Функции потерь

1 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 веришин, называемых ребрами
e_{ij}	это реберо между вершиными 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

2 Таблица с функциями потерь

Таблица 2: Методы неконтролируемого обучения

Метод	Контекстный эмбединг	Функция потерь	Подходы к оптимизации, используемые методы
Laplacian EigenMaps	$\times$	$\frac{1}{2}\sum_{i,j}\lvert\Phi_i-\Phi_j\rvert^2 a_{i,j}$	Matrix Factorization
Graph Factorization	$\times$	$\frac{1}{2}\sum_{i,j}(a_{i,j}-\Phi_i\cdot\Phi_j)^2$ $+\frac{\lambda}{2}\sum_i \lvert\lvert\Phi_i\rvert\rvert^2$	SGD with asynchronous optimization (ASGD)
HOPE	$\checkmark$	$\lvert\lvert\mathbf{C}-\Phi\cdot\Theta\rvert\rvert_F^2$, $C-$ может быть записана в разном виде, например для RootedPageRank, см. уравнение 1	Matrix Factorization
NetMF	$\checkmark$	$\lvert\lvert\log(\frac{vol(G)}{bT}(\sum_{r=1}^{T}\mathbf{P}^r)\mathbf{D}^{-1})-\Phi\cdot\Theta\rvert\rvert_F^2$ (значение параметров см. в описании к ур-ию 2)	Matrix Factorization
DeepWalk	$\checkmark$	$-\sum_{v_j\in V}\sum_{v_i\in N(v_j)} log\frac{\exp(\Theta_i\cdot\Phi_j)}{\sum_{v_k\in V}\exp(\Theta_k\cdot\Phi_j)}$, где $N(v)$ – см. таблицу 1. В данном случае случайные блуждания фиксированной длины	SGD, ASGD. Hierarchical Softmax
Node2Vec	$\checkmark$	$-\sum_{v_j\in V}\sum_{v_i\in N(v_j)} log\frac{\exp(\Theta_i\cdot\Phi_j)}{\sum_{v_k\in V}\exp(\Theta_k\cdot\Phi_j)}$ Случайные блуждания имеют два параметра - вероятноность перехода к вершинам, отвечающим за исследования 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. Эмбединги графов !	×	$\|\|\mathbf{X}-\hat{\mathbf{X}}\|\|_2^2$, 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	×	$-\sum_{i,j=1}^{n} sim_G(v_i,v_j)\log\frac{\exp(\Phi_i\cdot\Phi_j)}{\sum_{k=1}^{n}\exp(\Phi_i\cdot\Phi_k)}$ sim_G – распределение схожести вершин графа. В случае PPR, $sim_G(v,\cdot)$ – последняя вершина в одном случайном блуждании начатого с вершины v, в случаях других мер схожести, определяется уравнениями 5, 6	Gradient Descent, Noise Constractive 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	×	$\|\|(\hat{\mathbf{A}}-\mathbf{A})\odot B\|\|_F^2+\alpha\sum_{v_i,v_j\in V} a_{i,j}\|\|\Phi_i-\Phi_j\|\|_2^2$, где $\odot$ – произведение Адамара, B–матрицы смещений (biases). $\hat{\mathbf{A}}$– реконструкция матрицы смежности	SGD, Deep autoencoders
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

3 Замечания, идеи

3.1 HOPE

В методе HOPE минимизируется $||\mathbf{C} - \Phi \cdot \Theta||_F^2$. В данном случае элементы матрицы c_{ij} отражают близость вершин v_i, v_j. C можно представить в виде $C = \mathbf{M}_g^{-1}\mathbf{M}_l$, а матрицы $\mathbf{M}_g, \mathbf{M}_l$ 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:

$$\begin{aligned} \mathbf{M}_g &= \mathbf{I} - \alpha \cdot \mathbf{P}, \\ \mathbf{M}_l &= (1 - \alpha) \cdot \mathbf{I}, \end{aligned} \tag{1}$$

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

(?) Аналогия с функцией потерь в DeepWalk, если убрать softmax normalization (?)

3.2 ! 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 $\Phi^T\Theta = \log(\mathbf{M})$
4. Но теперь стоит задача найти сами эмбеддинги, если известно их произведение
5. В явном виде сложно найти, тогда минимизируем $||\log(\mathbf{M}) - \Phi^T\Theta||$
6. А решение данной минимизации будет приложение SVD к $\log(\mathbf{M})$.
7. $\log(\mathbf{M}) = U_d\Sigma_d V_d^T$.
8. а оптимальные значения эмбедингов $\Phi = U_d\sqrt{\Sigma_d}$

Например, алгоритм DeepWalk эквивалентен факторизации следующей матрицы:

$$\log(\frac{vol(G)}{T}(\sum_{r=1}^{T}\mathbf{P}^r)\mathbf{D}^{-1}) - \log(b) \tag{2}$$

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

А алгоритм LINE эквивалентен факторизации матрицы:

$$\log(vol(G)D^{-1}AD) - \log(b) \tag{3}$$

3.3 VERSE. Просто пояснения к методу, нет важных идей

VERSE [**?**] использует понятия схожести между вершинами и минимизируют расхождение Кульбака - Лейблера (KL) между распределением схожести вершин в графе sim_G и распределением схожести эмбедингов sim_E:

$$\sum_{v\in V} KL(sim_G(v,\cdot)||sim_E(v,\cdot)) \tag{4}$$

Также как и HOPE использует различные меры для распределения схожести в графе (PPR, SimRank, Adjacency similarity):

1. Personalized PageRank: один экзамепляр $sim_G(v,\cdot)$ это последняя вершина в одом случайном блуждании, начатом с вершины v.

2. SimRank: мера структорной взаимосвязи двух вершин, основана на предположении, что схожие вершины связаны с дргуими схожими вершинами. Определяется рекурсивно:
$$sim_G^{SR} = \frac{C}{|I(u)||I(v)|}\sum_{i=1}^{|I(u)|}\sum_{j=1}^{|I(v)|} sim_G^{SR}(I_i(u), I_j(v)) \tag{5}$$
$I(v)$ – множество соседей вершины v, с ребрами, входящими в v, C – число между 0 и 1, геометрически обесценивает важность дальних узлов.

3. Adjacency similarity: если $Out(u)$ – степень выходящих ребер из вершины u, то
$$sim_G^{ADJ}(u,v) = \begin{cases} 1/Out(u) & \text{если}(u,v) \in E \\ 0 & \text{иначе} \end{cases} \tag{6}$$