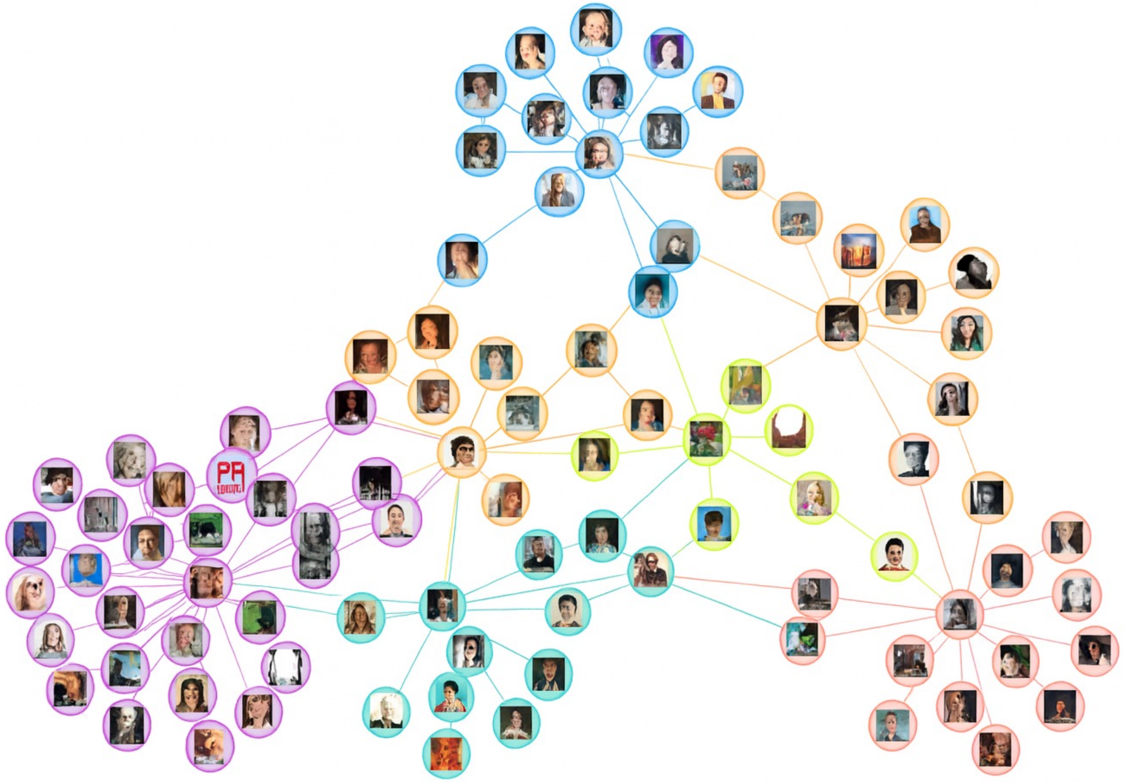
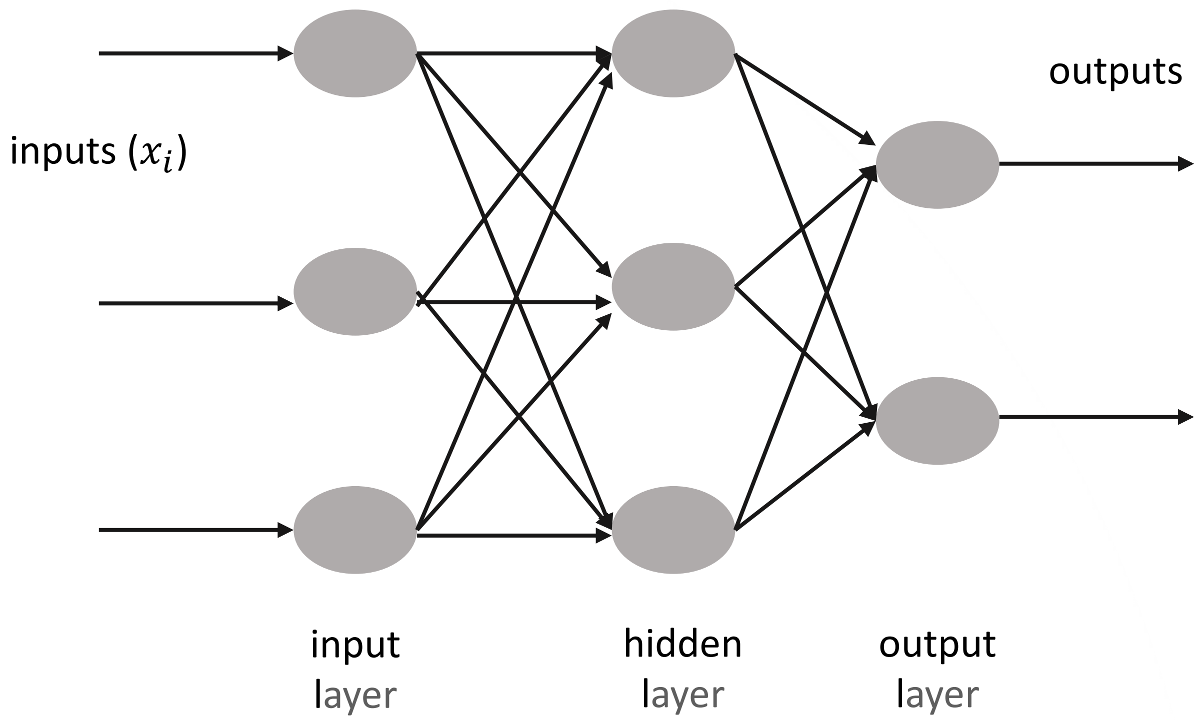
Графовые нейросети и их применение

Примеры применения графов



MLP – многослойный перцептрон

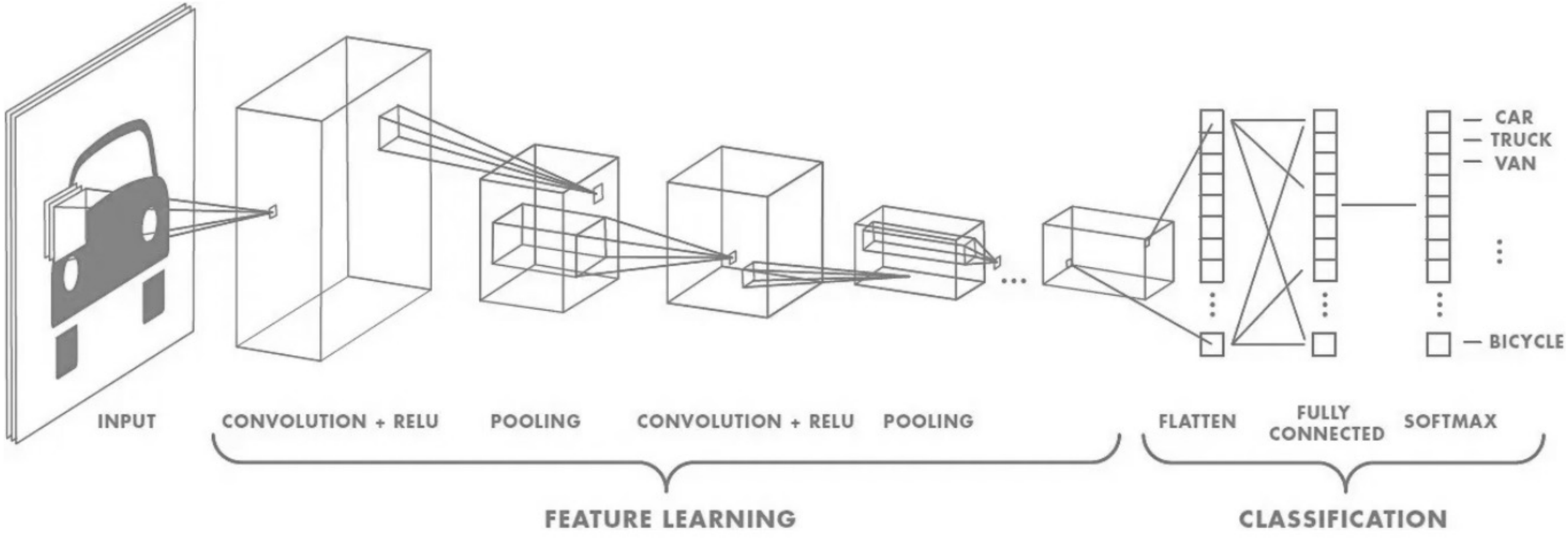


$$y = f\left(\sum_i w_i x_i + b\right)$$

где:

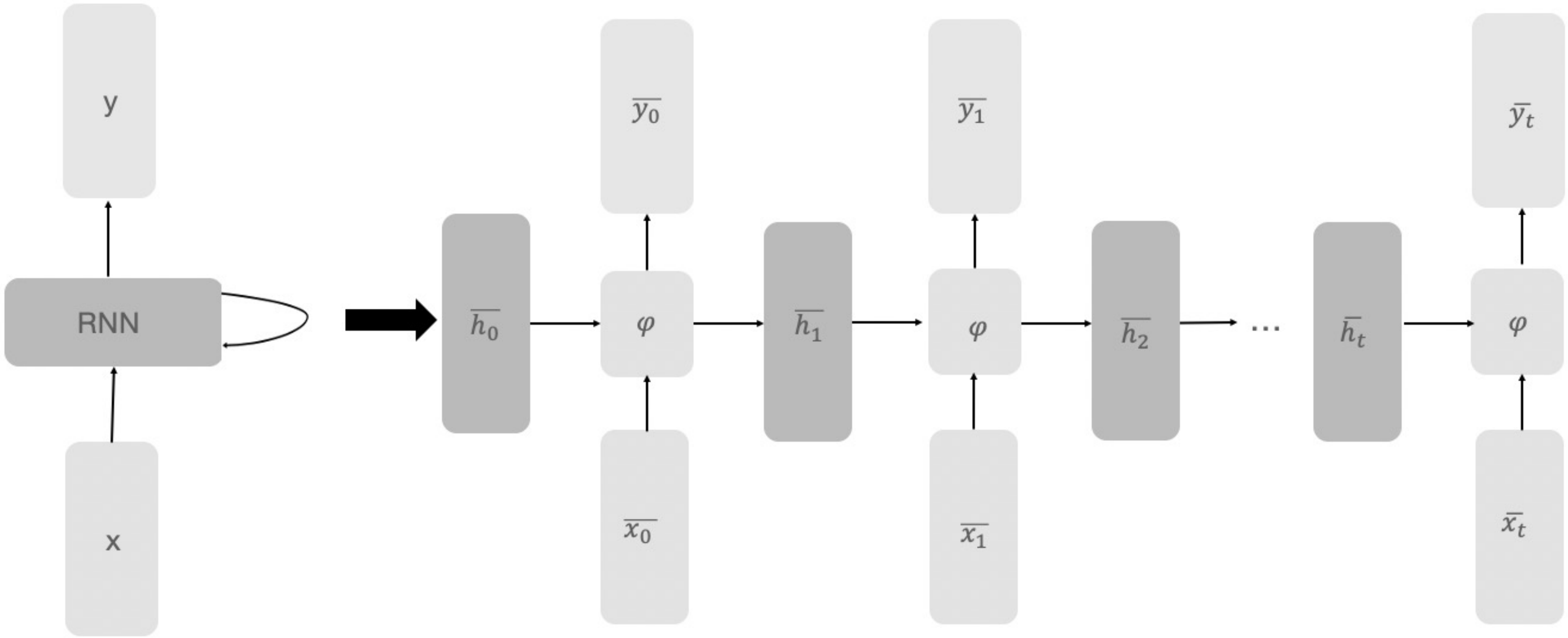
- x_i – входные признаки,
- w_i – веса,
- b – смещение (bias),
- $f(\cdot)$ – функция активации.

CNN – сверточные нейронные сети



Источник: https://medium.com/data-science/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

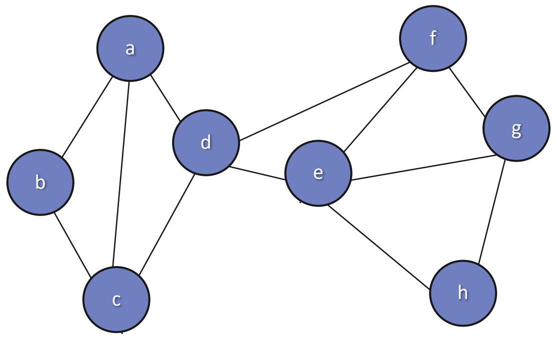
RNN – рекурентные нейронные сети



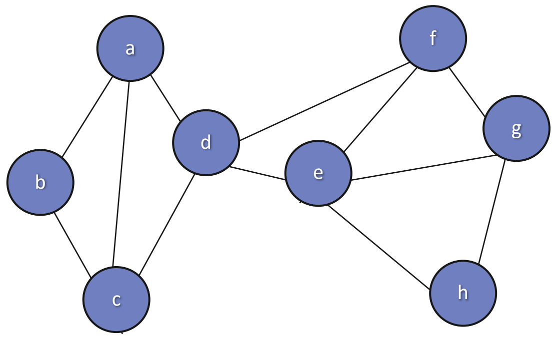
Особенность	Традиционные нейросети	Графовые нейросети (GNN)
Учет взаимодействий между объектами	Ограничен или отсутствует	Полноценное моделирование на графах
Работа с неевклидовыми структурами	Не поддерживается	Эффективно поддерживается
Учет топологии данных	Не предусмотрен	Вшит в саму архитектуру (message passing)
Масштабируемость (большие сети)	Сложно	Есть методы выборки соседей и обучения на подграфах
Динамические изменения (изменение структуры графа)	Практически не учитываются	Поддерживаются (например, Temporal GNN)

Graph Neural Networks (GNN)

Graph neural network (GNN) – это класс нейронных сетей для обработки данных, которые можно представить в виде графов



Graph Neural Networks (GNN)



Random walk →

Когда мы запускаем **случайное блуждание** по графу,
мы фактически задаём **Марковский процесс** — последовательность случайных перемещений между вершинами.

$$X_0, X_1, X_2, \dots, \quad X_t \in V$$

Условие Маркова:

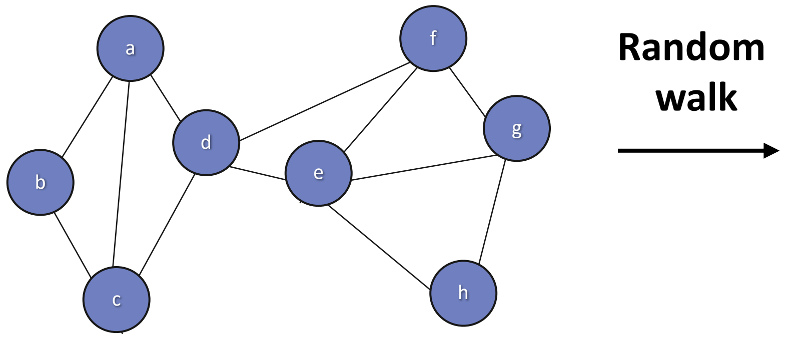
$$\Pr(X_{t+1} = j \mid X_t = i, X_{t-1}, \dots, X_0) = P_{ij}$$

Матрица P называется **матрицей переходных вероятностей**:

$$P_{ij} = \begin{cases} \dfrac{1}{\deg(i)}, & \text{если } (i, j) \in E \\ 0, & \text{иначе.} \end{cases}$$

То есть всё поведение блуждания полностью описано P.

Graph Neural Networks (GNN)



Неориентированный невзвешенный граф

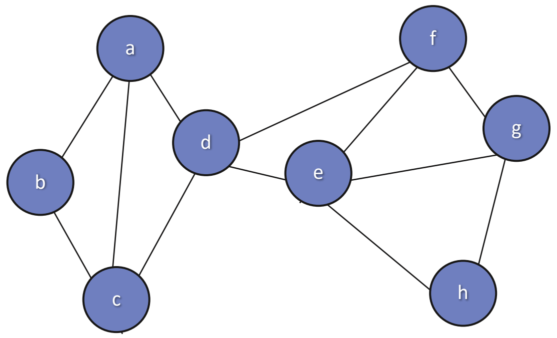
- $\deg(v)$ — степень вершины v.
- Переходная матрица $P \in \mathbb{R}^{n \times n}$:

$$P_{ij} = \begin{cases} \dfrac{1}{\deg(i)}, & (i,j) \in E, \\ 0, & \text{иначе.} \end{cases}$$

- Марковская динамика: $\Pr[X_{t+1} = j \mid X_t = i] = P_{ij}$.
- Вектор распределения $\pi^{(t)} \in \Delta^{n-1}$ эволюционирует так:

$$\pi^{(t+1)} = \pi^{(t)} P, \quad \pi^{(0)} \text{ задано.}$$

Graph Neural Networks (GNN)



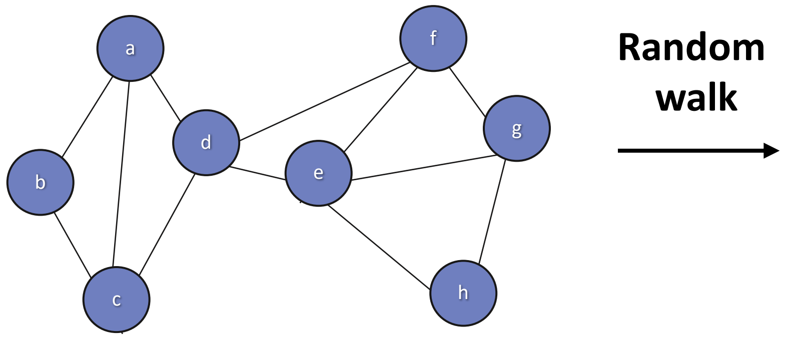
Random walk →

Взвешенный граф

Если ребро (i, j) имеет вес $w_{ij} > 0$, то

$$P_{ij} = \frac{w_{ij}}{\sum\limits_{k:(i,k)\in E} w_{ik}}.$$

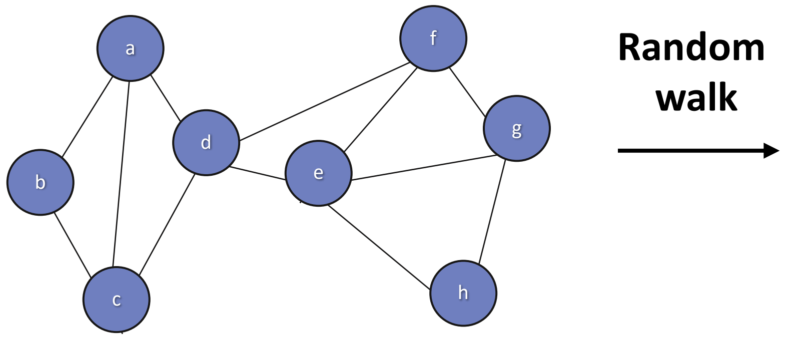
Graph Neural Networks (GNN)



Ориентированный граф

Переходы определяются по **исходящей** степени/сумме весов: $P_{ij} > 0$ только если есть дуга $i \to j$, а строки P нормированы к 1.

Graph Neural Networks (GNN)



Представим, что мы очень долго бродим по графу.

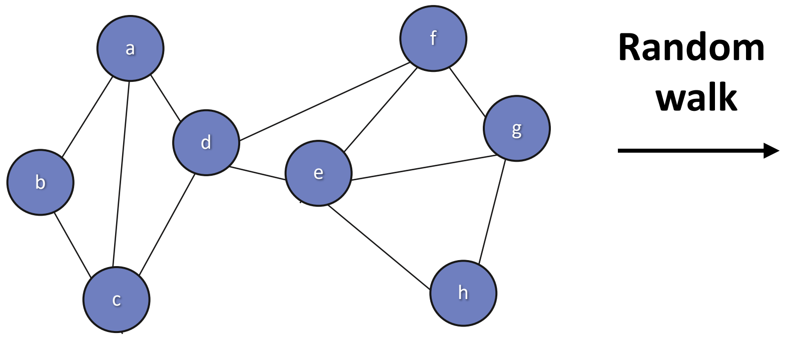
Где “в среднем” будет находиться частица?

Ответ: в **вершинах с большей степенью** — ведь туда ведёт больше путей.

Формально: стационарное распределение π — это такое распределение на вершинах,
которое **не меняется при применении матрицы переходов**:

$$\pi = \pi P.$$

Graph Neural Networks (GNN)



Стационарное распределение π удовлетворяет $\pi = \pi P$, $\sum_i \pi_i = 1$, $\pi_i \geq 0$.

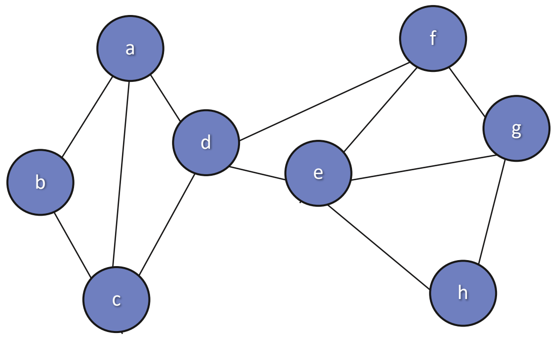
Для неориентированного графа без весов:

$$\pi_i = \frac{\deg(i)}{\sum_v \deg(v)} = \frac{\deg(i)}{2|E|}.$$

Свойство обратимости (detailed balance) (для неориентированного невзвешенного):

$$\pi_i P_{ij} = \pi_j P_{ji} = \frac{1}{2|E|}, \quad \forall (i,j) \in E.$$

Graph Neural Networks (GNN)



Random walk

Марковская цепь называется **эргодической**,
если при любом начальном состоянии X_0
распределение $P(X_t = i)$ **сходится** к π_i при $t \to \infty$:

$$\lim_{t \to \infty} P(X_t = i) = \pi_i.$$

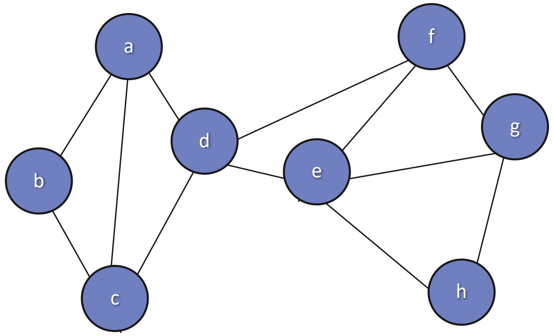
Связь с временем пребывания

Если процесс эргодичен, то доля времени, проведённая в вершине i,
сходится к её стационарной вероятности:

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \mathbf{1}_{\{X_t = i\}} = \pi_i.$$

То есть — **эмпирическая частота ≈ вероятность.**

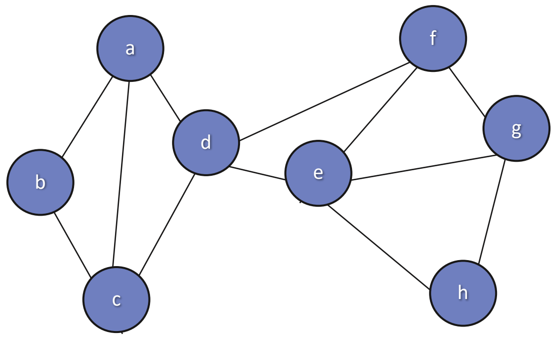
Graph Neural Networks (GNN)



Random walk

№	Seq.
1	1,3,7,3,5,7,5,1
2	4,8,4,2,8,4,1,4
3	5,7,5,1,5,1,2,1
4	8,6,2,6,2,6,8,6
...	...
N	1,3,1,5,7,5,1,5

Graph Neural Networks (GNN)



Random walk

№	Seq.
1	1,3,7,3,5,7,5,1
2	4,8,4,2,8,4,1,4
3	5,7,5,1,5,1,2,1
4	8,6,2,6,2,6,8,6
...	...
N	1,3,1,5,7,5,1,5

Word2Vec

Graph Neural Networks (GNN)

Для каждой пары (v_i, v_j), где v_j находится в окне вокруг v_i, обучаем модель:

$$\Pr(v_j \mid v_i) = \frac{\exp(\mathbf{z}_{v_i}^\top \mathbf{z}_{v_j})}{\sum_{u \in V} \exp(\mathbf{z}_{v_i}^\top \mathbf{z}_u)}.$$

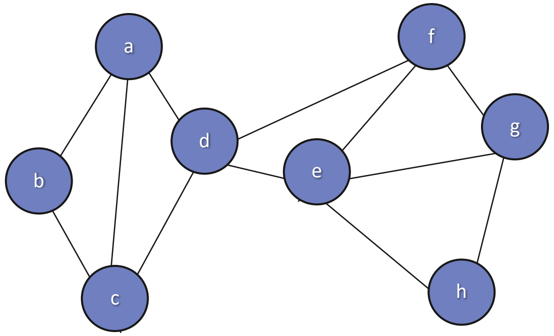
Оптимизация через **негативный семплинг**:

$$\mathcal{L} = -\log \sigma(\mathbf{z}_{v_i}^\top \mathbf{z}_{v_j}) - \sum_{v_k \sim P_n} \log \sigma(-\mathbf{z}_{v_i}^\top \mathbf{z}_{v_k}),$$

где P_n — распределение для негативных примеров.

Graph Neural Networks (GNN)

DeepWalk



Random walk

№	Seq.
1	1,3,7,3,5,7,5,1
2	4,8,4,2,8,4,1,4
3	5,7,5,1,5,1,2,1
4	8,6,2,6,2,6,8,6
...	...
N	1,3,1,5,7,5,1,5

Word2Vec

Output

[0.254 -0.163 -0.912 0.336 0.421 ... -0.503]

[-0.013 0.992 -0.725 -0.425 -0.532 ... -0.667]

...

[0.431 -0.861 0.956 -0.115 -0.038 ... -0.772]

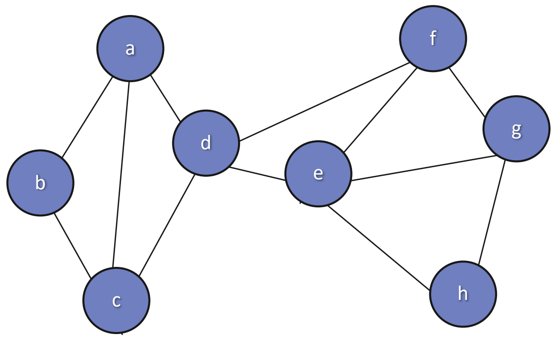
Graph Neural Networks (GNN)

DeepWalk

Когда частица делает шаг по графу,

следующий выбор зависит не только от текущей вершины v,

но и от того, **откуда она пришла** (t).



Random walk

№	Seq.
1	1,3,7,3,5,7,5,1
2	4,8,4,2,8,4,1,4
3	5,7,5,1,5,1,2,1
4	8,6,2,6,2,6,8,6
...	...
N	1,3,1,5,7,5,1,5

Word2Vec

Output

[0.254 -0.163 -0.912 0.336 0.421 ... -0.503]

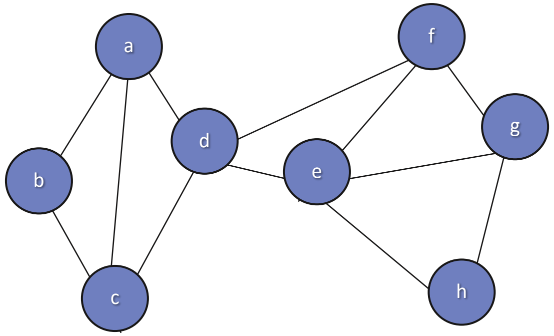
[-0.013 0.992 -0.725 -0.425 -0.532 ... -0.667]

...

[0.431 -0.861 0.956 -0.115 -0.038 ... -0.772]

Graph Neural Networks (GNN)

Node2Vec



Random walk

№	Seq.
1	1,3,7,3,5,7,5,1
2	4,8,4,2,8,4,1,4
3	5,7,5,1,5,1,2,1
4	8,6,2,6,2,6,8,6
...	...
N	1,3,1,5,7,5,1,5

Word2Vec

Output

[0.254 -0.163 -0.912 0.336 0.421 ... -0.503]

[-0.013 0.992 -0.725 -0.425 -0.532 ... -0.667]

...

[0.431 -0.861 0.956 -0.115 -0.038 ... -0.772]

$$\alpha_{p,q}(t,x) = \begin{cases} \dfrac{1}{p}, & \text{если } d_{tx} = 0 \text{ (возврат назад)}, \\ 1, & \text{если } d_{tx} = 1 \text{ (сосед текущего)}, \\ \dfrac{1}{q}, & \text{если } d_{tx} = 2 \text{ (дальше от текущего)}. \end{cases}$$

где d_{tx} — кратчайшее расстояние между предыдущей вершиной t и кандидатом x.

Graph Neural Networks (GNN)

Node2vec

$$\alpha_{p,q}(t,x) = \begin{cases} \frac{1}{p}, & \text{если } d_{tx} = 0 \text{ (возврат назад)}, \\ 1, & \text{если } d_{tx} = 1 \text{ (сосед текущего)}, \\ \frac{1}{q}, & \text{если } d_{tx} = 2 \text{ (дальше от текущего)}. \end{cases}$$

где d_{tx} — кратчайшее расстояние между предыдущей вершиной t и кандидатом x.

Параметр	Контролирует	Эффект
(p) (return parameter)	вероятность **вернуться** назад	маленький → частые возвраты (локальные контексты)
(q) (in-out parameter)	вероятность **идти дальше**	маленький → исследование дальних структур

Graph Neural Networks (GNN)

Связь DeepWalk с Node2Vec

DeepWalk = Node2Vec с $p = q = 1$.

Тогда $\alpha_{p,q}(t, x) = 1$ и переход становится обычным равновероятным.

Node2Vec → обобщает DeepWalk,

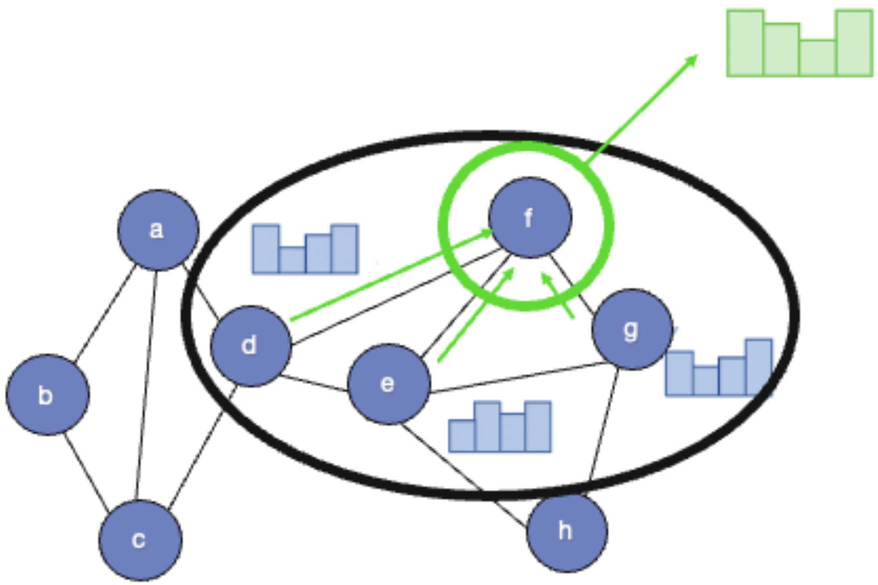
вводя **параметризованное распределение переходов**.

Graph Neural Networks (GNN)

Недостатки таких подходов

Ограничение	Почему важно
Эмбеддинги **фиксированные**	нельзя обновить под конкретную задачу (например, классификацию узлов)
Не учитываются **признаки узлов (features)**	работает только со структурой графа
Обучение **не end-to-end**	эмбеддинги → отдельно, классификатор → отдельно
Невозможно **передавать информацию** от соседей обучаемо	веса "передачи" не обучаются

Graph Neural Networks (GNN)



А можно ли сделать “передачу информации по графу” не через вероятности блужданий,
а через **обучаемое сообщение (message)** между узлами?

Message passing. Агрегация соседей

Основные шаги:

1. $AGGREGATE$ – построение сообщения m

Агрегация для узла v определяется как:

$$h_v^{(k+1)} = AGGREGATE(\{h_u^{(k)} \mid u \in N(v)\}),$$

где:

- $h_v^{(k+1)}$: новое представление узла v после $k + 1$ -й итерации.
- $N(v)$: множество соседей узла v.
- $AGGREGATE$: функция агрегации

Message passing. Агрегация соседей

Важно! Функция агрегации должна быть инвариантна к перестановкам

Примеры функций агрегации:

1. Суммирование:

$$h_v^{(k+1)} = \sum_{u \in N(v)} h_u^{(k)}$$

2. Усреднение:

$$h_v^{(k+1)} = \frac{1}{|N(v)|} \sum_{u \in N(v)} h_u^{(k)}$$

3. Максимум:

$$h_v^{(k+1)} = max_{u \in N(v)} h_u^{(k)}$$

Message passing. Обновление

Основные шаги:

2. $UPDATE$ – обновление сообщения m

$h_a^{(k+1)} = UPDATE(h_a^{(k)}, m_{N(a)}^{(k)})$

Общая формула обновления:

$$h_v^{(k+1)} = f(h_v^{(k)}, \quad AGGREGATE\ (\{h_u^{(k)} \mid u \in N(v)\})$$

где:

f – обучаемая функция (например, линейное преобразование + активация).

Пример: вектор признаков узла v может обновляться следующим образом:

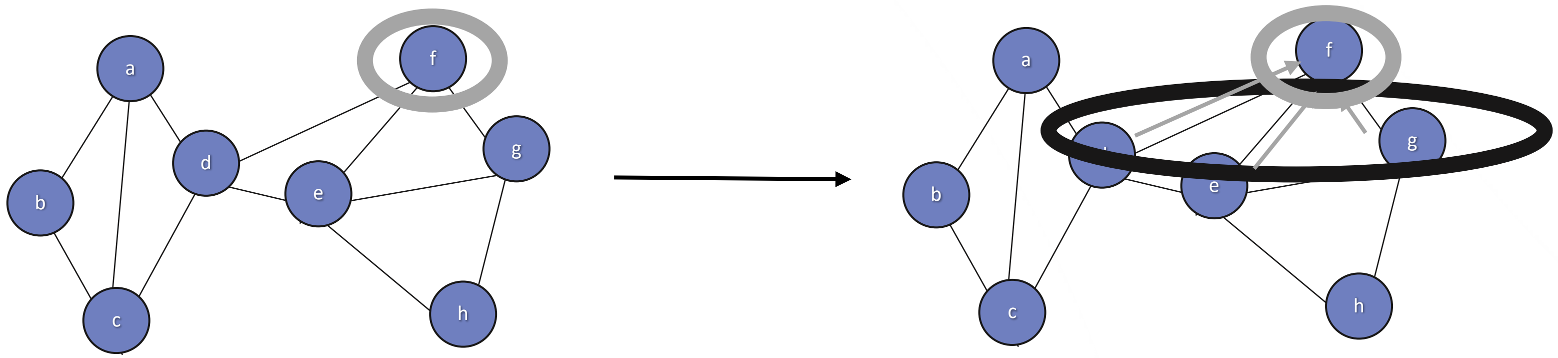
$$h_v^{(k+1)} = ReLU\left(W \cdot (\sum_{u \in N(v)} h_u^{(k)} + h_v^{(k)})\right)$$

где:

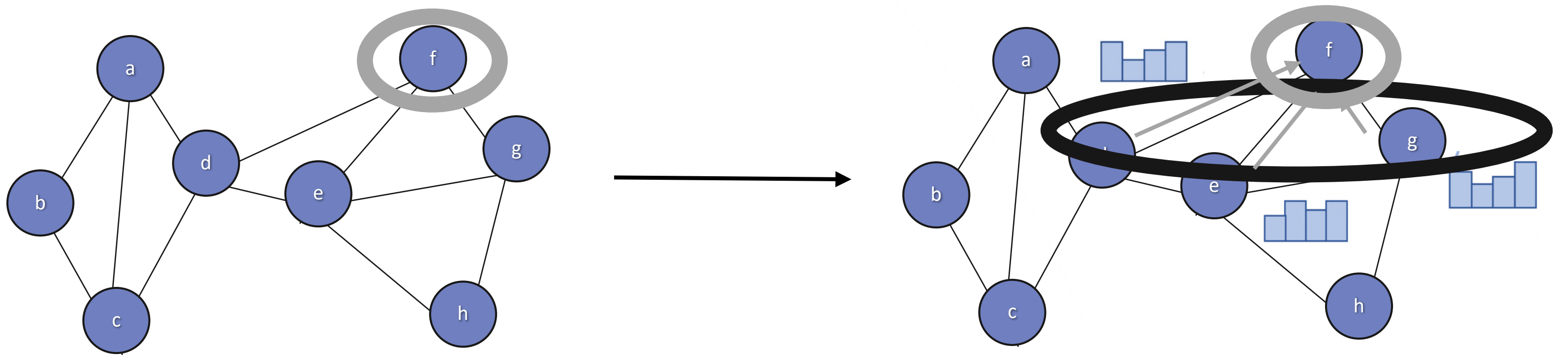
W – обучаемая матрица весов.

$ReLU$ – функция активации

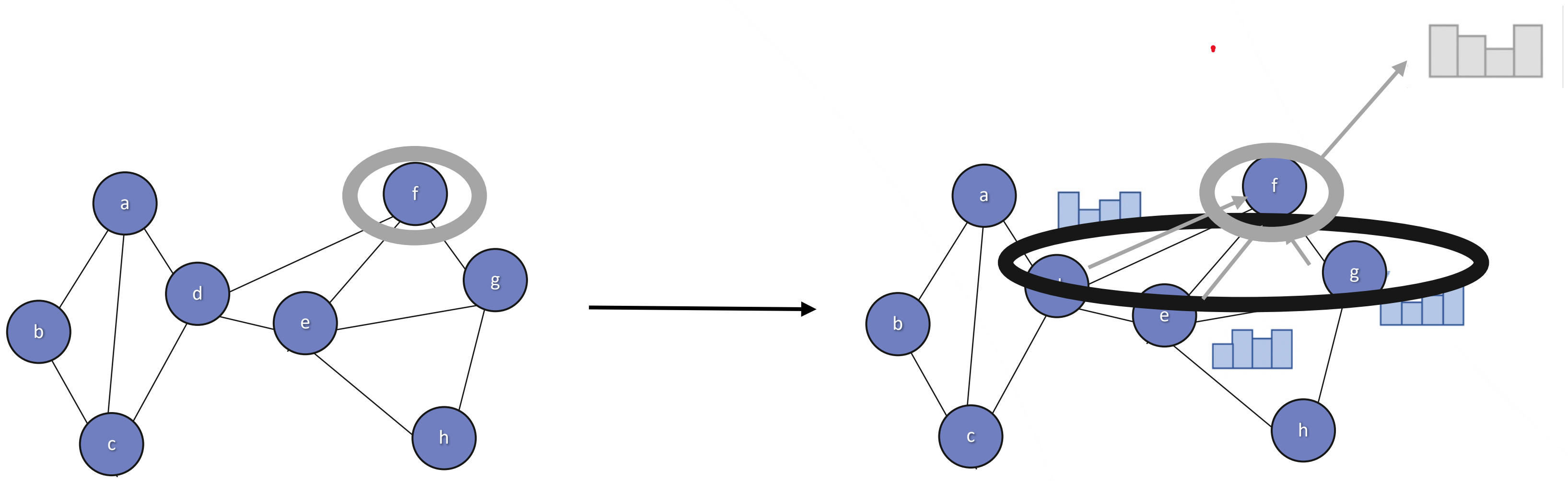
Message passing



Message passing



Message passing



Message passing. Общая формула

Формула для Message Passing:

$$h_v^{(k+1)} = UPDATE\left(h_v^{(k)}, AGGREGATE\left(\left\{ MESSAGE\left(h_u^k, e_{uv}\right) \mid u \in N(v) \right\}\right)\right),$$

где:

- $MESSAGE\left(h_u^k, e_{uv}\right)$ – сообщение от узла u к узлу v, которое может включать признаки рёбра e_{uv}.
- $AGGREGATE$ – агрегирует сообщения.
- $UPDATE$ – обновляет представление узла

message passing описывает процесс распространения информации между узлами графа через передачу сообщений, агрегацию и обновление.
Этот метод используется на каждом слое GNN для создания локальных и глобальных представлений узлов.

Формула узловых эмбедингов

После нескольких шагов Message Passing каждый узел агрегирует информацию от своих соседей и получает векторное представление, которое учитывает как локальные, так и глобальные свойства графа.

Формула:
На k -м слое нейронной сети:

$$h_v^{(k+1)} = f(h_v^{(k)}, AGGREGATE\ (\{h_u^{(k)} \mid u \in N(v)\}),$$

где $h_v^{(k+1)}$— это эмбединг узла v после $(k+1)$-го шага.

Источники

- **Perozzi, B., Al-Rfou, R., & Skiena, S.** (2014). *DeepWalk: Online learning of social representations.* In *Proceedings of the 20th ACM SIGKDD* - https://arxiv.org/abs/1403.6652
  — Введение метода **DeepWalk**, связывающего random walk и Word2Vec.
- **Grover, A., & Leskovec, J.** (2016). *node2vec: Scalable feature learning for networks.* In *Proceedings of the 22nd ACM SIGKDD* (pp. 855–864) - https://arxiv.org/abs/1607.00653
  — Модификация DeepWalk с параметрами ppp и qqq для гибридных обходов графа.
- **Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E.** (2017). *Neural message passing for quantum chemistry.* In *ICML* (pp. 1263–1272) - https://arxiv.org/abs/1704.01212
  — Введена общая **формула message passing**, ставшая стандартом для GNN
- A Comprehensive Survey on Graph Neural Networks — Wu Z. et al. (2019). Обширный обзор GNN, включая раздел про механизм message-passing. **Newman, M. E. J.** (2004). *Finding and evaluating community structure in networks.* Physical Review E, 69(2), 026113 - https://arxiv.org/pdf/1901.00596
  — Статья, где впервые предложена **модульность** и введена основа community detection.
- **Clauset, A., Newman, M. E. J., & Moore, C.** (2004). *Finding community structure in very large networks.* Physical Review E, 70(6), 066111 - https://arxiv.org/pdf/condmat/0408187
  — Алгоритм **Clauset–Newman–Moore (CNM)** для жадной оптимизации модульности.