



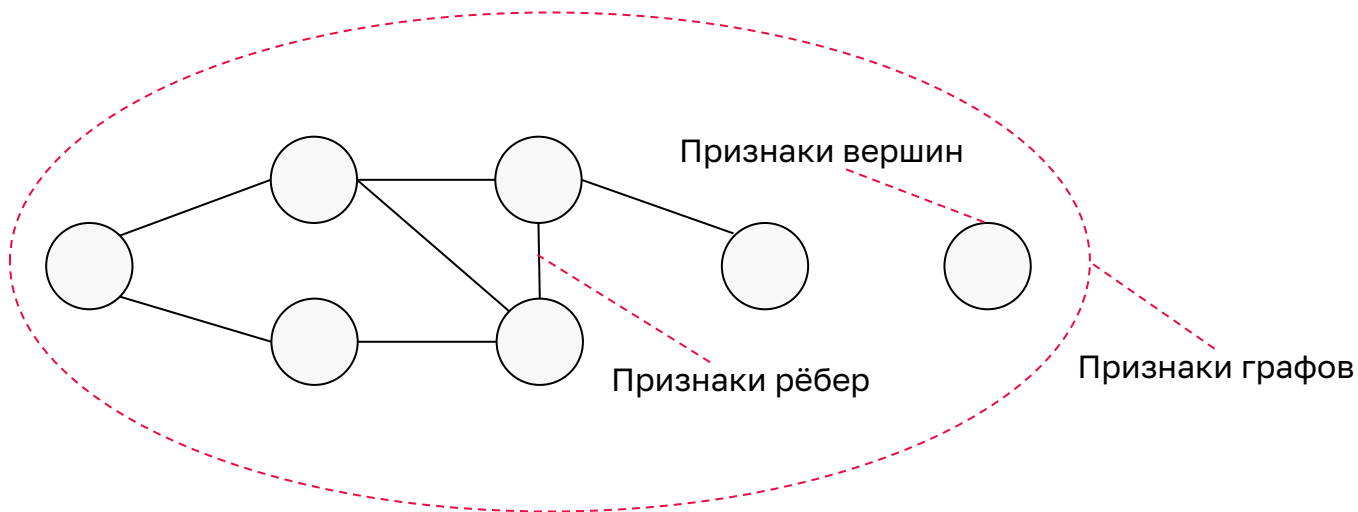
ІІТМО

Анализ графовых данных и глубокое обучение

Азимов Рустам
Высшая школа цифровой культуры

В предыдущих сериях

- Разработать правила получения признаков для вершин/рёбер/графов
- Выбор признаков для описания вершин/рёбер/графов является ключевым для качественного решения задачи



Hand-crafted features

- Признаки для вершин
 - Степень вершин
 - Показатели центральности
 - Коэффициент кластеризации
 - Графлеты
- Признаки для рёбер
 - Distance-based
 - Local neighborhood overlap
 - Global neighborhood overlap
- Признаки для графов
 - Graphlet Kernel
 - Weisfeiler-Lehman Kernel





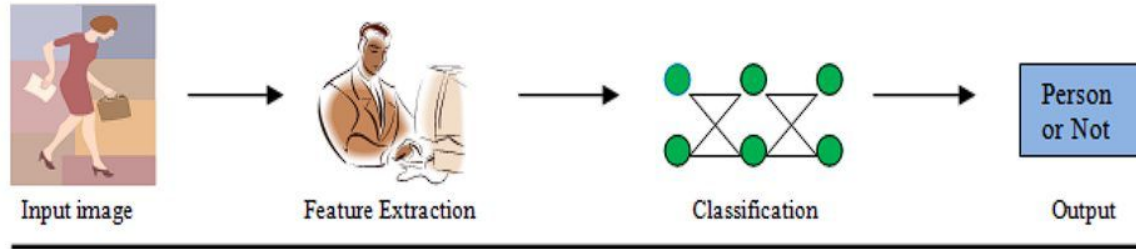
IITMO

Node Embeddings

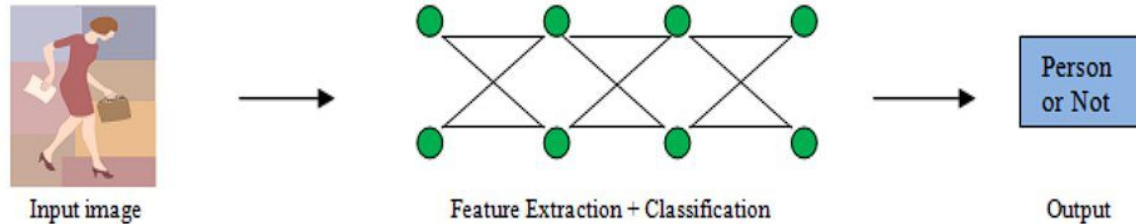
Feature Engineering vs Representation Learning



Machine Learning

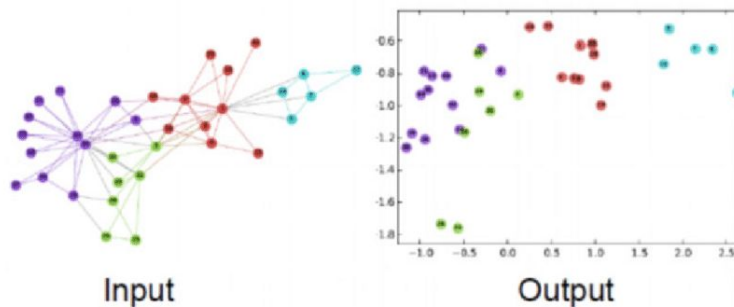
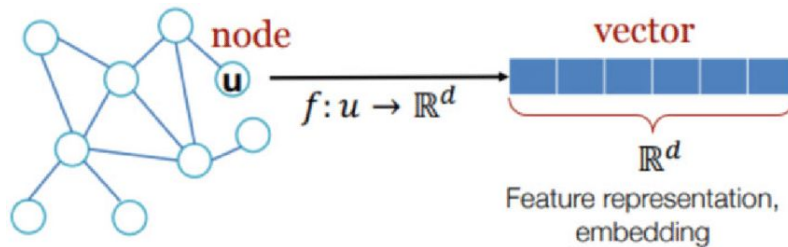


Deep Learning

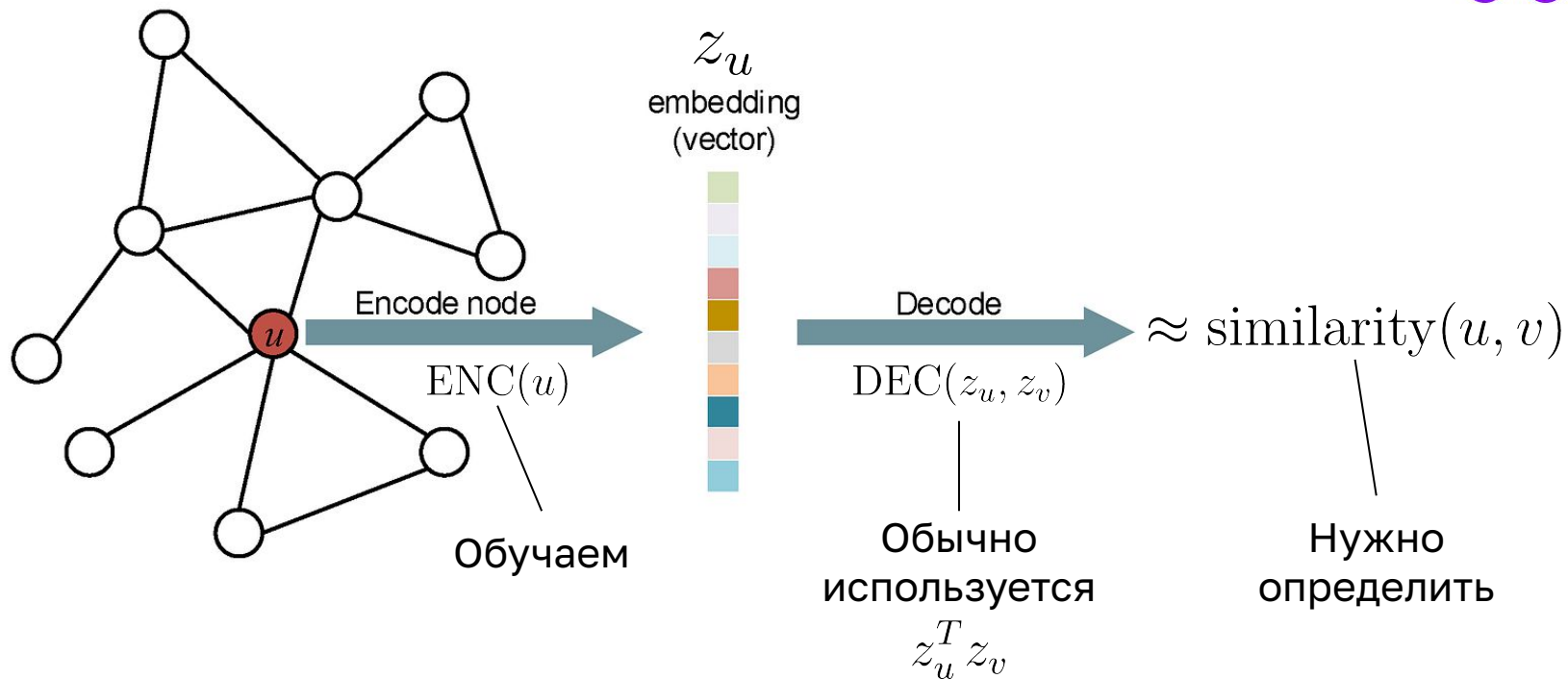


Graph Representation Learning

- Хочется научиться получать признаки автоматически и чтобы работало для разных задач
- Эмбединги должны описывать граф и быть не большой размерности
- Схожие вершины в графе должны иметь близкие эмбединги

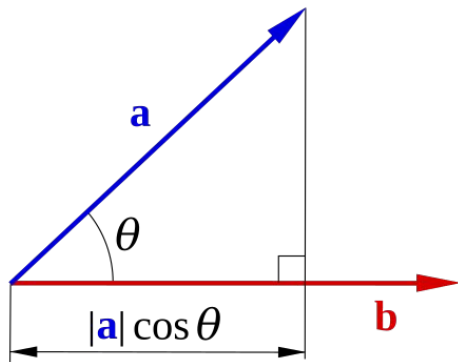


Encoder + Decoder



- Обучение: оптимизация параметров $\text{ENC}(u)$
- Максимизируем $z_u^T z_v$ для схожих вершин u и v
- У близких векторов больше скалярное произведение

$$z_u^T z_v = |z_u| |z_v| \cos \Theta$$



Матрица эмбедингов

- Как и в случае с некоторыми ядовыми методами параметрами являются сами эмбединги для каждой вершины



$$Z = \begin{bmatrix} Z_{11} & Z_{12} & \cdots & Z_{1n} \\ Z_{21} & Z_{22} & \cdots & Z_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ Z_{n1} & Z_{n2} & \cdots & Z_{nn} \end{bmatrix}$$

- Как и при выборе hand-crafted признаков ключевым является выбор: какие вершины считаем схожими?
 - Соседи?
 - Имеющие общих соседей?
 - Имеющие схожее локальное окружение?
- Рассмотрим новую идею, основанную на случайных блужданиях





IITMO

Node Embeddings by Random Walks

- Начинаем с некоторой вершины и случайно перемещаемся по рёбрам графа
- Решаем, что схожесть двух вершин в графе $\text{similarity}(u, v)$ – вероятность того, что вершины u и v обе попадутся в таком случайном блуждании
- Фиксируем некоторую стратегию блуждания R и для каждой пары вершин подсчитываем оценки вероятности посетить v , если начали с вершины u – $P_R(v|u)$



- Для процесса обучения — подбор таких эмбедингов вершин, чтобы их скалярное произведение отражало описанное свойство схожести

$$z_u^T z_v \approx \text{similarity}(u, v)$$

- Такой подход учитывает как локальное окружение вершин, так и более длинные пути

- Из каждой вершины u графа запускаем серию случайных блужданий фиксированной длины
- Подсчитываем $N_R(u)$ — мультимножество посещённых вершин
- Из метода максимального правдоподобия получаем оптимизационную задачу

$$\arg \max_z \sum_{u \in V} \log P(N_R(u) | z_u)$$

- Переходим к negative log-likelihood и параметризуем вероятность с использованием softmax

$$\arg \min_z \mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} -\log\left(\frac{e^{z_u^T z_v}}{\sum_{t \in V} e^{z_u^T z_t}}\right)$$

$O(|V|^2)$ softmax

- Оптимизируем эмбединги z с помощью, например, SGD (Stochastic Gradient Descent)

DeepWalk: Negative sampling

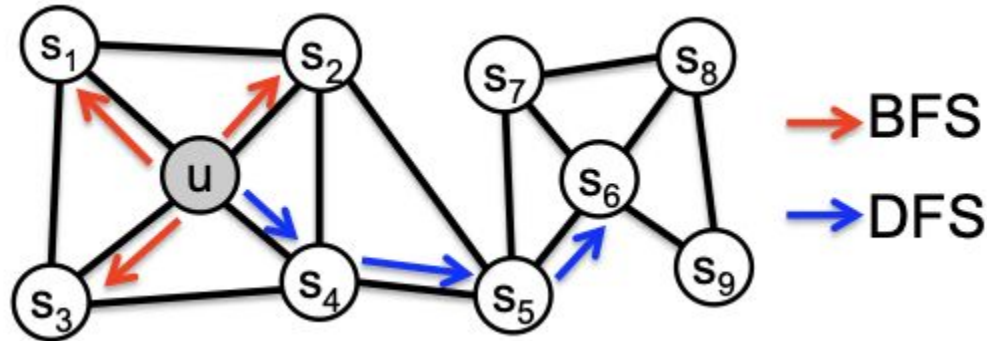
- Для снижения сложности вычислений нормируем не по всем вершинам, а только по небольшому числу вершин t_i не из случайных блужданий (negative nodes or negative samples)

$$-\log\left(\frac{e^{z_u^T z_v}}{\sum_{t \in V} e^{z_u^T z_t}}\right) \approx \log(\sigma(z_u^T z_v)) + \sum_{i=1}^k \log(\sigma(-z_u^T z_{t_i}))$$

- Обычно берут k от 5 до 20 (чем выше k , тем точнее, но дольше)
- t_i выбираем случайно, но с вероятностью пропорциональной степеням вершин
- Для эффективности можно брать любые вершины, а не только negative nodes



- Хотим обобщить идею случайных блужданий для более гибкого определения соседей $N_R(u)$ где настраиваются доли локальной и глобальной информации о графе

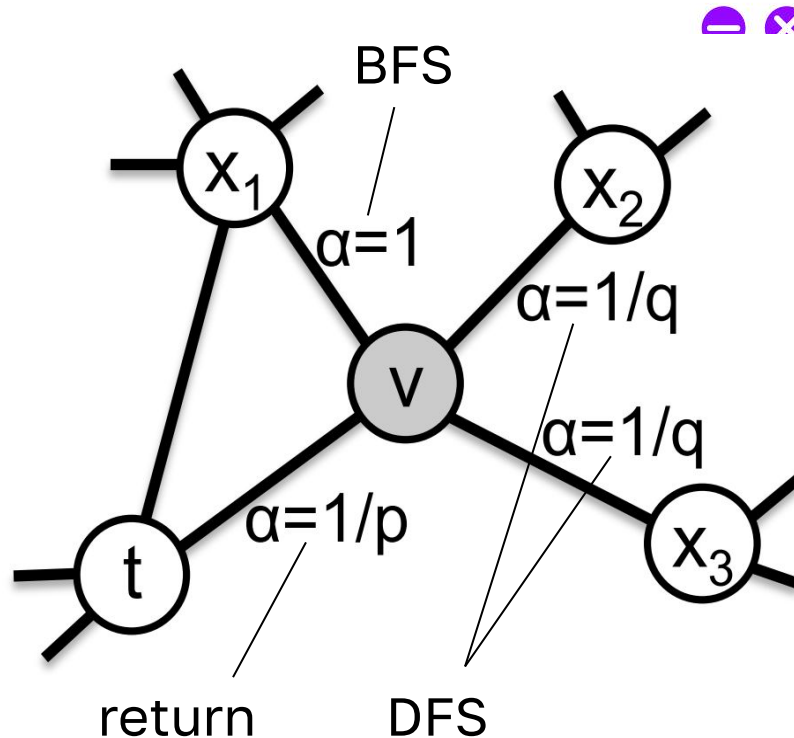


- Для вычисления $N_R(u)$ настраиваем два параметра смещенного случайного блуждания 2го порядка
- Параметр p — настраивает частоту возвращения к предыдущей вершине
- Параметр q — настраивает соотношение BFS vs DFS стратегий обходов



node2vec: Смещенное блуждание

- Пусть прошлый шаг был из t в v
- Для следующего шага вычисляются числа пропорциональные вероятностям перехода
- Зависят от параметров p и q
- Затем они нормируются и получаем вероятности



- Заранее подсчитаем такие вероятности для всех (t, v)
- Для вычисления $N_R(u)$ симулируем r случайных блужданий длины l , начиная из каждой вершины u
- Оптимизируем тот же функционал для нахождения эмбедингов
- $p < 1$ — локальная информация о графе, $p > 1$ — глобальная
- Маленький q — чаще BFS, большой q — чаще DFS
- Линейная сложность, легко использовать параллельные вычисления



Node Embeddings

- Есть и другие алгоритмы, для конкретной задачи нужно подобрать наиболее подходящий
- Например, node2vec, хорошо себя показывает на классификации вершин
- Алгоритмы, основанные на случайных блужданиях, довольно эффективные





IITMO

Graph Embeddings

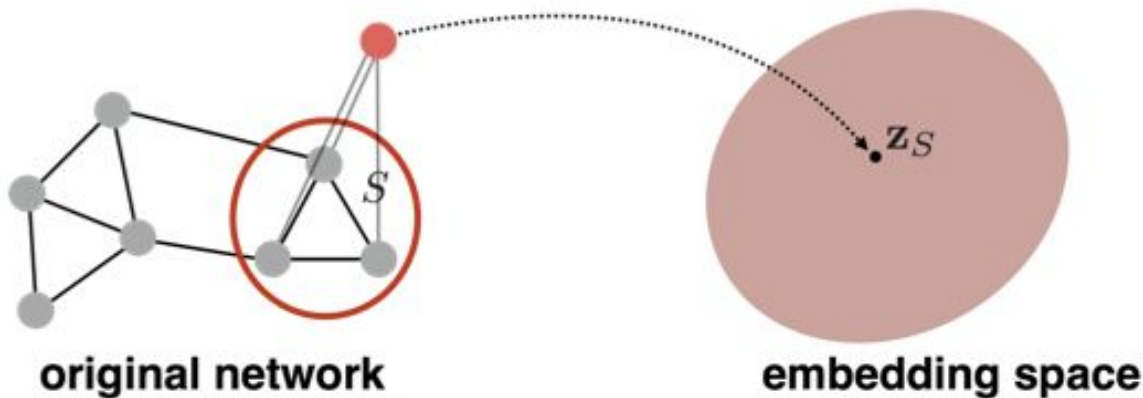
Эмбединги для графов

- Хотим закодировать информацию о всем графе
- Например, решать задачу классификации графов
 - Токсичность или нетоксичность веществ
 - Наличие нужного свойства лекарства
- Простое решение – суммировать или агрегировать эмбединги вершин

$$z_G = \sum_{v \in G} z_v$$

Эмбединги для графов

- Другая идея – добавить новую вершину и связать ее со всем графом или подграфом, для которого хотим получить эмбединг
- Затем использовать просто эмбединг этой новой вершины



- Рассмотренные сегодня методы для получения эмбедингов вершин/графов не позволяют работать с новыми вершинами/графами (не из тренировочного набора)
 - Добавление нового пользователя соцсети
 - Придется пересчитывать все эмбединги
- Если вершины слишком далеко, то даже идентичная локальная структура не поможет найти сходство с помощью случайных блужданий
- Не используют признаки вершин, рёбер, графов



Заключение

