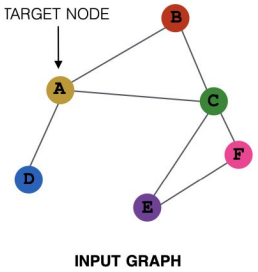
ИТМО

Анализ графовых данных и глубокое обучение

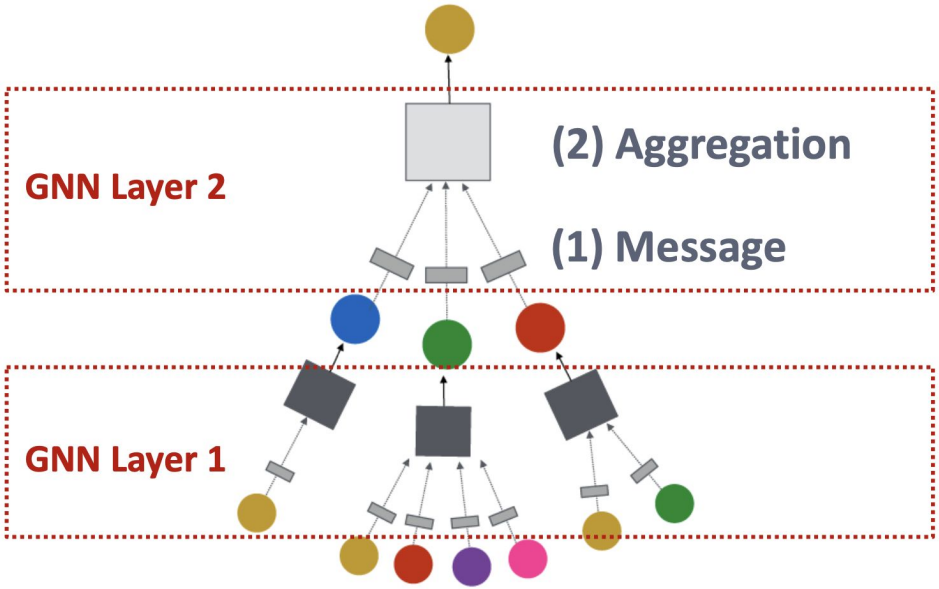
Азимов Рустам
Высшая школа цифровой культуры

В предыдущих сериях

ИТМО



(5) Learning objective



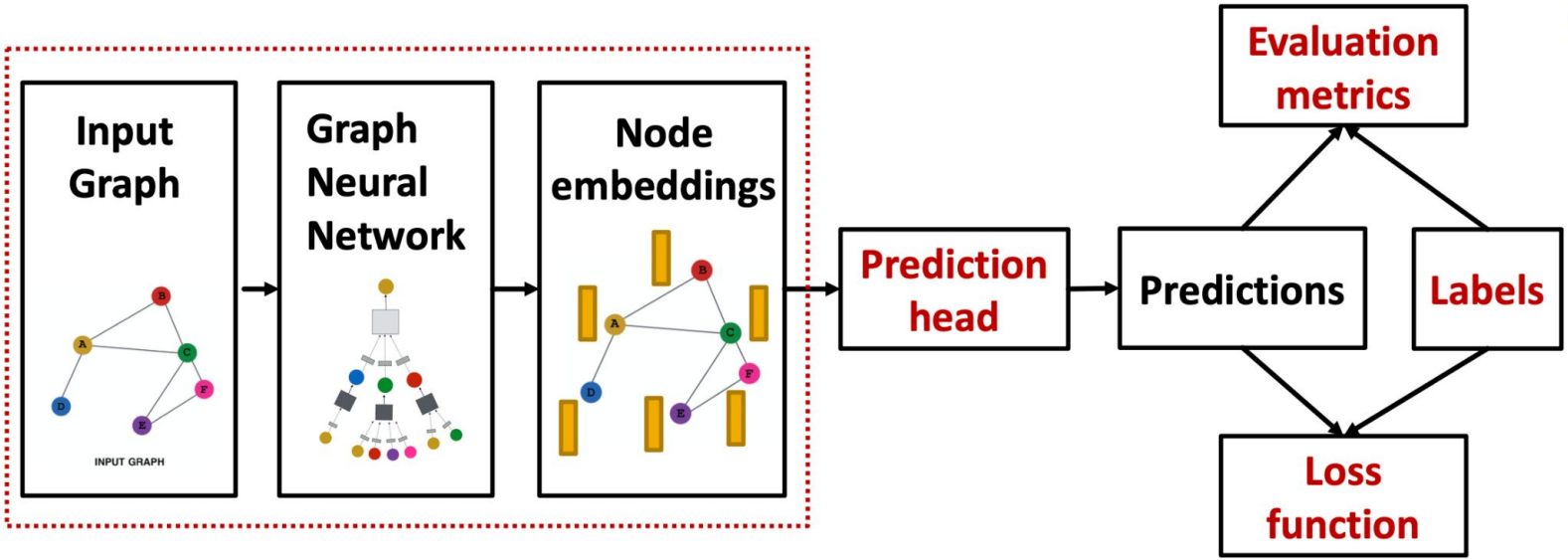
(3) Layer connectivity

(4) Graph augmentation

ИТМО

GNN Training Pipeline

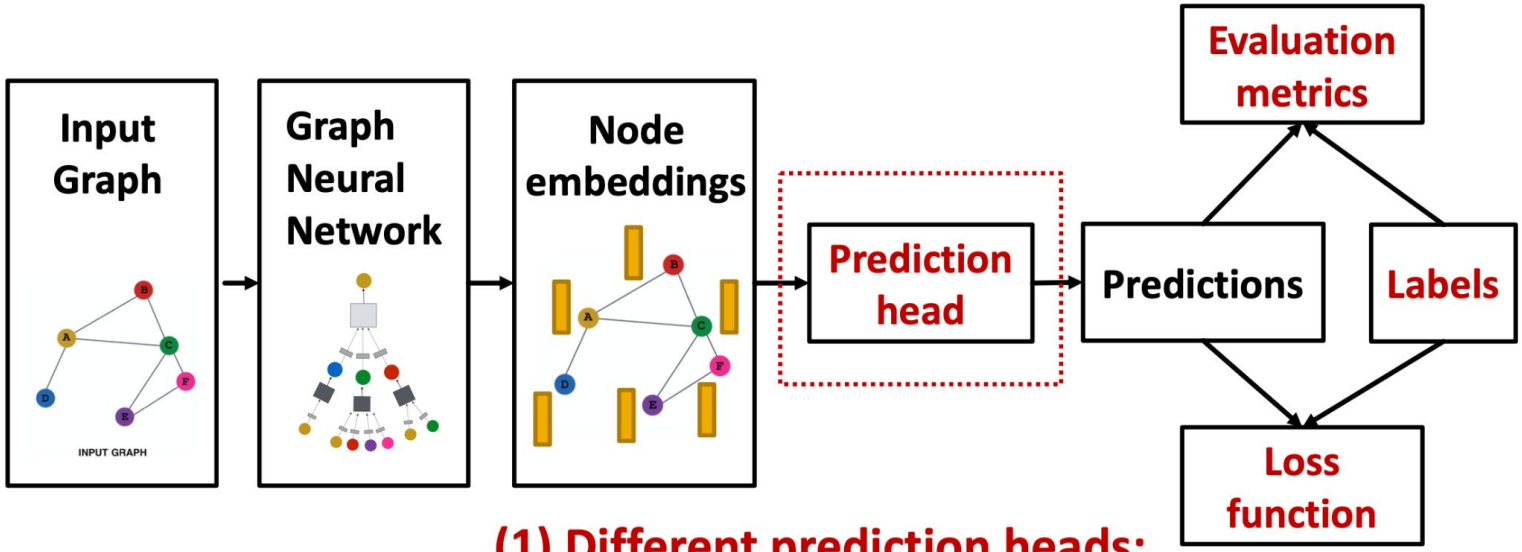
Training Pipeline



Output of a GNN: set of node embeddings

$$\{\mathbf{h}_v^{(L)}, \forall v \in G\}$$

Prediction head



(1) Different prediction heads:

- Node-level tasks
- Edge-level tasks
- Graph-level tasks

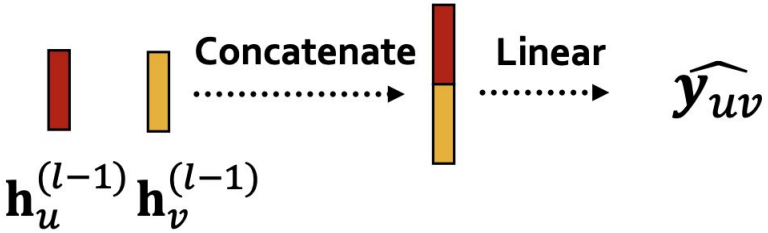
Node-level head

- Можно использовать линейный слой над обученными эмбеддингами $h_v^{(l)}$

$$\hat{\boldsymbol{y}}_{\boldsymbol{v}} = \mathrm{Head}_{\mathbf{node}}(\mathbf{h}_{\boldsymbol{v}}^{(L)}) = \mathbf{W}^{(H)}\mathbf{h}_{\boldsymbol{v}}^{(L)}$$

- Размер линейного слоя зависит от количества целевых признаков *y* для задачи регрессии/классификации

Edge-level head (1)



$$\hat{\boldsymbol{y}}_{\boldsymbol{uv}} = \text{Linear}(\text{Concat}(\mathbf{h}_u^{(L)}, \mathbf{h}_v^{(L)}))$$

Edge-level head (2)

ИТМО

- 1-way prediction

$$\hat{y}_{uv} = (\mathbf{h}_u^{(L)})^T \mathbf{h}_v^{(L)}$$

- *k*-way prediction (идея, схожая с multi-head attention в GAT)

$$\hat{y}_{uv}^{(1)} = (\mathbf{h}_u^{(L)})^T \mathbf{W}^{(1)} \mathbf{h}_v^{(L)}$$

$$\dots$$

$$\hat{y}_{uv}^{(k)} = (\mathbf{h}_u^{(L)})^T \mathbf{W}^{(k)} \mathbf{h}_v^{(L)}$$

$$\hat{y}_{uv} = \text{Concat}(\hat{y}_{uv}^{(1)}, \dots, \hat{y}_{uv}^{(k)}) \in \mathbb{R}^k$$

Graph-level head

ИТМО

(1) Global mean pooling

$$\hat{\mathbf{y}}_G = \text{Mean}(\{\mathbf{h}_v^{(L)} \in \mathbb{R}^d, \forall v \in G\})$$

(2) Global max pooling

$$\hat{\mathbf{y}}_G = \text{Max}(\{\mathbf{h}_v^{(L)} \in \mathbb{R}^d, \forall v \in G\})$$

(3) Global sum pooling

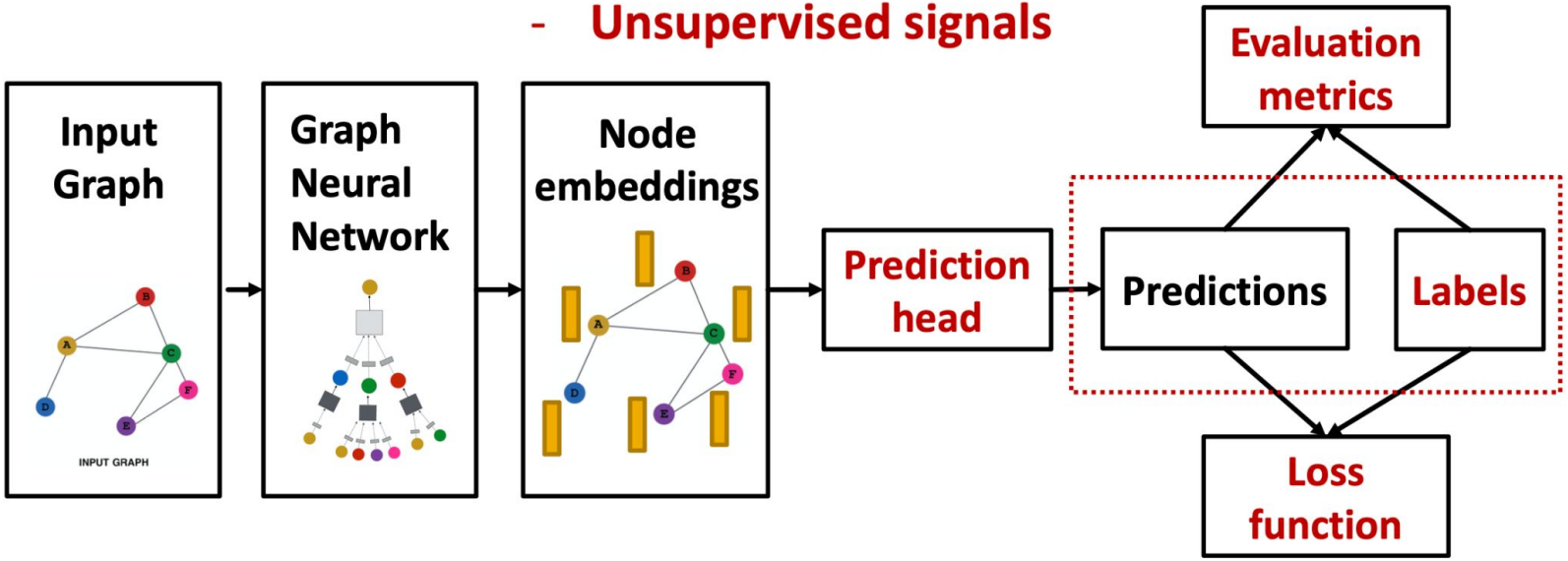
$$\hat{\mathbf{y}}_G = \text{Sum}(\{\mathbf{h}_v^{(L)} \in \mathbb{R}^d, \forall v \in G\})$$

Training Pipeline

ИТМО

(2) Where does ground-truth come from?

- **Supervised labels**
- **Unsupervised signals**



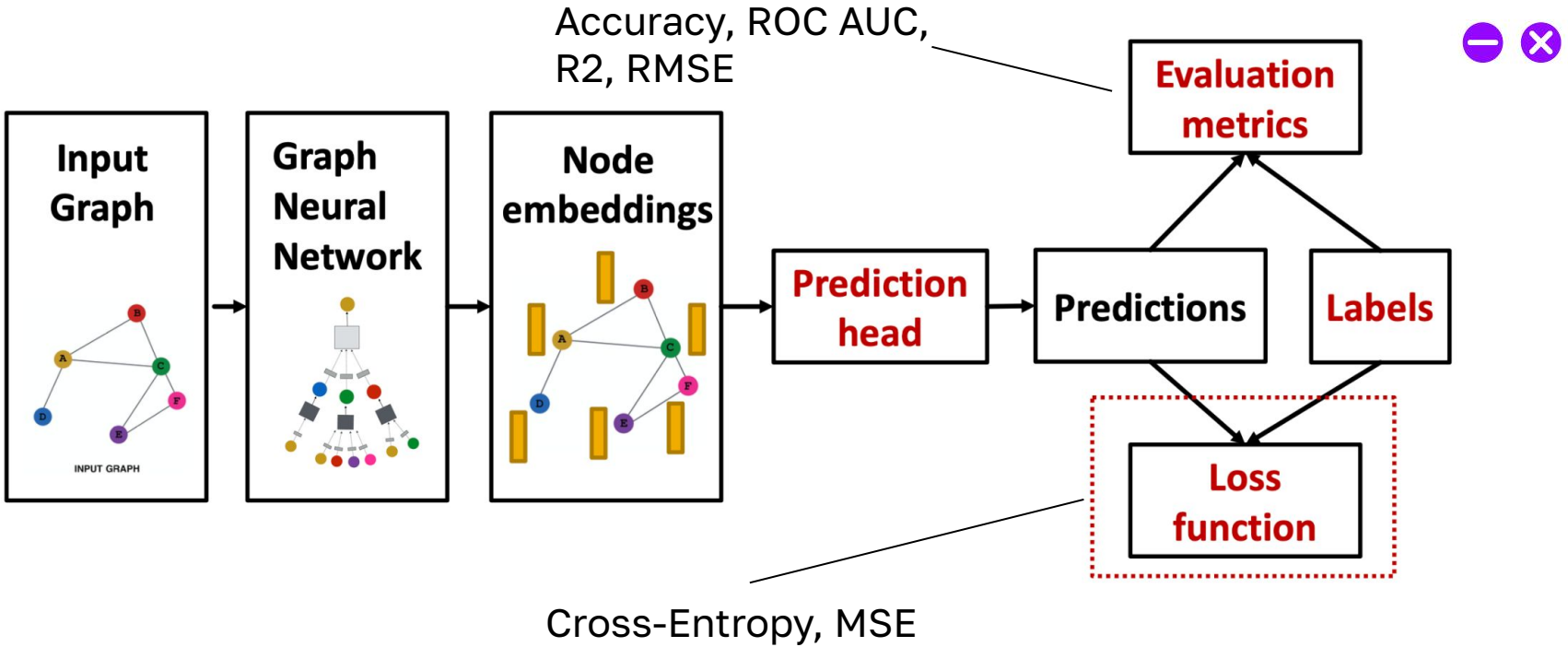
Supervised Labels

- Метки вершин, рёбер или графов уже даны
- Например, каждый граф — лекарство и известны некоторые токсичные и нетоксичные лекарства

Unsupervised Signals

- Дан только граф/графы без меток
- Можно самостоятельно выделить метки для предсказания
- Для вершин, например, предсказывать clustering coefficient, PageRank, ...
- Или к какому кластеру принадлежит вершина
- Для рёбер — убрать некоторые из них и научится предсказывать их наличие
- Для графов, например, предсказывать изоморфны ли два графа

Training Pipeline



Dataset Splitting

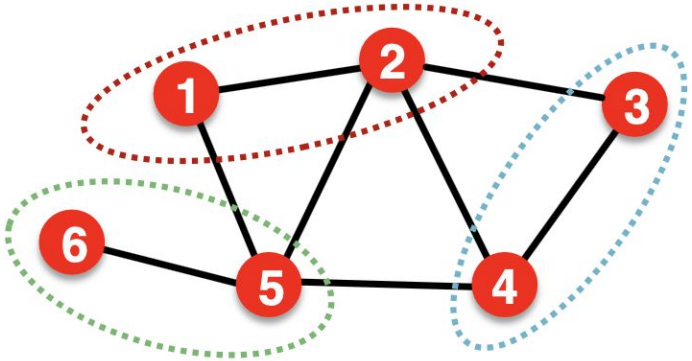
ИТМО

- Как разделить набор графовых данных на train/validation/test?
- Легко, если набором является множество графов
- В чём сложность, когда имеется только один граф?

Training

Validation

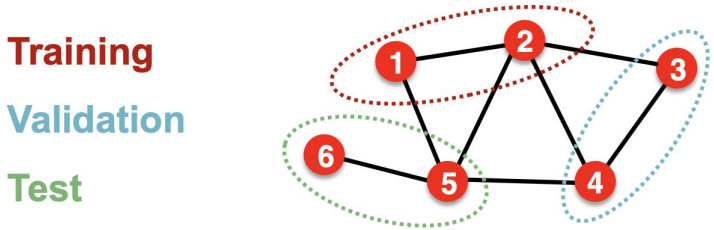
Test



Transductive Setting

ИТМО

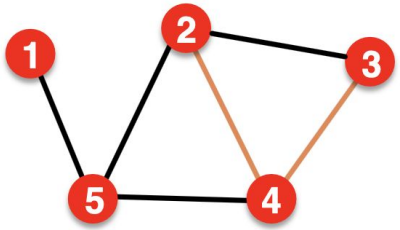
- Позволяем быть влиянию частей набора данных друг на друга
- На каждом этапе доступен весь граф
- При обучении вычисляются эмбеддинги с использованием всего графа, но loss оптимизируется только для тренировочных данных
- Аналогично при валидации метрику подсчитываем на валидационных данных



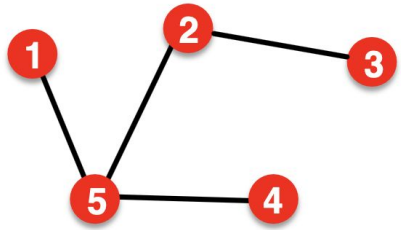
Пример: Link Prediction

ИТМО

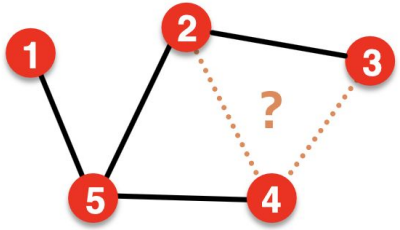
- Отдельного внимания заслуживает задача предсказания связей в графе
- Прячем некоторые рёбра, чтобы на них оптимизировать предсказания



Original graph

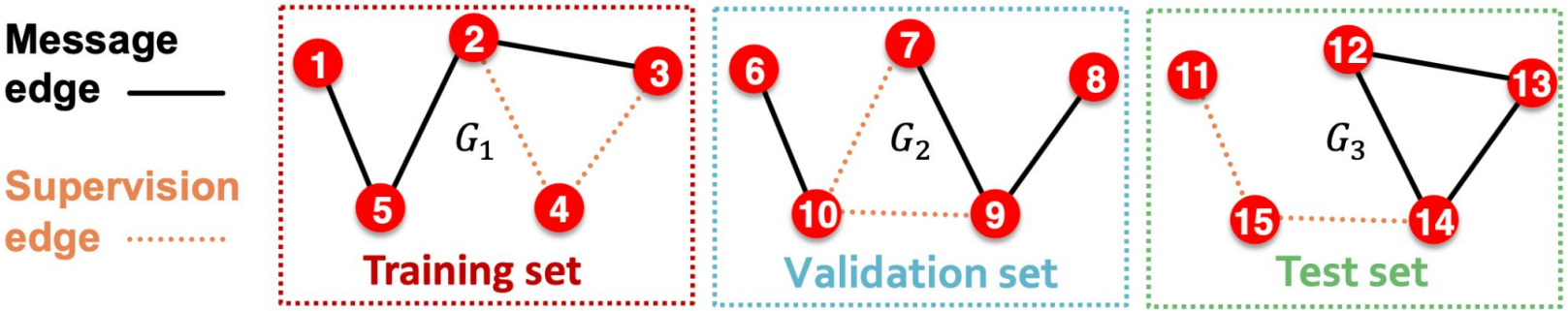


Input graph to GNN

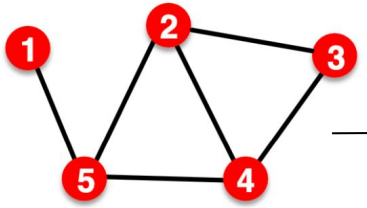


Predictions made by GNN

Inductive Link Prediction

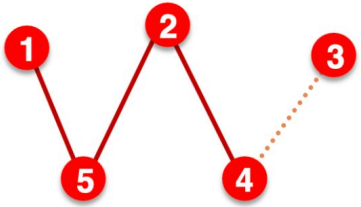


Transductive Link Prediction

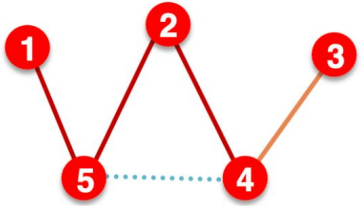


The original graph

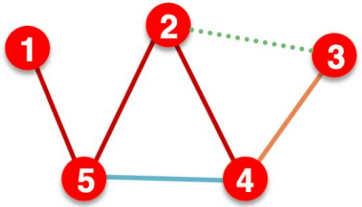
Training message edges
Training supervision edges
Validation edges
Test edges



(1) At training time:
Use **training message edges** to predict **training supervision edges**



(2) At validation time:
Use **training message edges & training supervision edges** to predict **validation edges**



(3) At test time:
Use **training message edges & training supervision edges & validation edges** to predict **test edges**

ИТМО

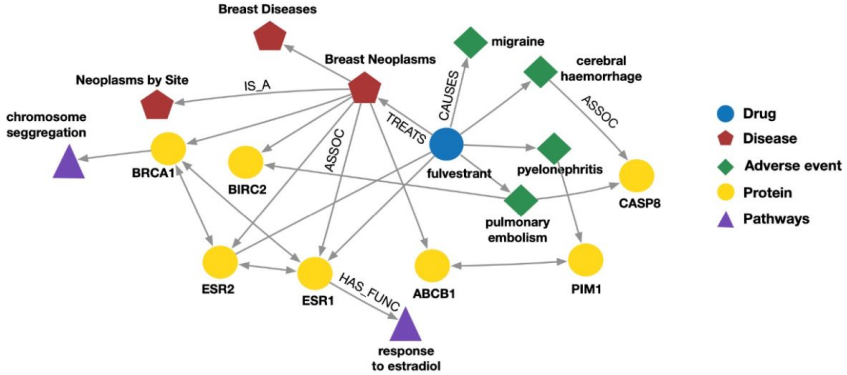
Heterogeneous Graphs

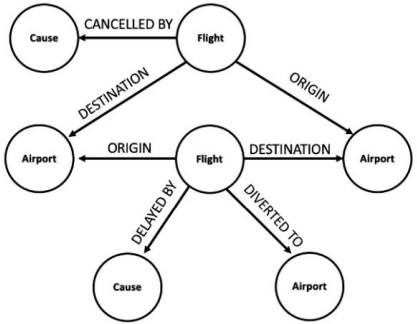
Гетерогенные графы

ИТМО

- Мы рассмотрели pipeline для однородных графов
- Но часто на практике встречаются гетерогенные графы с несколькими видами вершин и рёбер

Гетерогенные графы





Biomedical Knowledge Graphs

Example node: Migraine
Example relation: (fulvestrant, Treats, Breast Neoplasms)
Example node type: Protein
Example edge type: Causes

Event Graphs

Example node: SFO
Example relation: (UA689, Origin, LAX)
Example node type: Flight
Example edge type: Destination

Гетерогенные графы

ИТМО

- Есть много способов сделать гетерогенный граф гомогенным и использовать рассмотренные в предыдущих лекциях методы
- Например, one-hot кодированием добавить информацию о типе вершин в их признаки (вершины-авторы [1, 0], вершины-статьи [0, 1]
- Но иногда нужно использовать именно гетерогенные графы
    - У разного типа вершин/рёбер разная размерность признаков
    - Существенное отличие связей требует обучения разных моделей

GCN

ИТМО

Message

$$\mathbf{h}_v^{(l)} = \sigma\left(\sum_{u \in N(v)} \mathbf{W}^{(l)} \frac{\mathbf{h}_u^{(l-1)}}{|N(v)|}\right)$$

Aggregation

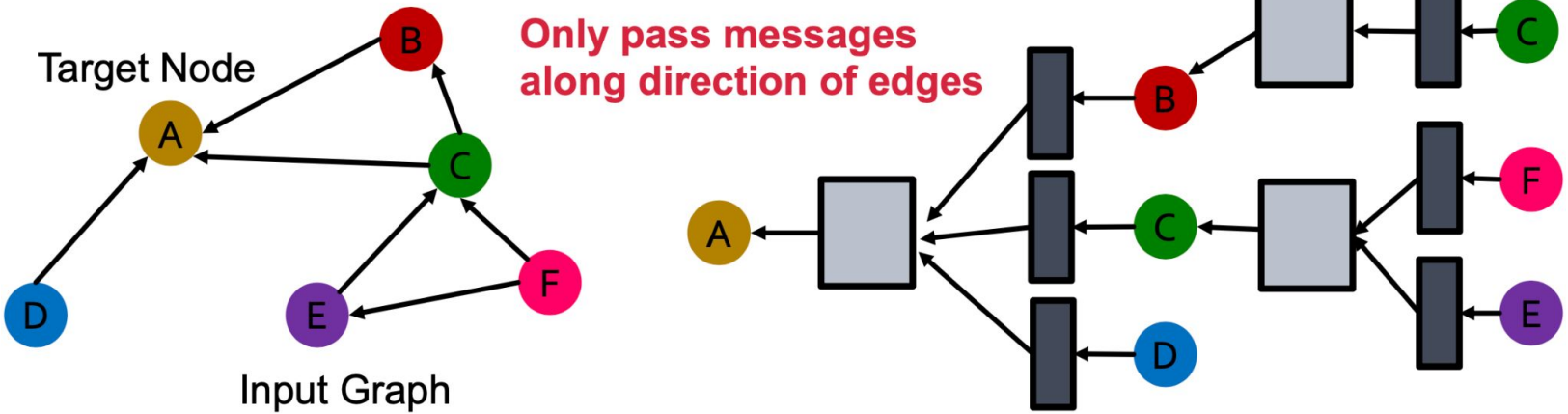
- Сообщения от соседей: $\mathbf{m}_u^{(l)} = \frac{1}{|N(v)|}\mathbf{W}^{(l)}\mathbf{h}_u^{(l-1)}$
- Агрегация:

$$\mathbf{h}_v^{(l)} = \sigma\left(\text{Sum}\left(\left\{\mathbf{m}_u^{(l)}, u \in N(v)\right\}\right)\right)$$

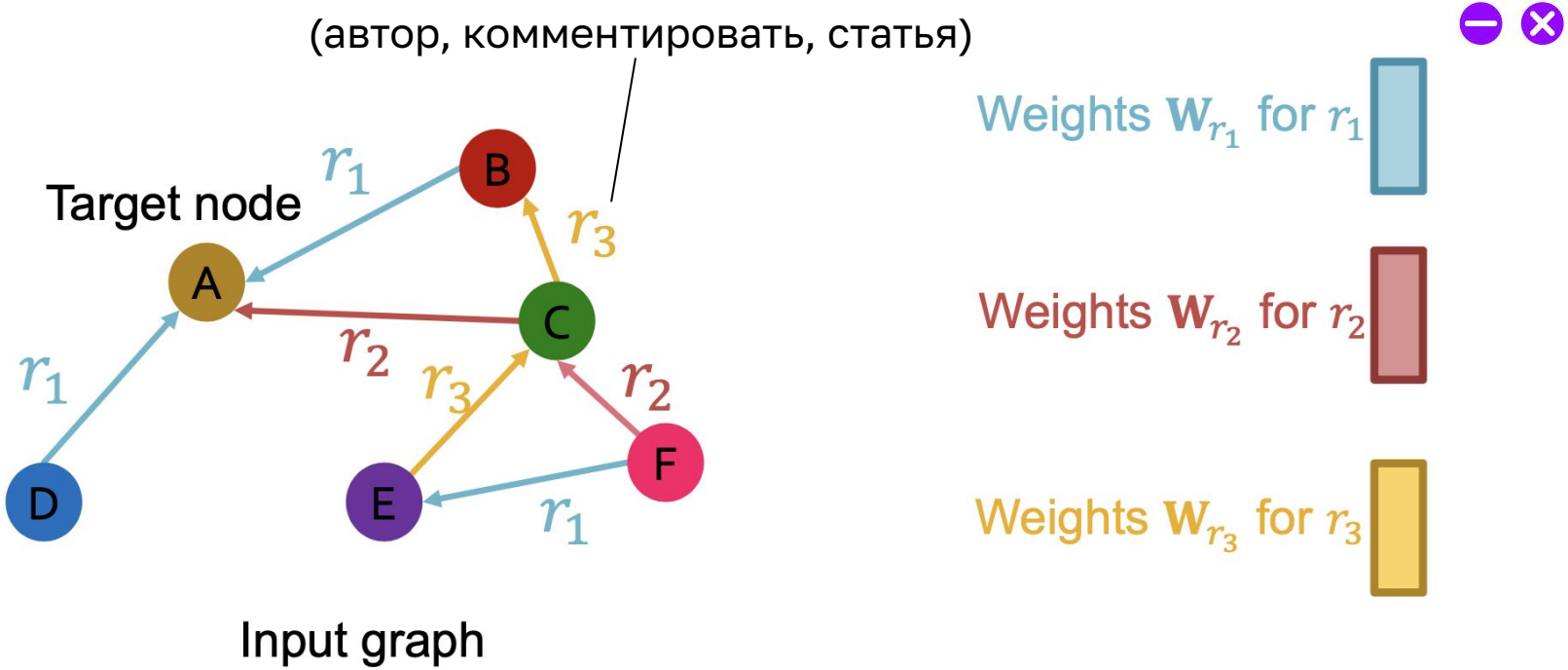
- В GCN использовали петли для каждой вершины, чтобы одним из соседей была сама вершина

GCN

ИТМО


Target Node
A
B
C
D
E
F
Input Graph
Only pass messages
along direction of edges
A
B
C
D
C
F
E

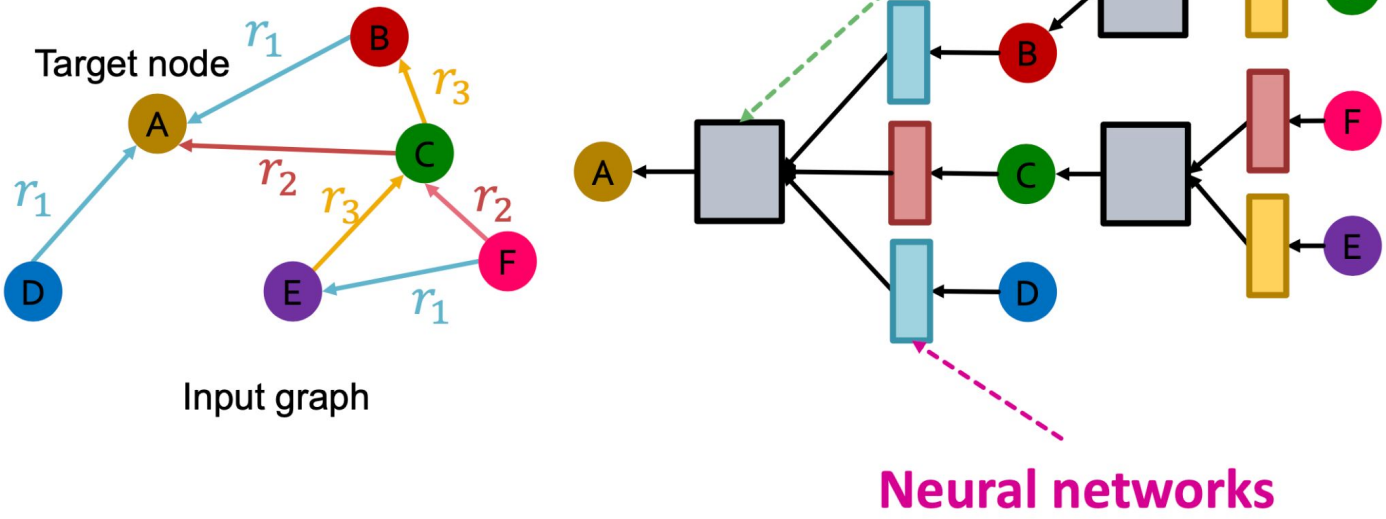
Relational GCN



Relational GCN

ИТМО

- Use different neural network weights for different relation types!



Relational GCN

ИТМО

- Relational GCN (RGCN)

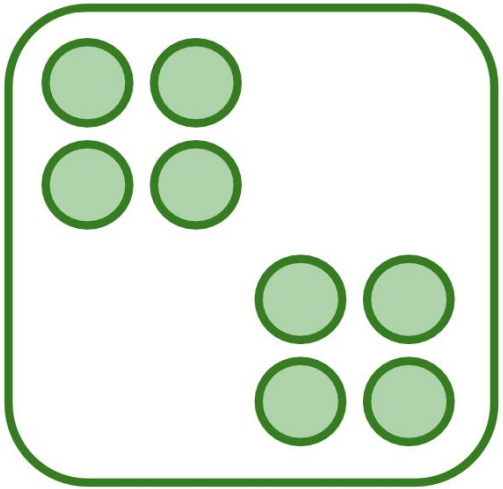
$$\mathbf{h}_v^{(l+1)} = \sigma\left(\sum_{r\in R}\sum_{u\in N_v^r}\frac{1}{c_{v,r}}\mathbf{W}_r^{(l)}\mathbf{h}_u^{(l)} + \mathbf{W}_0^{(l)}\mathbf{h}_v^{(l)}\right)$$

- Сообщения нормализуются отдельно по каждому типу отношений
- Существенно увеличивается количество параметров с ростом числа отношений
- Появляется проблема переобучения
    - Использовать блочно-диагональные матрицы весов
    - Использовать базис весов

Block Diagonal Weights

ИТМО

$$\mathbf{W}_r =$$



Limitation: only nearby neurons/dimensions can interact through W

Basis Learning

ИТМО

Базисные матрицы весов

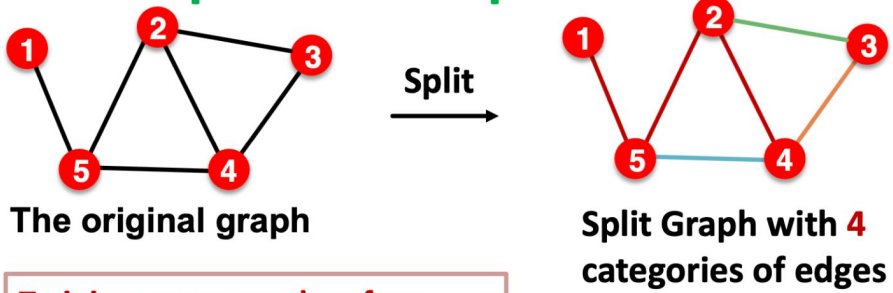
$$\mathbf{W}_r = \sum_{b=1}^{B} a_{rb} \cdot \mathbf{V}_b$$

Обучаемые скаляры для каждого отношения

Пример: Link Prediction

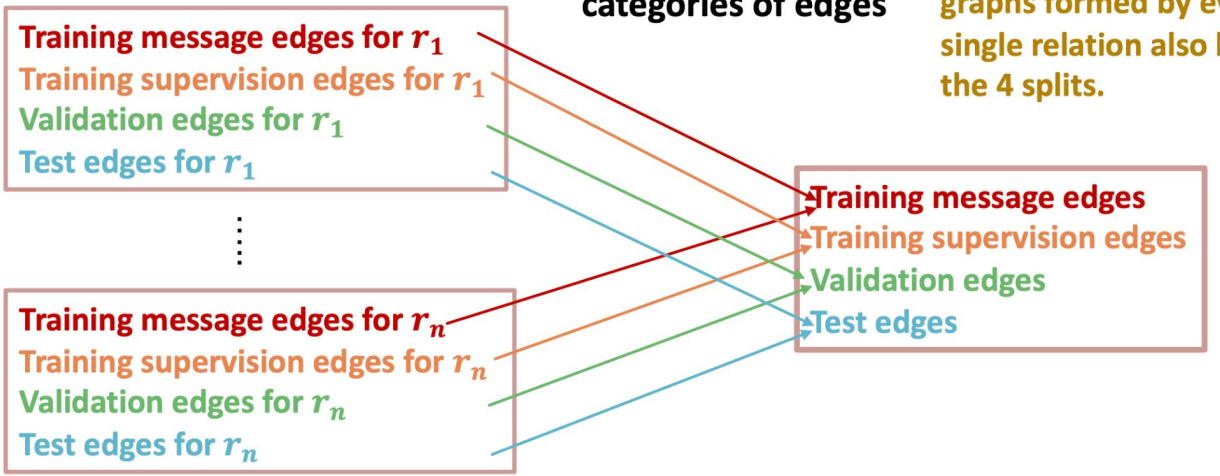
ИТМО

- **Link prediction split:**



Every edge also has a relation type, this is independent of the 4 categories.

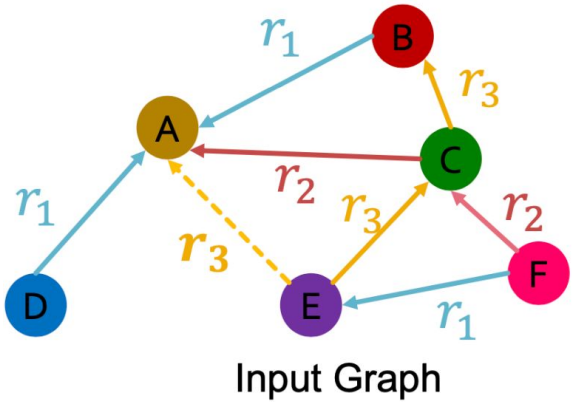
In a heterogeneous graph, the homogeneous graphs formed by every single relation also have the 4 splits.



Пример: Link Prediction

ИТМО

- **Training:**



Input Graph

1. Use RGCN to score the training supervision edge (E, r_3, A)
2. Create a negative edge by perturbing the supervision edge (E, r_3, B)
3. Use GNN model to score negative edge
4. Optimize a standard cross entropy loss

1. Maximize the score of training supervision edge
2. Minimize the score of negative edge

$$\ell = -\log \sigma\left(f_{r_3}(h_E, h_A)\right) - \log(1 - \sigma(f_{r_3}(h_E, h_B)))$$

Relational GNN

ИТМО

- Идеи RGCN могут быть применены и для создания RGraphSAGE, RGAT, ...
- При вычислении сообщений от соседей можно обучать отдельные функции для каждого отношения $\mathbf{m}_u^{(l)} = \mathbf{W}_r^{(l)} \mathbf{h}_u^{(l-1)}$
- Агрегировать можно в два этапа

$$\mathbf{h}_v^{(l)} = \text{Concat}\left(\text{Sum}\left(\left\{\mathbf{m}_u^{(l)}, u \in N_r(v)\right\}\right)\right)$$

- Отдельный (пре)постпроцессинг для каждого типа вершин

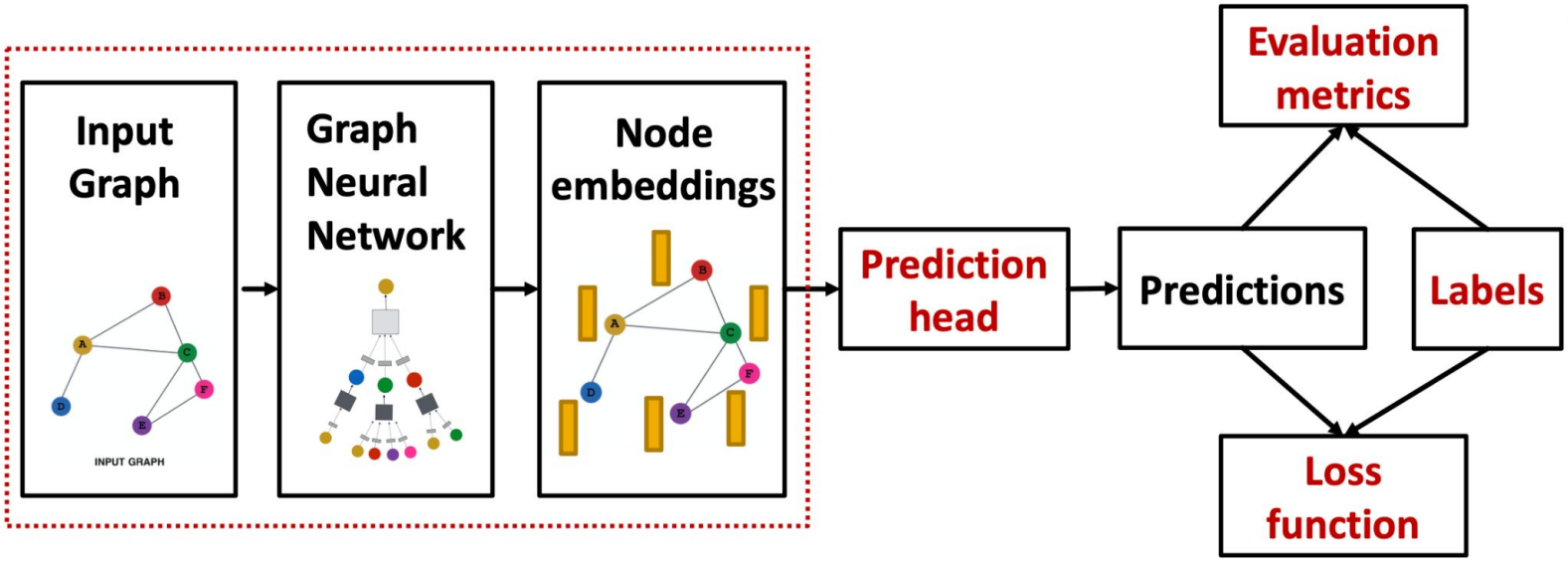
$$\mathbf{h}_v^{(l)} = \text{MLP}_{T(v)}(\mathbf{h}_v^{(l)})$$

  - $T(v)$ is the type of node v

Heterogeneous Graph Manipulation

- Feature manipulation
  - Добавлять в признаки различные статистики (clustering coefficient, ...) можно по отдельности для каждого типа отношений или как обычно, используя весь граф
- Graph structure manipulation
  - Сэмплирование соседей/подграфа стратифицировано по каждому отношению или как обычно из всего графа

Заключение



Output of a GNN: set of node embeddings

$$\{\mathbf{h}_v^{(L)}, \forall v \in G\}$$