

1 Формулировка проблемы

Введение

Все методы обучения представлений графовых данных можно разделить на две категории: контролируемые и неконтролируемые. Трудность контролируемых задач состоит в том, что для обучения требуется много размеченных данных. А в качестве функции потерь всегда используется перекрёстная энтропия. Хотя чаще бывает так, что размеченных данных мало. Одним из выходов из ситуации служит неконтролируемое обучение, которое строит векторы в низкоразмерном пространстве, основываясь на структуре графа, а уже в векторном пространстве решаются последующие задачи. Функция потерь в данном классе методов является важным элементом, так как именно она определяет то, как будут расположены относительно друг друга вектора в векторном пространстве. Общая идея функций потерь неконтролируемого построения представлений – устремлять ближе друг к другу вектора тех вершин, что и в графе «близки/схожи». В векторном пространстве достаточно просто определить понятие близости – разница или скалярное произведение векторов. Поэтому отличия функций потерь скорее ограничиваются различиями в определении такого же понятия между вершинами в самом графе.

Кроме упомянутой в начале классификации построений представлений графовых данных существует еще одна, разделяющая методы на трансдуктивные и индуктивные. Первый класс методов не позволяет переводить в вектора вершины, не участвующие в тренировке, так как настраивает сами вектора вершин. Второй класс методов к каким-либо образом инициализированным представлениям вершин применяет «графовые свертки». В процессе тренировки настраиваются веса сверточного слоя, что позволяет работать с новыми вершинами.

Наиболее общими классами методов являются неконтролируемые и индуктивные, поэтому они и будут рассмотрены в данной работе.

Постановка задачи

Обозначим матрицу Φ , состоящую из векторов - представлений вершин графа в низкоразмерном пространстве следующим образом:

$$\Phi = \text{Alg}(G, W^{(1)}, \dots, W^{(K)})$$

Тогда задача заключается в поиске оптимальных значений весов функции потерь \mathcal{L} , зависящей от Φ , а значит, по транзитивности, и от весов $W^{(k)}$, а также, возможно, и от какого-то дополнительного параметра C , разного для каждого из методов:

$$\{W^{(1)*}, \dots, W^{(K)*}\} = \arg \min_{\{W^{(k)}: W^{(k)} \in \mathbb{R}^2\}} \mathcal{L}(\Phi, C)$$

где $G = (V, E)$ – граф. Alg – способ перевода вершин графа в представления, состоящий из последовательности графовых сверток $\text{Conv}^{(k)}$ (например, GCN, GAT и тд.) с нелинейным преобразованием между ними (ReLU, tanh, sigmoid и пр). $W^{(k)}$ – веса k – го слоя

из Alg. В общем виде, каждая свёртка записывается следующим образом:

$$\text{Conv}^{(k)}(W^{(k)}, G) = \sigma(W^{(k)} \cdot \text{AGGREGATE}(G)),$$

функция *AGGREGATE* собирает информацию по соседям, в зависимости от свертки это может быть - сумма/среднее/конкатенация представлений соседних вершин для каждой рассматриваемой.

Можно заметить, что отличие между свертками Conv в способе собирания информации от соседей. А отличие функций потерь друг от друга в способе представления расстояния между вершинами в графе.

В имеющихся сравнительных анализах [1, 2] методы рассматриваются в том виде, как представлены в оригинальных статьях, то есть в конкретных сочетаниях сверток с функциями потерь, показавшими наибольшую эффективность для конкретного метода, или и вовсе в трансдуктивной манере. Поэтому целью данной работы, является абстрагирование от способа перевода вершин в низкоразмерное пространство и акцентирование внимание на функциях потерь. И, как следствие, для фиксированных сверток и на разных задачах необходимо оценить как работают различные функции потерь, чтоб выявить их принципиальные различия и определить сходства.

Список используемых источников

- [1] Khosla, M.; Setty, V.; Anand, A. A Comparative Study for Unsupervised Network Representation Learning. IEEE Trans. Knowl. Data Eng. 2019
- [2] W. L. Hamilton, R. Ying, and J. Leskovec, “Representation learning on graphs: Methods and applications,” arXiv:1709.05584, 2017.
- [3] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2018. Deep Graph Infomax. In International Conference on Learning Representations (ICLR).

2 Необходимые обозначения

Таблица 1: Необходимые обозначения

Обозначение	Расшифровка
$G = (V, E)$	Граф
$V = \{v_i\}$	непустое множество вершин
$ V = n$	Количество вершин в графе
$E = \{e_{ij}\}$	множество пар вершин, называемых ребрами
e_{ij}	это ребро между вершинами v_i и v_j
$ E = m$	Количество рёбер в графе
$\mathbf{A} = \{a_{ij}\}$	Матрица смежности
\mathbf{I}	Единичная матрица
\mathbf{D}	Матрица степеней вершин графа
$\mathbf{P} = \mathbf{D}^{-1} \mathbf{A}$	Матрица переходов
$\mathbf{C} = \{c_{i,j}\}$	Матрица схожести вершин. В простом случае $\mathbf{C} = \mathbf{A}$
r	Размерность признакового пространства
\mathbf{X}	Матрица признаков вершин, размерностью $n \times r$
\mathcal{L}	Функция потерь
d	Размерность эмбедингов
Φ, Θ	Матрицы эмбедингов, размерностью $n \times d$
Φ_i, Θ_i	Ряд в матрицах эмбедингов. Исходный и контекстный эмбединги вершины v_i
$N(v)$	Соседи вершины v внутри окна определенного размера последовательности случайного блуждания

3 Таблица с функциями потерь

Таблица 2: Методы неконтролируемого обучения

Метод	Контекстный эмбединг	Функция потерь	Подходы к оптимизации, используемые методы
Laplacian EigenMaps	×	$\frac{1}{2} \sum_{i,j} \Phi_i - \Phi_j ^2 a_{i,j}$	Matrix Factorization
Graph Factorization	×	$\frac{1}{2} \sum_{i,j} (a_{i,j} - \Phi_i \cdot \Phi_j)^2 + \frac{\lambda}{2} \sum_i \ \Phi_i\ ^2$	SGD with asynchronous optimization (ASGD)
HOPE	✓	$\ \mathbf{C} - \Phi \cdot \Theta\ _F^2$, \mathbf{C} – может быть записана в разном виде, например для RootedPageRank, см. уравнение 1	Matrix Factorization
NetMF	✓	$\ \log(\frac{vol(G)}{bT} (\sum_{r=1}^T \mathbf{P}^r) \mathbf{D}^{-1}) - \Phi \cdot \Theta\ _F^2$ (значение параметров см. в описании к ур-ию 3)	Matrix Factorization
DeepWalk	✓	$-\sum_{v_j \in V} \sum_{v_i \in N(v_j)} \log \frac{\exp(\Theta_i \cdot \Phi_j)}{\sum_{v_k \in V} \exp(\Theta_k \cdot \Phi_j)}$, где $N(v)$ – см. таблицу 1. В данном случае случайные блуждания фиксированной длины	SGD, ASGD. Hierarchical Softmax
Node2Vec	✓	$-\sum_{v_j \in V} \sum_{v_i \in N(v_j)} \log \frac{\exp(\Theta_i \cdot \Phi_j)}{\sum_{v_k \in V} \exp(\Theta_k \cdot \Phi_j)}$ Случайные блуждания имеют два параметра - вероятность перехода к вершинам, отвечающим за исследование локальной и глобальной структур	SGD with negative Sampling

Struc2Vec	✓	$-\sum_{v_j \in V} \sum_{v_i \in N(v_j)} \log \frac{\exp(\Theta_i \cdot \Phi_j)}{\sum_{v_k \in V} \exp(\Theta_k \cdot \Phi_j)}$ Случайные блуждания строятся по контекстному графу, учитываемому структурную схожесть вершин	Hierarchical Softmax
Seed. Эмбе- динги гра- фов !	×	$\ \mathbf{X} - \hat{\mathbf{X}}\ _2^2$, где $\hat{\mathbf{X}}$ – Реконструкция графа.	Gradient descent, deep autoencoders
App	✓	$-\sum_{v_j \in V} \sum_{v_i \in N(v_j)} \log \frac{1}{1+\exp(-\Theta_i \cdot \Phi_j)}$ для построения случайных последовательностей используется метод Monte-Carlo End-Point	Skip-Gram with Negative Sampling
VERSE	×	$-\sum_{i,j=1}^n \text{sim}_G(v_i, v_j) \log \frac{\exp(\Phi_i \cdot \Phi_j)}{\sum_{k=1}^n \exp(\Phi_i \cdot \Phi_k)}$ sim_G – распределение схожести вершин графа. В случае PPR, $\text{sim}_G(v, \cdot)$ – последняя вершина в одном случайном блуждании начатого с вершины v , в случаях других мер схожести, определяется уравнениями ??, ??	Gradient Descent, Noise Constrictive Estimations. Negative Sampling
GraphSAGE	×	$-\sum_{v_j \in V} \sum_{v_i \in N(v_j)} \log \frac{1}{1+\exp(-\Phi_i \cdot \Phi_j)}$	SGD, Negative Sampling
SDNE	×	$\ (\hat{\mathbf{A}} - \mathbf{A}) \odot \mathbf{B}\ _F^2 + \alpha \sum_{v_i, v_j \in V} a_{i,j} \ \Phi_i - \Phi_j\ _2^2$, где \odot – произведение Адамара, \mathbf{B} –матрицы смещений (biases). $\hat{\mathbf{A}}$ – реконструкция матрицы смежности	SGD, Deep autoencoders
DGI	×	$\frac{1}{N+M} (\sum_{i=1}^N \mathbb{E}_{(X,A)} [\log \mathcal{D}(\Phi_i, s)] + \sum_{j=1}^M \mathbb{E}_{(\tilde{X}, \tilde{A})} [\log(1 - \mathcal{D}(\tilde{\Phi}_j, s))])$ где s – суммарный вектор графа, \mathcal{D} – дискриминатор	Noise Constrictive Estimations
LINE-1	×	$-\sum_{v_i, v_j \in V} a_{ij} \log \frac{1}{1+\exp(-\Phi_i \cdot \Phi_j)}$	ASGD with Negative Sampling
LINE-2	✓	$-\sum_{v_i, v_j \in V} a_{ij} \log \frac{\exp(\Phi_i \cdot \Theta_j)}{\sum_{v_k \in V} \exp(\Phi_i \cdot \Theta_k)}$	ASGD with Negative Sampling

4 Замечания, идеи

Можно считать, что основных общих классов функций потерь 5 штук:

Таблица 3: Общий вид функций потерь

id	Общий вид	Методы
1	$-\sum_{i=1}^n \sum_{j=1}^n c_{i,j} \log \frac{\exp(\Phi_i \cdot \Theta_j)}{\sum_{k=1}^n \exp(\Phi_i \cdot \Theta_k)}$	VERSE, LINE
2	$-\sum_{i=1}^n \sum_{j: v_j \in N(v_i)} c_{i,j} \log \frac{\exp(\Phi_i \cdot \Theta_j)}{\sum_{k=1}^n \exp(\Phi_i \cdot \Theta_k)}$	DeepWalk, Node2Vec, Struc2Vec, APP, GraphSage
3	$\ C - \Phi \cdot \Theta\ ^2$	HOPE, NetMF, Graph Factorization
4	$tr(\Phi^T L \Phi) = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \Phi_i - \Phi_j ^2 a_{ij}$	LaplacianEigenMaps, SDNE-1
5	Ошибки реконструкции (O.P.) $\ \hat{M} - M\ $	Seed (O.P. матрицы фичей $M = X$), SDNE-2 (O.P. матрицы смежности $M = A$)

4.1 Схожесть

1. ф.п. вида 2 можно относительно просто перевести в вид 3. Об этом подробнее в пункте 4.3, в методе NetMF
2. ф.п. вида 1 тоже можно перевести в вид 3, все в том же NetMF. Но с другой стороны, можно попробовать представить ф.п. вида 1 в виде 2. Об этом идет речь в "A Comparative Study for Unsupervised Network Representation Learning". См. пункт 4.6
3. есть ощущение, что ф.п. вида 4 можно представить в виде 3, но что-то пока не получается
4. вообще необходимо ли рассматривать ф.п. вида 5? Там требуется дополнительное действие, по сравнению с предыдущими ф.п. - декодер.
5. матрица S^{PPR} , которая присутствует и HOPE, VERSE, APP означает насколько схожи вершины графа, причем схожесть измеряется по вероятности достичь одной

вершины из другой в ходе случайных блужданий. Но не все так просто, ведь в S^{PPR} , записанной в явном виде, вершина а достигла вершины б в ходе неограниченного по длине блуждания. А в DeepWalk, Node2Vec, блуждания ограничены и по количеству и по длине. Подробнее по PageRank'у написано в пункте 4.4

6. Насколько эквивалентно две ф.п $\| - \Phi \cdot \Theta \|$ и $\sum_{ij} C_{ij} \log(\sigma(\Phi \cdot \Theta))$

4.2 HOPE

В методе HOPE минимизируется $\|C - \Phi \cdot \Theta\|_F^2$. В данном случае элементы матрицы c_{ij} отражают близость вершин v_i, v_j . C можно представить в виде $C = M_g^{-1} M_l$, а матрицы M_g, M_l отличаются для каждой из мер схожести. Рассмотрим:

Rooted PageRank В данном случае $C_{i,j}^{RPR}$ – вероятность того, что случайное блуждание, начавшееся в вершине i в steady state окажется в вершине j . При α – равной вероятности рандомного перехода к соседу:

$$\begin{aligned} M_g &= I - \alpha \cdot P, \\ M_l &= (1 - \alpha) \cdot I, \end{aligned} \tag{1}$$

То есть, чем меньше вероятность того, что i, j окажутся концами одного случайного блуждания в устойчивом состоянии, тем меньше скалярное произведение соответствующих эмбедингов, а значит сами эмбединги отдаляются. И наоборот. Чем больше вероятность того, что i, j окажутся концами одного случайного блуждания в устойчивом состоянии, тем больше скалярное произведение соответствующих эмбедингов, тем ближе друг к другу эти эмбединги.

4.3 ! NetMF!

Каждая из четырех моделей: DeepWalk, LINE, PTE, Node2vec, выполняют неявную матричную факторизацию. Для каждой из обозначенных моделей выведены матричные формы, следующим образом:

1. В изначальной функции потерь делаются несколько замен обозначений
2. Так как функция потерь минимизируется по $\Phi^T \Theta$, то берется производная от \mathcal{L} по $\Phi^T \Theta$ и приравнивается к нулю
3. Ищутся корни. Итого находят выражение для $\Phi^T \Theta$. Обозначают для краткости $\Phi^T \Theta = \log(M)$
4. Но теперь стоит задача найти сами эмбединги, если известно их произведение
5. В явном виде сложно найти, тогда минимизируем $\|\log(M) - \Phi^T \Theta\|$
6. А решение данной минимизации будет SVD на $\log(M)$.
7. $\log(M) = U_d \Sigma_d V_d^T$.

8. Оптимальные значения эмбедингов $\Phi = U_d \sqrt{\Sigma_d}$

Например алгоритм LINE эквивалентен факторизации матрицы (не softmax, а sogmoid + Negative Samling):

$$\log(\text{vol}(G)D^{-1}AD) - \log(b) \quad (2)$$

А алгоритм DeepWalk эквивалентен факторизации следующей матрицы:

$$\log\left(\frac{\text{vol}(G)}{T} \left(\sum_{r=1}^T \mathbf{P}^r\right) \mathbf{D}^{-1}\right) - \log(b) \quad (3)$$

в данном случае b негативных примеров для Negative Sampling, $\text{vol}(G)$ - объем взвешенного графа (сумма степеней всех вершин), T - длина случайного блуждания, r - размер окна в DeepWalk.

4.4 Немного про PageRank

(где упоминается: HOPE, VERSE, APP)

PPR для вершины v .

s - начальное распределение. Имеет длину n (количество вершин). One-hot вектор, где каждый элемент относится к каждой вершине графа, 1 стоит на одном месте - относительно той вершины, которую рассматриваем. Тогда $s \cdot \mathbf{P}$ - вектор длины n , где каждый элемент означает вероятность того, что вершина v перешла в соответствующую номеру индекса элемента вектора вершину.

$s \cdot \mathbf{P}\mathbf{P}$ - означает уже вектор, с вероятностями для вершины v оказаться в соответствующей вершине спустя два прыжка. и т.д. К тому же, если мы введем параметр α - вероятность того, что переход к соседу вообще будет совершен, то соответственно итоговый вектор вероятностей будет является вектором $ppr_v(t)$ и будет записан в виде:

$$ppr_v(t+1) = \alpha \cdot ppr_v(t)\mathbf{P} + (1-\alpha)s \quad (4)$$

Можно рассмотреть попарные схожести и записать уравнение выше в матричном виде. Тогда начально распределение будет единичной матрицей (Т.е., $S^{PPR}(0) = \mathbf{I}$), а уравнение будет в следующем виде:

$$S^{PPR}(t+1) = \alpha \cdot S^{PPR}(t)\mathbf{P} + (1-\alpha)\mathbf{I} \quad (5)$$

Сходится ли? Покажем это:

$$\begin{aligned} \mathbf{S}^{PPR}(t) &= \alpha^2 \cdot \mathbf{S}^{PPR}(t-2)\mathbf{P}^2 + \alpha(1-\alpha)\mathbf{P} + (1-\alpha)\mathbf{I} = \\ &= \alpha^t \mathbf{S}^{PPR}(0)\mathbf{P}^t + (1-\alpha) \sum_{i=0}^{t-1} \alpha^i \mathbf{P}^i \end{aligned} \quad (6)$$

Теперь можем достаточно просто найти предел $\lim_{t \rightarrow \infty} \mathbf{S}^{PPR}(t)$.

Имеем: $0 < \alpha < 1$ и у матрицы \mathbf{P} каждый элемент тоже больше нуля и меньше единицы по построению. Тогда предел первого слагаемого равен нулю, а второе слагаемое - сумма геометрической убывающей прогрессии. Можно применить формулу суммы убывающей геометрической прогрессии. (либо, по-другому, вспомнить формулу $(\mathbf{P}^n - \mathbf{I}) = (\mathbf{P} - \mathbf{I})(\mathbf{P}^{n-1} + \mathbf{P}^{n-2} + \dots)$):

$$\mathbf{S}^{PPR} = \lim_{t \rightarrow \infty} \mathbf{S}^{PPR}(t) = (1 - \alpha)(\mathbf{I} - \alpha\mathbf{P})^{-1} \quad (7)$$

У данной, итоговой предельной матрицы каждый элемент означает вероятность перехода точки i в ходе случайного блуждания с вероятностью рестарта α в точку j . (Именно такая матрица рассмотрена в HOPE и VERSE).

4.5 связь \mathbf{S}^{PPR} с DeepWalk

4.5.1 Идея, высказанная в VERSE

По утверждению авторов статьи VERSE, DeepWalk при стремлении длины случайных блужданий и количества случайных блужданий к бесконечности, делает примерно то же, что и PPR .

Доказательство: в deepWalk от вершины запускается случайное блуждание бесконечной длины, но рассматривается окно длины w . X - случайная величина, означающая насколько далеко блуждание ушло от изначальной точки в пределах окна. Соответственно, распределение вероятностей данной случайной величины:

$$Pr(X = j) = \frac{2}{w(w+1)}(w - j + 1) \quad (8)$$

Оно получено следующим образом: после проведения случайного блуждания, в качестве контекста выбирают все точки, входящие в данное с.б, причем внутри одного случайного блуждания повторяющиеся точки берутся один раз.

Бесконечное случайное блуждание внутри окна w может быть одним из w видов:

1. может так и не выйти дальше чем соседняя вершина и бесконечно крутиться к соседу и обратно. Тогда контекст - 1 вершина.
2. может продвинуться на 2 вершины от нынешней и обратно, так и не выйдя за пределы. Тогда в качестве контекста уже берется две вершины.
3. и т.д.
4. Уйдет на длину w или дальше, тогда все вершины внутри окна w будут выбраны в качестве контекста по одному разу

Отсюда видно, что всего w вариантов случайных блужданий, попавших в окно w . Кроме того, вершина, непосредственно соседствующая с начальной будет выбрана w раз,

вторая $w - 1$ раз и т.д. То есть всего выбрано $\sum_{i=1}^w i = \frac{w(w+1)}{2}$ вершин, а вершина на расстоянии j рёбер будет выбрана $(w - j + 1)$ раз.

Автором статьи утверждается, что распределение $\text{sim}(v, \cdot)$ схоже с распределением выше. Но " $\text{sim}(v, \cdot)$ –биномиальное распределение, а $\text{Pr}(X = j)$ –треугольное, но если взять maximum likelihood estimation для параметра биномиального треугольным распределением, то $\alpha = \frac{w-1}{w+1}$ " (с), из видео самого автора.

(s - вектор со всеми элементами кроме одного равными нулю, а единице равен тот элемент соответствующий вершине, с которой начался старт). Тогда:

$$\text{sim}(v, \cdot) = (1 - \alpha)s \cdot (\mathbf{I} - \alpha\mathbf{P})^{-1} \quad (9)$$

($\text{sim}^{PPR}(v, \cdot)$ Personalized PageRank: один экземпляр $\text{sim}_G(v, \cdot)$ это последняя вершина в одом случайном блуждании, начатом с вершины v . - цитата из статьи)

Я честно пыталась, но не могу связать эти два распределения друг с другом.

4.5.2 Чуть более в лоб или (?) простая марковская цепь (?)

В методе DeepWalk важно количество пар (i, j) (i –целевая, j –контекст) внутри окон всех запущенных случайных блужданий. (Если рассмотрим упрощенный вариант, когда длина случайного блуждания равна длине окна $2w + 1$) то есть, если окно имеет длину w , то вероятность появления пары (i, j) считается как объединение событий: A_1 = случайным блужданием из i в j попали одним шагом $\cup \dots \cup A_w$ = случайным блужданием из i в j попали спустя w шагов $\cup A_{-1}$ = случайным блужданием попали из вершины j в вершину i одним шагом $\cup A_{-w}$ = случайным блужданием из j в i попали спустя w шагов. Тогда, вероятность внутри одного блуждания появления пары (i, j) равна

$$\text{Pr}(i, j) = \text{Pr}(A_{-w} \cup \dots \cup A_{-1} \cup A_1 \cup \dots \cup A_w) = \sum_{r=1}^w (\mathbf{P}^r(i, j) + \mathbf{P}^r(j, i)) \quad (10)$$

То есть, если от вершины i было запущено γ случайных блужданий, то пар (i, j) будет $\gamma \sum_{r=1}^w (\mathbf{P}^r(i, j) + \mathbf{P}^r(j, i))$ штук, или в матричном виде: $\gamma \sum_{r=1}^w (\mathbf{P}^r + \mathbf{P}^{\mathbf{T}^r})$ А т.к. каждый элемент матрицы \mathbf{P} больше нуля и меньше единицы, то применим формулу убывающей геометрической прогрессии:

$$\mathbf{C} = \gamma \sum_{r=1}^w (\mathbf{P}^r + \mathbf{P}^{\mathbf{T}^r}) = \gamma (\mathbf{P}(\mathbf{I} - \mathbf{P}^w)(\mathbf{I} - \mathbf{P})^{-1} + \mathbf{P}^{\mathbf{T}}(\mathbf{I} - \mathbf{P}^{\mathbf{T}^w})(\mathbf{I} - \mathbf{P}^{\mathbf{T}})^{-1}) \quad (11)$$

За \mathbf{C} мы обозначили так называемую котекстную матрицу. Теперь,

$$\sum_{i,j} c_{ij} \log\left(\frac{\exp(\Phi_i \cdot \Theta_j)}{\sum_{k \in V} (\exp(\Phi_i \cdot \Theta_k))}\right) \quad (12)$$

Эквивалентно функции потерь DeepWalk для случая, когда длина окна равна длине случайных блужданий

В Comparative Sudy получают что-то похожее, но как бы нормализованное

4.6 Comparative Sudy. Из вида 1 в вид 2

(Рассматривают случайные блуждания бесконечной длины)

4.6.1 C для DeepWalk

В качестве \mathbf{C} взяли матрицу, которую получили в статье NetMf. То есть утверждается, что матрицу из $\|\mathbf{C} - \Phi \cdot \Theta\|$ можно использовать в функции потерь

$$\sum_{i,j} c_{ij} \log\left(\frac{\exp(\Phi_i \cdot \Theta_j)}{\sum_{k \in V} (\exp(\Phi_i \cdot \Theta_k))}\right) \quad (13)$$

С поправкой на нормализацию \mathbf{C} . А матрица \mathbf{C} получена по алгоритму описанному в 4.3.

Итак,

$$\frac{c_{ij}}{\sum_{u,v|c_{u,v}>0} c_{uv}} \rightarrow \frac{1}{2w} \sum_{r=1}^w \left(\frac{d_i}{\sum_i d_i} \cdot (\mathbf{P}^r)_{(i,j)} + \frac{d_j}{\sum_i d_i} \cdot (\mathbf{P}^r)_{(j,i)} \right) \quad (14)$$

Слагаемое $\frac{d_i}{\sum_i d_i}$ является начальным распределением, означающим вероятность того, что первая вершина в случайном блуждании именно i

4.6.2 C для Node2Vec

Вводится другая матрица переходов, учитывающая "откуда" пришла вершина:

$$\mathbf{P}_{k \rightarrow i \rightarrow j} = \frac{T_{k \rightarrow i \rightarrow j}}{\sum_j T_{k \rightarrow i \rightarrow j}} \quad (15)$$

Что по сути является вероятностью пеерйти от вершины i к вершине j , если вершина i только пришла от вершины k ($T = \frac{1}{p}$ или $\frac{1}{q}$ или 1)

Осталось только получить вероятность оказаться в вершине i , что и обозначают за \mathbf{X} – стационарное распределение случайных блужданий Как поняла я. \mathbf{X}_k – вектор с вероятностями для вершины k перейти в вершины, соответствующие индексам элементов этого вектора.

Тогда всё по аналогии с DeepWalk:

$$\frac{c_{ij}}{\sum_{u,v|c_{u,v}>0} c_{uv}} \rightarrow \frac{1}{2w} \sum_{r=1}^w (\mathbf{X}_{ki} \cdot (\mathbf{P}^r)_{(k,i,j)} + \mathbf{X}_{kj} \cdot (\mathbf{P}^r)_{(j,k,i)}) \quad (16)$$

4.6.3 C для APP,VERSE

Тут очень просто, матрица \mathbf{C} – это и есть S^{PPR}