}} = A^k$$

- по A^k строится производная матрица матрица $B(k) \in \mathbb{R}^{N \times N}$:

$$B_{ij}(k) = \left[\log \frac{A_{ij}^k}{\sum_i A_{ij}^k} - \log \beta \right]_+$$

GraRep

- Для которой производится низкоранговое разложение
- используя сокращенный SVD ранга $d < N$:

$$B(k) = U_d \Sigma_d V_d^T = \underbrace{\left(U_d \Sigma_d^{1/2} \right)}_{Q(k)} \cdot \underbrace{\left(\Sigma_d^{1/2} V^T \right)}_{S(K)}$$

- Строки Q_k - d -мерные эмбеддинги вершин.
- Итоговые эмбеддинги для i :

$$\text{concat}_{k=1,2\dots K} \{ Q_{i:}(k) \}$$

- В целом, вычисляем матрицу похожести вершин $\{S_{ij}\}_{ij} \in \mathbb{R}^{N \times N}$ и вычисляем эмбеддинги как строки первой матрицы U низкорангового разложения⁸:

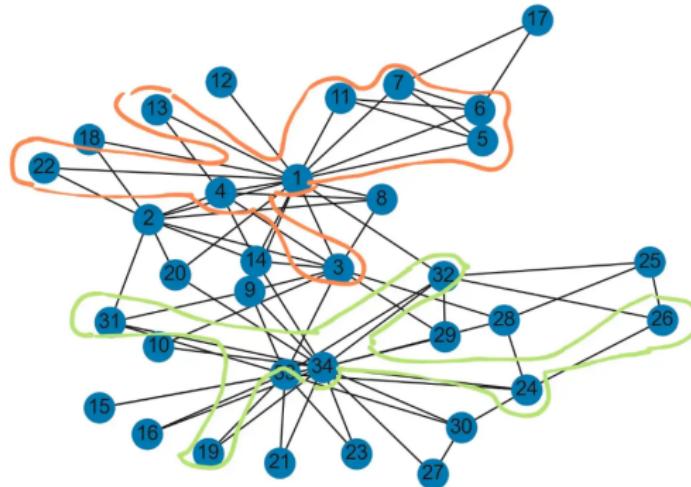
$$\|S - U \tilde{U}^T\|^2 \rightarrow \min_{U, \tilde{U}: \text{rg } U \leq d, \text{rg } \tilde{U} \leq d}$$

⁸High-order Proximity Preserved Embedding For Dynamic Networks (2018).

Содержание

- 1 Введение
- 2 Матричные критерии для эмбеддингов
- 3 Эмбеддинги из обхода графа
- 4 Эмбеддинги через автокодировщик
- 5 Графовые нейросети

DeepWalk⁹



- Для вершин генерируем пути обхода небольшой длины.
- Скользим окном фикс. ширины по сгенерированному пути.
- По центр. v_t предсказываем соседей окна (SkipGram).

⁹<https://arxiv.org/pdf/1403.6652.pdf>

DeepWalk

$$\frac{1}{T} \sum_{t=1}^T \sum_{-c \leq i \leq c, i \neq 0} \ln p(v_{t+i} | v_t) \rightarrow \max_{\theta}$$

$$p(v_{t+i} | v_t) = \frac{\exp \left(\tilde{\phi}_{v_t}^T \phi_{v_{t+i}} \right)}{\sum_{v=1}^N \exp \left(\tilde{\phi}_{v_t}^T \phi_v \right)}$$

- Знаменатель вычисляется за $O(N)$.
 - используем Hierarchical SoftMax или Negative Sampling.
- Эмбеддинги: $v \rightarrow \phi_v$ либо $\tilde{\phi}_v$, либо $[\phi_v; \tilde{\phi}_v]$.

Node2vec¹⁰

- Node2vec - развитие DeepWalk.
- В разных задачах от эмбеддинга нужны разные признаки:
 - глобальные (характеристика сообщества, которому узел принадлежит)
 - блужнание - поиском в глубину
 - локальные (структурная роль вершины [переход между сообществами, хаб в центре, точка на краю])
 - блуждание - поиск в ширину

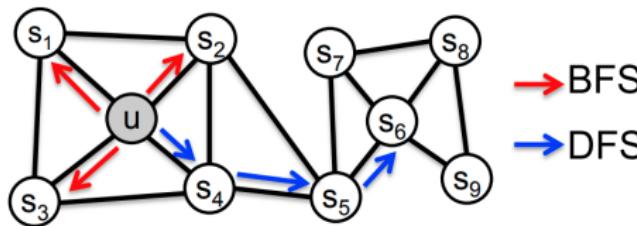
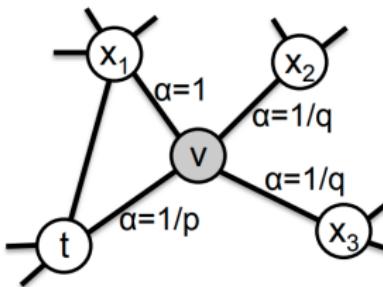


Figure 1: BFS and DFS search strategies from node u ($k = 3$).

¹⁰<https://arxiv.org/pdf/1607.00653.pdf>

Node2vec

- Node2vec регулирует глубину обхода параметрами $p, q > 0$.
- Для этого $p(j|i) \propto \alpha w_{ij}$ где α определяет что важнее:
 - обход в ширину (малое p)
 - или обход в глубину (малое q)
- Иллюстрация генерации α для узла v (только что перешли из t)
 - d_{tx} - мин. путь от t до x



Node2vec

На примере совстречаемости персонажей *Les Miserables* запуск node2vec (с последующей кластеризацией эмбеддингов цветом)

- $p = 1, q = 0.5$: обнаружение сообществ
- $p = 1, q = 2$: определение структурных ролей персонажей [эпизодические, связующие, остальные]

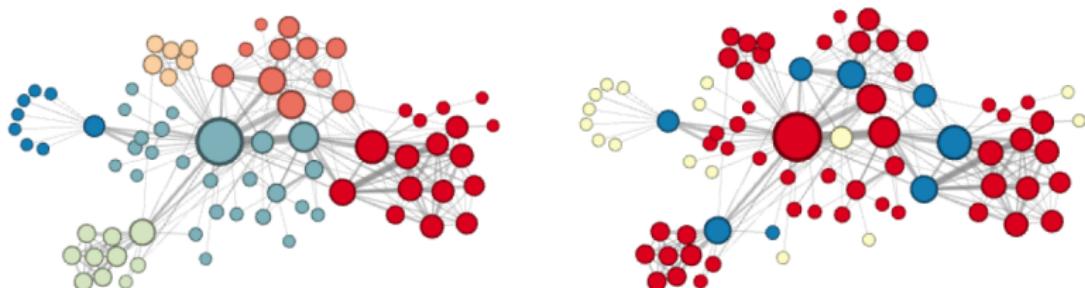


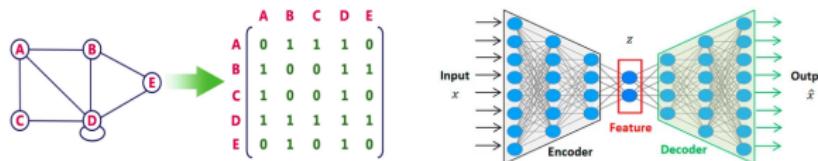
Figure 3: Complementary visualizations of *Les Misérables* co-appearance network generated by *node2vec* with label colors reflecting homophily (top) and structural equivalence (bottom).

Содержание

- 1 Введение
- 2 Матричные критерии для эмбеддингов
- 3 Эмбеддинги из обхода графа
- 4 Эмбеддинги через автокодировщик
- 5 Графовые нейросети

SDNE¹¹

- Предыдущие подходы простые.
- Structural Deep Network Embedding (SDNE): ↑ сложность генерации эмбеддингов за счёт многослойного автокодировщика.



¹¹<https://www.kdd.org/kdd2016/papers/files/rfp0191-wangAemb.pdf>

SDNE

- Автокодировщик восстанавливает строки матриц смежности для вершин i, j :
- близость 2го порядка: вершины близки, если имеют общих соседей

$$\mathcal{L}_1 = \sum_{i=1}^N \|(\hat{x}_i - x_i) \odot b_i\|_2^2$$

- важно сохранить имеющиеся связи (но если её нет, то это не значит что она не подошла бы):

$$b_{ij} = \begin{cases} 1, & w_{ij} = 0 \\ \beta > 1, & w_{ij} > 0 \end{cases}$$

SDNE

- Дополнительно сближаются близкие вершины (как в Laplacian Eigenmaps):

$$\mathcal{L}_2 = \sum_{i,j} w_{ij} \|\hat{\mathbf{y}}_i - \mathbf{y}_j\|_2^2$$

- Итоговая задача (с регуляризацией):

$$\mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_{reg} \rightarrow \min_{\theta}$$

Structural Deep Network Embedding

- Метод ↑ качество предсказания рёбер:
 - исключили часть рёбер, обучили по оставшимся, смотрим качество восстановления удалённых

Table 5: *precision@k* on ARXIV GR-QC for link prediction

Algorithm	P@2	P@10	P@100	P@200	P@300	P@500	P@800	P@1000	P@10000
<i>SDNE</i>	1	1	1	1	1*	0.99**	0.97**	0.91**	0.257**
<i>LINE</i>	1	1	1	1	0.99	0.936	0.74	0.79	0.2196
<i>DeepWalk</i>	1	0.8	0.6	0.555	0.443	0.346	0.2988	0.293	0.1591
<i>GraRep</i>	1	0.2	0.04	0.035	0.033	0.038	0.035	0.035	0.019
<i>Common Neighbor</i>	1	1	1	0.96	0.9667	0.98	0.8775	0.798	0.192
<i>LE</i>	1	1	0.93	0.855	0.827	0.66	0.468	0.391	0.05

Significantly outperforms Line at the: ** 0.01 and * 0.05 level, paired t-test.

- Параметры модели - веса, а не представления вершин.
 - для новой вершины сразу можем посчитать представление

DNGR¹²

- Deep Neural Networks for Learning Graph Representations (DNGR)



¹²Deep Neural Networks for Learning Graph Representations.

DNGR

- ➊ строится матрица совстречаемости вершин со строками
- ➋ p_0 - one-hot закодированная i -ая вершина, A -матрица переходов

$$p_i = \sum_{k=1}^K \alpha_k p_0 A^k, \quad \alpha_k \downarrow \text{посл-ть параметров}$$

- ➌ Строится матрица positive pointwise mutual information
 - \downarrow эффект часто встречающихся вершин ("стоп-слов")

$$PPMI_{ij} = \left[\log \left(\frac{p_{ij}}{p_i p_j} \right) \right]_+$$

- ➍ Автокодировщик учится восстанавливать строки $PPMI_{i:}$
 - эмбеддинг - внутр. представление автокодировщика
 - можем считать эмбеддинги новых вершин

DNGR

- Автокодировщик настривался слой за слоем
 - старые слои кодировщика фиксировались при обучении последующих
 - англ. greedy layer-wise pretraining
- Использовался denoising-автокодировщик
 - по зашумлённому входу - восстанавливаем чистый вход
 - зашумление: зануление небольшого #входов
 - обозн. красным

Содержание

- 1 Введение
- 2 Матричные критерии для эмбеддингов
- 3 Эмбеддинги из обхода графа
- 4 Эмбеддинги через автокодировщик
- 5 Графовые нейросети

Графовые нейросети

- Графовые нейросети (graph neural networks, GNN) генерируют эмбеддинги вершин.
 - сочемшают кодирование геометрии и признаков вершин x_u
 - если не признаков нет, то можно задать $[1, 1, \dots, 1]$, посчитать признаки по геометрии либо one-hot-enc вершин.

$$h_u^{(0)} = x_u \quad \# \text{ инициализация}$$

для $k = 1, 2, \dots, K$: # message passing K раз

$$h_u^{(k+1)} = \text{UPDATE} \left(h_u^{(k)}, \text{AGGREGATE}^{(k)} \left(\left\{ h_v^{(k)} \right\}_{v \in \mathcal{N}(u)} \right) \right)$$

$$z_u = h_u^{(K)} \quad \# \text{ выходные эмбеддинги вершин}$$

пример:
$$h_u^{(k)} = \sigma \left(W_{self}^{(k)} h_u^{(k-1)} + W_{neigh}^{(k)} \sum_{v \in \mathcal{N}(u)} h_v^{(k-1)} + b^{(k)} \right)$$

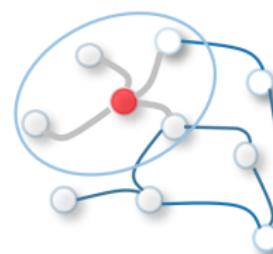
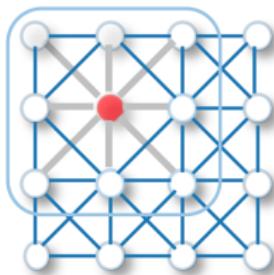
- Настройка весов - под итоговую задачу с учителем.

Графовые нейросети¹⁴

- Граф. свёрт. сети (Graph Convolutional Networks, GCN¹³).
- Операция свёртки на графах:

$$h_u^{(k)} = \sigma \left(W^{(k)} \sum_{v \in \mathcal{N}(u) \cup \{u\}} \frac{h_v}{\sqrt{|\mathcal{N}(u)| |\mathcal{N}(v)|}} \right)$$

- Не различает соседей $\uparrow, \downarrow, \leftarrow, \rightarrow$



¹³Kipf, Welling (2016).

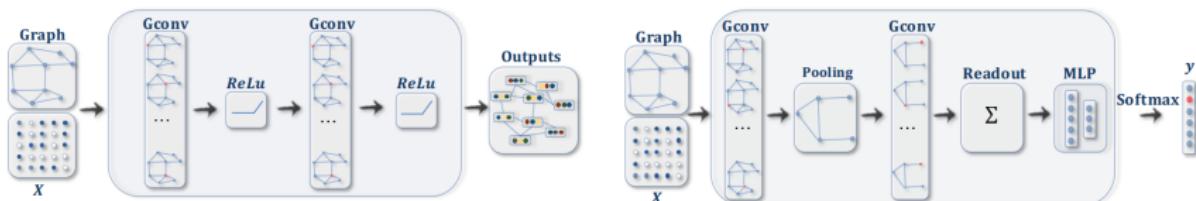
¹⁴Обзор графовых нейросетей.

Графовые нейросети

- Свёртки могут считать эмбеддинги вершин.
- Прогноз для графа:
 - после свёрток можно произвести глобальный пулинг и прогноз

$$\hat{y} = \text{MLP} \left(\frac{1}{N} \sum_u h_u \right)$$

- либо серию частичных пулингов (graph coarsening) за счёт кластеризации и прогноз.



Графовые нейросети с вниманием

- Graph Attention Network, GAT¹⁵: ↑ гибкость моделей
 - веса соседей при агрегации зависят от состояний соседей
 - $[a;b] = \text{concat}(a, b)$

$$h_u^{(k)} = \sigma \left(W^{(k)} \sum_{v \in \mathcal{N}(u) \cup \{u\}} \alpha_{uv}^{(k)} h_v \right)$$

$$\alpha_{uv} = \frac{\exp(a^T [Wh_u; Wh_v])}{\sum_{\tilde{v} \in \mathcal{N}(u)} \exp(a^T [Wh_u; Wh_{\tilde{v}}])}$$

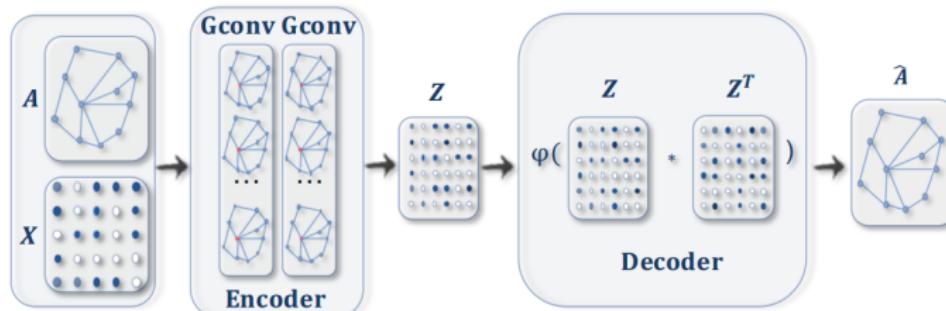
- Другие варианты внимания:

$$\alpha_{uv} = \frac{\exp(h_u^T Wh_v)}{\sum_{\tilde{v} \in \mathcal{N}(u)} \exp(h_u^T Wh_{\tilde{v}})}; \quad \alpha_{uv} = \frac{\exp(\text{MLP}(h_u, h_v))}{\sum_{\tilde{v} \in \mathcal{N}(u)} \exp(\text{MLP}(h_u, h_{\tilde{v}}))}$$

¹⁵<https://arxiv.org/pdf/1710.10903.pdf>

Восстановление рёбер

Восстанавливать (рекомендовать) рёбра графа можно с помощью автокодировщика¹⁶:



A GAE for network embedding. The encoder uses graph convolutional layers to get a network embedding for each node. The decoder computes the pair-wise distance given network embeddings. After applying a non-linear activation function, the decoder reconstructs the graph adjacency matrix. The network is trained by minimizing the discrepancy between the real adjacency matrix and the reconstructed adjacency matrix.

¹⁶<https://arxiv.org/abs/1611.07308>

Генерация графов

Генерация новых графов, используя GAN¹⁷:

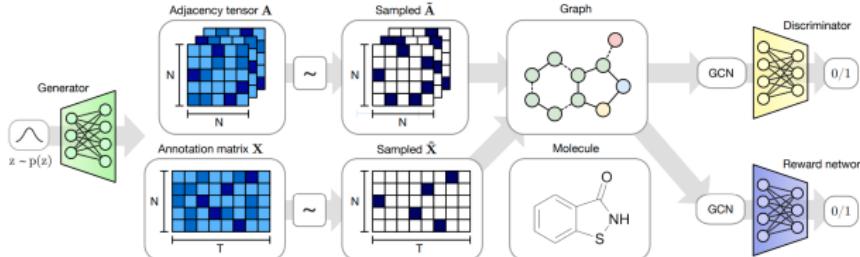
- генератор учится генерировать матрицы смежности A и характеристики связей X , неразличимые от настоящих:

$$P_A, P_X = \sigma(\text{MLP}(z))$$

$$A \sim \text{Bernoulli}(P_A)$$

$$X \sim \text{Bernoulli}(P_X)$$

- дискриминатор - графовая нейросеть



¹⁷ <https://arxiv.org/abs/1805.11973>

Дополнительная информация

- Простая обзорная статья.
- Обзор графовых нейросетей.
- Другой обзор графовых нейросетей.
- Книга Graph Representation Learning (2020).