DynGEM: метод глибокого вкладення динамічних графів

Palash Goyal, Nitin Kamra, Xinran He, Yan Li,

Університет Південної Каліфорнії

Анотація

Вкладення великих графів у простори малої розмірності останнім часом привернуло значний інтерес у зв'язку з широким застосуванням, таким як візуалізація графів, передбачення зв’язків та класифікація вузлів. Існуючі методи зосереджені на обчисленні вкладення статичних графів. Однак на пракиці багато графів є динамічними, стійко розвиваються з часом. Наївне застосування існуючих алгоритмів вкладення кожного зображення динамічних графів окремо зазвичай призводить до незадовільної продуктивності з точки зору стійкості, гнучкості та ефективності. Для вирішення проблеми у цій роботі представлено ефективний алгоритм DynGEM, заснований на останніх досягненнях глибоких автокодувальників у вкладенні графів. Основні переваги DynGEM:

1. вкладення стійкі у часі;
2. може обробляти динамічні графи;
3. має кращий час роботи, ніж при використанні статичних методів вкладення на кожному із зображень динамічного графа.

DynGEM протестовано на виконанні різних задач, включаючи візуалізацію графа, реконструкцію графа, передбачення зв’язків та виявлення аномалій (як на штучних, так і на реальних наборах даних). Результати експерименту демонструють чудову стійкість і масштабованість підходу.

1. Вступ

Багато важливих задач у мережевому аналізі включають створення прогнозів щодо вузлів та/або ребер у графі, а це вимагає ефективних алгоритмів для вилучення значущих ознак та побудови передбачуваних функцій. Серед

багатьох спроб досягнення цієї мети, останнім часом велику увагу привертає вкладення графа, тобто вивчення низьковимірного представлення кожного вузла в графі, яке точно фіксує його зв’язок з іншими вузлами. Продемонстровано, що вкладення графів перевершує альтернативи у багатьох задачах навчання з учителем, таких як класифікація вузлів, передбачення зв’язку та реконструкція графа [Ahmed et al., 2013; Perozzi et al., 2014; Cao et al., 2015; Tang et al., 2015; Grover and Leskovec, 2016; Ou et al., 2016].

Були запропоновані різні підходи до вкладення статичного графа [Goyal and Ferrara, 2017]. Одні приклади включають моделі на основі SVD [Belkin and Niyogi, 2001; Roweis and Saul, 2000; Tenenbaum et al., 2000; Cao et al., 2015; Ou et al., 2016], які розкладають лапласівську або матрицю суміжності високого порядку для створення вкладень вузлів. Інші – моделі на основі випадкових блукань (Random-walk) [Grover and Leskovec, 2016; Perozzi et al., 2014], які створюють вкладення з локалізованих випадкових блукань та багатьох інших [Tang et al., 2015; Ahmed et al., 2013; Cao et al., 2016; Niepert et al., 2016]. Нещодавно Wang та інші розробили інноваційну модель, SDNE, яка використовує глибокий автокодувальник для обробки нелінійності для створення більш точних вкладень [Wang et al., 2016]. Багато інших методів, які обробляють атрибутивні графи та створюють уніфіковане вкладення, також було запропоновано не так давно [Chang et al., 2015; Huang et al., 2017a; Huang et al., 2017b].

Однак на практиці багато графів, таких як соціальні мережі, є динамічними і розвиваються з часом. Наприклад, створюються нові посилання (коли люди знаходять нових друзів), а старі зникають. Більш того, до графа можна додавати нові вузли (наприклад, користувачі можуть приєднуватися до соціальної мережі) і створювати нові посилання на існуючі вузли. Зазвичай динамічні графи представляють як набір зображень графа на різних часових кроках [Leskovec et al., 2007].

Існуючі роботи, що зосереджуються на динамічних вкладеннях, часто застосовують алгоритми статичного вкладення до кожного зображення динамічного графа, а потім вирівнюють отримані статичні вкладення за часовими кроками [Hamilton et al., 2016; Kulkarni et al., 2015]. Наївне застосування існуючих алгоритмів статичного вкладення кожного зображення окремо призводить до низької продуктивності через наступні проблеми:

- **Стійкість**: вкладення, створене статичними методами, не є стійким, тобто вкладення графів на послідовних часових кроках може істотно відрізнятися, навіть якщо самі графи не дуже змінюються.
- **Зростання графів**: нові вузли можна додавати до графа і створювати нові зв'язки з вже існуючими вузлами в міру того, як динамічний граф зростає з часом. Усі існуючі підходи передбачають фіксовану кількість вузлів при навчанні вкладень графів, тож, не можуть бути застосовані для обробки зростаючих графів.
- **Масштабованість**: навчання вкладень для кожного зображення окремо призводить до лінійного часу виконання за кількістю зображень. Оскільки навчання одного вкладення вимагає багато обчислень, наївний підхід не масштабується до динамічних мереж з великою кількістю зображень.

Були також і спроби навчити вкладення динамічних графів шляхом накладання тимчасового регуляризатора аби забезпечити тимчасову гладкість над вкладеннями послідовних зображень [Zhu et al., 2016]. Такий спосіб не підходить для динамічних графів, де послідовні часові кроки можуть значно відрізнятися, а отже не може використовуватися в таких задачах, як виявлення аномалій. Більш того, цей підхід – модель на основі факторизації графа (надалі скорочено як GF) [Ahmed et al., 2013], DynGEM перевершує ці моделі, як показано в експериментах у розділі 5. [Dai et al., 2017] вивчали вкладення динамічних графів, проте вони зосереджувалися на двочасткових графах (bipartite graphs) спеціально для взаємодії користувача з елементом.

У цій статті розроблено ефективний алгоритм вкладення графів, так званий DynGEM, для створення стійких вкладень динамічних графів. DynGEM має в основі глибокий автокодувальник та використовує останні досягнення глибокого навчання для створення нелінійних вкладень. Замість того, щоб вивчати вкладення кожного зображення з початку, DynGEM послідовно створює вкладення зображення в момент часу t із вкладення зображеня в момент часу t - 1. Зокрема, ініціалізується вкладення з попереднього часового кроку, а потім виконується градієнтне навчання. Такий підхід не тільки забезпечує стійкість вкладення в часі, але й призводить до ефективного навчання, оскільки для всіх вкладень після першого часового кроку потрібно небагато ітерацій для наближення. Для обробки динамічних графів зі зростаючою кількістю вузлів, поступово збільшується розмір нейронної мережі за допомогою евристичного PropSize для динамічного визначення кількості прихованих елементів, необхідних для кожного зображення. Як додаток до запропонованої моделі вводяться суворі показники стійкості для динамічних вкладень графів.

Як на штучних, так і на реальних наборах даних, результати експерименту демонструють, що запропонований підхід забезпечує подібну чи навіть кращу точність реконструкції графа та робить прогнозування зв’язків ефективнішим, ніж існуючі статичні підходи. DynGEM також застосовний до задач динамічної візуалізації графів і виявлення аномалій, які є неможливими для багатьох попередніх підходів статичного вкладення.

2. Визначення та початкові відомості

Позначимо зважений граф як G (V, E), де V – множина вершин, а E – множина ребер. Зважену матрицю суміжності G позначено через S. Якщо $(u, v) \in E$, то $s_{ij} > 0$ позначає вагу ребра (u, v); інакше $s_{ij} = 0$. Через $s_i = [s_{i,1}, \ldots, s_{i,|V|}]$ позначено і-тий рядок матриці суміжності.

Для графа $G = (V, E)$, вбудований граф є відображенням $f: V \rightarrow R^d$, а саме $y_v = f(v) \; \forall \; v \in V$. Необхідно, щоб $d \ll |V|$ а функція f зберігала деяку міру близькості, визначену на графі G. Інтуїтивно, якщо два вузли u і v «подібні» в графі G, їх вкладення y_u та y_v повинні бути близькими один до одного в просторі вкладень. Позначення $f(G) \in R^{|V| \times d}$ використовується для матриці вкладень усіх вузлів у графі G.

У цій роботі розглянуто проблему динамічного вкладення графів. Динамічний граф G представлений у вигляді серії зображень, тобто $G = \{G_1, \ldots, G_T\}$, де $G_t = (V_t, E_t)$, T – число зображень. Враховано зростаючі графи, тобто $V_t \subseteq V_{t+1}$, нові вузли можуть приєднуватися до динамічного графа і створювати посилання на існуючі вузли. Видалені вузли розглянуто як частину графа з нульовими вагами до решти вузлів. Між E_t і E_{t+1} немає зв'язку і між зображеннями можуть утворюватися нові ребра у той час як наявні ребра можуть зникати.

Динамічне вкладення графів розширює концепцію вкладення в динамічні графи. Дано динамічний граф $G = \{G_1, \ldots, G_T\}$, динамічне вкладення графа є часовою послідовністю відображень $F = \{f_1, \ldots, f_T\}$ так, що відображення f_t є вкладенням графа G_t і всі відображення зберігають міру близькості для відповідних графів.

Успішний алгоритм вкладення динамічного графа повинен створювати стійкі вкладення з часом. Інтуїтивно, стійке динамічне вкладення – те, в якому послідовні вкладення відрізняються лише на невелику величину, якщо основні графи змінюються, тобто якщо G_{t+1} не дуже відрізняється від G_t, вкладення виводить $Y_{t+1} = f_{t+1}(G_{t+1})$ і $Y_t = f_t(G_t)$ також змінюється на незначну величину.

Якщо казати більш конкретно, нехай $S_t(\tilde{V})$ – зважена матриця суміжності індукованого підграфу множини вузлів $\tilde{V} \subseteq V_t$, а $F_t(\tilde{V}) \in R^{|\tilde{V}| \times d}$ – вкладення всіх вузлів у $\tilde{V} \subseteq V_t$ миттєвого зображення t. Абсолютну стійкість ми визначаємо як:

$$\mathcal{S}_{abs}(\mathcal{F};t) = \frac{\|F_{t+1}(V_t) - F_t(V_t)\|_F}{\|S_{t+1}(V_t) - S_t(V_t)\|_F}.$$

Іншими словами, абсолютна стійкість будь-якого вкладення F є відношенням різниці вкладень до різниці матриць суміжності. Оскільки це визначення стійкості залежить від розмірів задіяних матриць, визначено іншу міру, яка називається відносною стійкістю і є інваріантною до розміру матриць суміжності та вкладення:

$$\mathcal{S}_{rel}(\mathcal{F};t) = \frac{\|F_{t+1}(V_t) - F_t(V_t)\|_F}{\|F_t(V_t)\|_F} \Big/ \frac{\|S_{t+1}(V_t) - S_t(V_t)\|_F}{\|S_t(V_t)\|_F}.$$

Далі визначено константу стійкості:

$$K_{\mathcal{S}}(\mathcal{F}) = \max_{\tau,\tau'} |\mathcal{S}_{rel}(F;\tau) - \mathcal{S}_{rel}(F;\tau')|.$$

Динамічне вкладення F є стійким, якщо воно має невелику константу стійкості. Очевидно, що чим менше $K_S(F)$, тим стійкіше вкладення F. У експериментах використано константу стійкості як метрику для порівняння стійкості алгоритму DynGEM з іншими алгоритмами.

3. DynGEM: модель вкладення динамічного графа

Останні досягнення в глибокому навчанні без учителя показали, що автокодувальники можуть успішно вивчати дуже складні низьковимірні представлення даних для виконання різних задач [Bengio et al., 2013]. DynGEM використовує глибокий автокодувальник для відображення вхідних даних у нелінійний латентний простір, щоб фіксувати тенденції підключення у зображенні графа на будь-якому часовому кроці. Модель є напівконтрольованою та мінімізує комбінацію двох цільових функцій, що відповідають наближенню першого та другого порядків відповідно. Модель автокодувальника зображена на рисунку 1, а умовні позначення наведені в таблиці 1 (як у [Wang et al., 2016]). Символи з кришкою призначені для декодувальника.

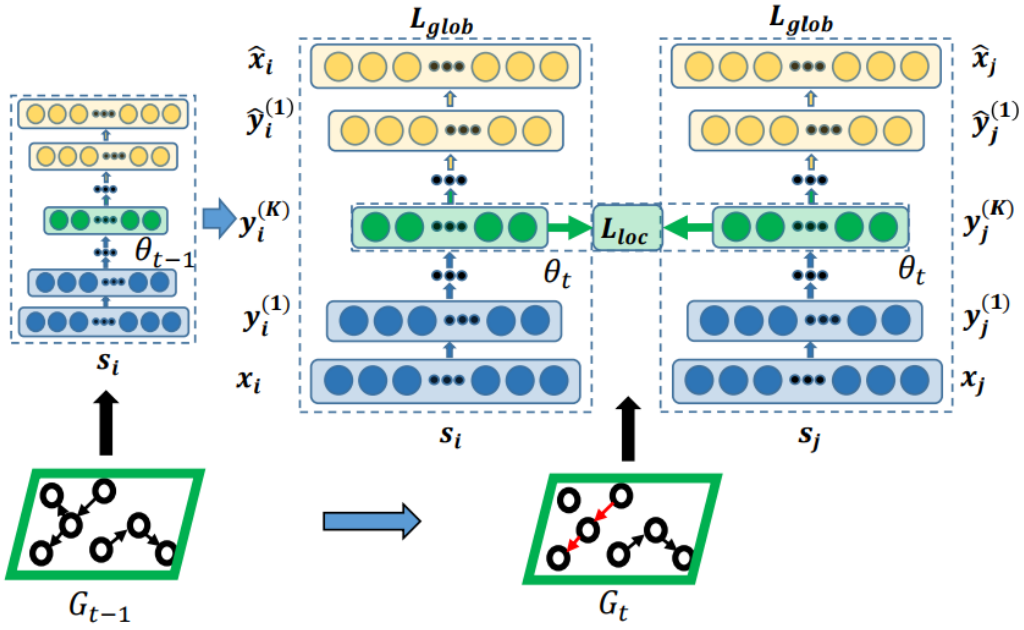


Рисунок 1. DynGEM: модель вкладення динамічного графа. На малюнку показано два зображення динамічного графа та глибокий автокодувальника на кожному кроці.

Таблиця 1. Умовні позначення для глибокого автокодувальника

Symbol	Definition
n	number of vertices
K	number of layers
$S = \{\boldsymbol{s_1}, \cdots, \boldsymbol{s_n}\}$	adjacency matrix of G
$X = \{\boldsymbol{x_i}\}_{i\in[n]}$	input data
$\hat{X} = \{\hat{\boldsymbol{x}}_{\boldsymbol{i}}\}_{i\in[n]}$	reconstructed data
$Y^{(k)} = \{\boldsymbol{y_i^{(k)}}\}_{i\in[n]}$	hidden layers
$Y = Y^{(K)}$	embedding
$\boldsymbol{\theta} = \{W^{(k)}, \hat{W}^{(k)}, \boldsymbol{b^{(k)}}, \hat{\boldsymbol{b}}^{(\boldsymbol{k})}\}$	weights, biases

Околиця вузла v_i задається як $s_i \in R^n$. Для будь-якої пари вузлів v_i і v_j графа G_t модель бере їх околиці як вхідні дані: $x_i = s_i$, $x_j = s_j$ і передає їх через автокодувальник для генерування d-вимірних вкладень $y_i = y_i^{(K)}$ і $y_j = y_j^{(K)}$ на виходах кодувальника. Декодувальник відновлює околиці $\hat{x}_i$ та $\hat{x}_j$ із вкладень y_i та y_j.

3.1. Обробка зростаючих графів

Обробка динамічних графів зростаючих розмірів вимагає наявність механізму розширення моделі автокодувальника зі збереженням ваг з попередніх часових кроків навчання. Ключовим моментом є рішення про те, як кількість прихованих шарів і кількість прихованих елементів має зростати, коли до графа додається більше вузлів. Ми пропонуємо евристичний PropSize для обчислення нових розмірів усіх шарів, яка гарантує, що розміри двох послідовних шарів знаходяться в межах певного коефіцієнта один до одного.

PropSize: пропонується для обчислення нових розмірів шарів нейронної мережі на кожному часовому кроці та додавання, за необхідності, нових шарів. Для кодувальника ширина шарів обчислюється для кожної пари послідовних шарів (l_k і l_{k+1}), починаючи з вхідного шару ($l_1 = x$) і першого прихованого шару ($l_2 = y^{(1)}$), доки не буде виконано наступну умову для кожної послідовної пари шарів: $size(l_{k+1}) \geq \rho \times size(l_k)$, де $0 < \rho < 1$ підібраний гіперпараметр. Якщо умова не виконується для будь-якої пари (l_k, l_{k+1}), ширина шару для l_{k+1} збільшується для задоволення евристики. Варто зауважити, що розмір вкладеного шару $y = y^{(K)}$ завжди залишається фіксованим на d і ніколи не розширюється. Якщо правило PropSize не виконується на передостанньому та вкладеному шарі кодувальника, між ними додається більше шарів (з розмірами, що задовольняють правило) доки правило не буде виконуватися. Ця процедура також застосовується до шарів декодувальника, починаючи з вихідного шару ($\hat{x}$) і продовжуючи всередину до шару вкладення (або $\hat{y} = \hat{y}^{(K)}$) для обчислення нових розмірів шару.

Визначивши кількість шарів і кількість прихованих елементів у кожному шарі, використовуємо підходи Net2WiderNet і Net2DeeperNet з [Chen et al., 2015] для розширення глибокого автокодувальника. Net2WiderNet дозволяє розширювати шари, тобто додавати більше прихованих елементів до існуючого шару нейронної мережі, зберігаючи при цьому функцію, що обчислюється цим

шаром. Net2DeeperNet додає новий шар між двома існуючими шарами, змушуючи його точно повторювати відображення тотожності. Це можна зробити для функції активації ReLU, але не для сигмоїдної.

Комбінація розширення та поглиблення автокодувальника з PropSize, Net2WiderNet і Net2DeeperNet на кожному часовому кроці дозволяє працювати з динамічними графами зі зростаючою кількістю вузлів в часі й демонструє чудову продуктивність, що і показали експерименти.

3.2. Функція втрат та навчання

Щоб дізнатися параметри моделі, зважена комбінація трьох компонентів мінімізується на кожному часовому кроці: $L_{net} = L_{glob} + \alpha L_{loc} + \nu_1 L_1 + \nu_2 L_2$, де α, ν_1 та ν_2 – гіперпараметри, обрані як відносні ваги цільових функцій.

$L_{loc} = \sum_{i,j}^{n} s_{ij} \|\boldsymbol{y_i} - \boldsymbol{y_j}\|_2^2$ – близькість першого порядку, яка відповідає локальній структурі графа.

$L_{glob} = \sum_{i=1}^{n} \|(\hat{\boldsymbol{x}}_i - \boldsymbol{x_i}) \odot \boldsymbol{b_i}\|_2^2 = \|(\hat{X} - X) \odot B\|_F^2$ – близькість другого порядку, яка відповідає глобальному сусідству кожного вузла в графі і зберігається шляхом неконтрольованої реконструкції околиці кожного вузла. b_i є вектором з $b_{ij} = 1$ при $s_{ij} = 0$, або $b_{ij} = \beta$ у випадку $s_{ij} > 1$. Регуляризатори $L_1 = \sum_{k=1}^{K} \left(\|W^{(k)}\|_1 + \|\hat{W}^{(k)}\|_1\right)$ $L_2 = \sum_{k=1}^{K} \left(\|W^{(k)}\|_F^2 + \|\hat{W}^{(k)}\|_F^2\right)$ додаються, щоб стимулювати розрідженість ваг мережі та запобігти перенавчанню моделі відповідно до структури графа. DynGEM вивчає параметри θ_t глибокого автокодувальника на кожному часовому кроці t і використовує $Y_t^{(K)}$ як вихід вкладення графа G_t.

3.3. Стійкість за рахунок повторного використання попереднього кроку вкладення

Для динамічного графа $G = \{G_1, \ldots, G_T\}$ навчаємо модель глибокого автокодувальника на G_1, використовуючи випадкову ініціалізацію параметрів θ_1.

Для всіх наступних часових кроків ініціалізуємо модель з параметрами θ_t параметрами попереднього часового кроку θ_{t-1}, перш ніж розширювати чи поглиблювати модель. Це призводить до прямої передачі знань про структуру від f_{t-1} до f_t, тому моделі потрібно лише дізнатися про зміни між G_{t-1} і G_t. Навчання дуже швидко збігається за кілька ітерацій для часових кроків: {2, …, T}. Ще важливішим є те, що він гарантує стійкість, забезпечуючи, що вкладення Y_t залишається близьким до Y_{t-1}. Варто зазначити, що на відміну від [Zhu et al., 2016] не використовуються явні регуляризатори аби зберегти вкладення на часових кроках t − 1 і t близькими. Якщо зображення графа в моменти t − 1 і t істотно відрізняються, то відповідні вкладення f_{t-1} і f_t також мають бути відмінні. Результати в розділі 5 показують кращу стійкість і швидший час роботи запропонованого методу порівняно з іншими алгоритмами.

3.4. Техніки масштабованості

Попередні моделі глибокого автокодувальника [Wang et al., 2016] для вкладення статичних графів використовують сигмоїдну функцію активації і навчаються за допомогою стохастичного градієнтного спуску (SGD).

У запропонованій моделі використовується ReLU у всіх шарах автокодувальника для підтримки зважених графів, оскільки ReLU будує позитивні входи s_i. Вона також прискорює навчання, так як похідну ReLU легко обчислити, а похідна сигмоїдної функції активації вимагає обчислення експоненти [Glorot et al., 2011]. Насамкінець, ReLU дозволяє градієнтам L_{loc} і L_{glob} ефективно поширюватися через кодувальник і запобігає ефекту зникнення градієнта [Glorot et al., 2011], який спостерігається для сигмоїдної функції активації.

Також використовується імпульс Нестерова [Sutskever et al., 2013] з налаштованими гіперпараметрами, який сходиться набагато швидше, ніж SGD.

Насамкінець, можна спостерігати кращу продуктивність для всіх задач з комбінацією регуляризаторів L1- та L2-норми.

Псевдокод навчання моделі DynGEM для одного зображення динамічного графа наведений в алгоритмі 1. Його можна викликати багаторазово на кожному зображенні динамічного графа для створення динамічного вкладення.

Algorithm 1: Algorithm: DynGEM

Input: Graphs $G_t = (V_t, E_t), G_{t-1} = (V_{t-1}, E_{t-1})$, Embedding Y_{t-1}, autoencoder weights $\boldsymbol{\theta}_{t-1}$
Output: Embedding Y_t
From G_t, generate adjacency matrix S, penalty matrix B
Create the autoencoder model with initial architecture
Initialize $\boldsymbol{\theta}_t$ randomly **if** $t = 1$, **else** $\boldsymbol{\theta_t} = \boldsymbol{\theta}_{t-1}$
if $|V_t| > |V_{t-1}|$ **then**
    Compute new layer sizes with PropSize heuristic
    Expand autoencoder layers and/or insert new layers
end if
Create set $\mathcal{S} = \{(s_i, s_j)\}$ for each $e = (v_i, v_j) \in E_t$
for $i = 1, 2, \ldots$ **do**
    Sample a minibatch M from $\mathcal{S}$
    Compute gradients $\nabla_{\boldsymbol{\theta}_t} L_{net}$ of objective L_{net} on M
    Do gradient update on $\boldsymbol{\theta}_t$ with nesterov momentum
end for

4. Експерименти

4.1. Набори даних

Продуктивність DynGEM оцінюється як на штучних, так і на реальних динамічних графах. Набори даних наведено в таблиці 2.

Таблиця 2. Набори даних. |V|, |E| і T позначають кількість вузлів, ребер і довжину часових рядів відповідно.

	$\|V\|$	$\|E\|$	T
SYN	1,000	79,800-79,910	40
HEP-TH	1,424-7,980	2,556-21,036	60
AS	7716	10,695-26,467	100
ENRON	184	63-591	128

Штучні дані (SYN): штучні динамічні графи створюються за допомогою стохастичної блочної моделі [Wang and Wong, 1987]. Перше зображення

динамічного графа створюється, щоб отримати три спільноти однакового розміру з імовірністю всередині блоку 0,2 і ймовірністю перехресного блоку 0,01. Для створення наступних графів випадковим чином обираються вузли на кожному часовому кроці та переміщуються до іншої спільноти. SYN використовується для візуалізації змін у вкладеннях, коли вузли змінюють спільноти.

HEP-TH [Gehrke et al., 2003]: Оригінальний набір даних містить тези доповідей на конференції High Energy Physics Theory в період з січня 1993 року до квітня 2003 року. Для кожного місяця створено мережу співпраці, з використанням всіх документів, опублікованих до цього місяця. Дані беруться за перші п’ять років і генерується часовий ряд, що містить 60 графів зі збільшенням кількості вузлів від 1424 до 7980.

Автономні системи (AS) [Leskovec and Krevl, 2014]: це комунікаційна мережа «who-talks-to-whom» із BGP (Border Gateway Protocol). Набір даних містить 733 екземпляри з 8 листопада 1997 року по 2 січня 2000 року. Розглядається підмножина цього набору даних: перші 100 зображень.

ENRON [Klimt and Yang, 2004]: набір даних містить електронні листи між співробітниками Enron Inc. з січня 1999 року по липень 2002 року. Граф обробляється як і в [Park et al., 2009], обираються листи лише між топ-менеджерами кожного тижня, починаючи з січня 1999 року.

4.2. Алгоритми та метрики оцінювання

Порівнюється продуктивність наступних алгоритмів динамічного вкладення для кількох задач:

- SDNE^2: застосовується незалежно до кожного зображення динамічної мережі.
- $[SDNE/GF]_{align}$: застосовується алгоритм SDNE або GF до кожного зображення окремо і для вирівнювання повертається вкладення, як у [Hamilton et al., 2016].

- GF_{init}: застосовується алгоритм GF, вкладення якого в момент часу t ініціалізується вкладенням в момент $t - 1$.
- DynGEM: запропонований алгоритм для динамічного вкладення графів. Розмір вкладення d = 20 для ENRON і 100 для всіх інших наборів даних. Використовуються два прихованих шари в глибокому автокодувальнику з початковими розмірами (згодом вони можуть розширюватися) для кожного набору даних: ENRON = [100, 80], {HEP-TH, AS, SYN} = [500, 300]. Структури нейронної мережі обираються шляхом неформального пошуку за набором архітектур. Встановлено $\rho = 0{,}3$, спад розміру кроку для SGD = 10^{-5}, момент – 0,99. Інші параметри встановлюються за допомогою пошуку сітки з відповідною перехресною перевіркою наступним чином: $\alpha \in [10^{-6}, 10^{-5}]$, $\beta \in [2, 5]$ і $\nu_1 \in [10^{-4}, 10^{-6}]$ і $\nu_2 \in [10^{-3}, 10^{-6}]$.

У експериментах оцінено продуктивність вищевказаних моделей у задачах реконструкції графа, передбачення зв'язків, стійкості вкладення та виявлення аномалій. Для перших двох задач: реконструкції графа та передбачення зв'язків, використано Mean Average Precision (MAP) як показник (див. [Wang et al., 2016]). Щоб оцінити стійкість динамічного вкладення, використано константу стійкості $K_S(F)$, визначену в розділі 2. Усі експерименти проведено на системі Ubuntu 14.04.4 LTS з 32 ядрами, 128 ГБ ОЗУ та тактовою частотою 2,6 ГГц. Використано графічний процесор Nvidia Tesla K40C.

5. Результати та аналіз

5.1. Відновлення графа

Очікується, що вкладення точно відновлять граф. Ребра графа між парами вершин відновляться із вкладень, використовуючи декодувальник зі запропонованої моделі автокодувальника. Ранжируємо пари вершин відповідно до їх реконструйованої близькості. Потім обчислюється відношення реальних зв'язків у верхніх k парах вершин як точність реконструкції. Показник MAP

реконструкції графа, усереднений за зображеннями наборів даних, наведено в таблиці 3.

Таблиця 3: Середнє MAP у реконструкції графа

	SYN	HEP-TH	AS	ENRON
GF_{align}	0.119	0.49	0.164	0.223
GF_{init}	0.126	**0.52**	0.164	0.31
$SDNE_{align}$	0.124	0.04	0.07	0.141
SDNE	**0.987**	0.51	0.214	0.38
DynGEM	**0.987**	0.491	**0.216**	**0.424**

Результати показують, що DynGEM перевершує всі алгоритми на основі факторизації графа, за винятком HEP-TH.

5.2. Передбачення зв'язків

Іншим важливим застосуванням вкладення графів є передбачення зв'язків, перевіряється, наскільки добре модель може передбачати неспостережувані ребра. Мережа повинна бути здатна не тільки відновлювати ребра, видимі їй під час навчання, але і передбачити ребра, які відсутні в навчальних даних. Щоб перевірити це, випадковим чином приховується 15% ребер мережі в момент часу t ($G_{thidden}$). Тренується динамічне вкладення з використанням зображень $\{G_1, \ldots, G_{t-1}, G \backslash G_{thidden}\}$ і прогнозуються приховані ребра на знімку в момент t. Прогнозуються найбільш імовірні (найбільш зважені) ребра, які не входять до спостережуваного набору ребер як приховані. Потім передбачення порівнюються з $G_{thidden}$, щоб отримати оцінку точності. Точність передбачень, усереднена за t від 1 до T, наведено в таблиці 4.

Таблия 4. Середнє MAP у прогнозуванні зв'язків

	SYN	HEP-TH	AS	ENRON
GF_{align}	0.027	0.04	0.09	0.021
GF_{init}	0.024	0.042	0.08	0.017
$SDNE_{align}$	0.031	0.17	0.1	0.06
SDNE	0.034	0.1	0.09	0.081
DynGEM	**0.194**	**0.26**	**0.21**	**0.084**

Можна помітити, що DynGEM здатний передбачити відсутні ребра краще, ніж решта алгоритмів для всіх наборів даних. Оскільки підходи на основі факторизації графів працюють гірше, ніж підхід на основі SDNE і запропонований алгоритм DynGEM, для решти задач порівнюються лише алгоритми на основі DynGEM та SDNE.

5.3. Стійкість методів вкладення

Стійкість вкладення має вирішальне значення для такої задачі, як виявлення аномалій. Оцінюється стійкість запропонованої моделі у порівнянні з іншими методами на чотирьох наборах даних (таблиця 5).

Таблиця 5. Константи стійкості $K_S(F)$ вкладень в динамічні графи

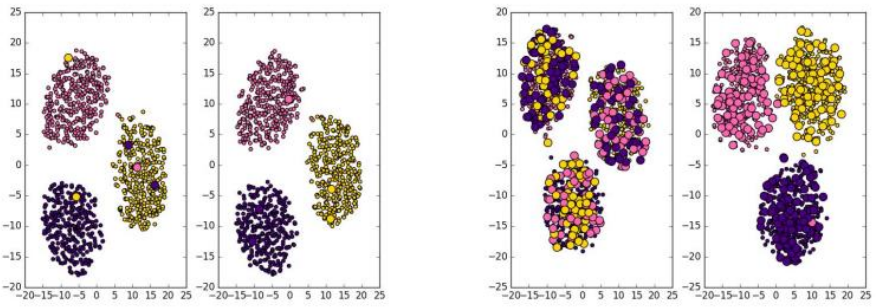
	SYN	HEP-TH	AS	ENRON
SDNE	0.18	14.715	6.25	19.722
$SDNE_{align}$	0.11	8.516	2.269	18.941
DynGEM	**0.008**	**1.469**	**0.125**	**1.279**

Запропонована модель значно перевершує інші та забезпечує стійкі вкладення із кращою продуктивністю відновлення графа (див. таблицю 3 у розділі 5.1 щодо помилок реконструкції за цієї стійкості). У наступному розділі продемонстровано, що цю стійкість можна використовувати для візуалізації та виявлення аномалій у реальних динамічних мережах.

5.4. Візуалізація

Одним із важливих застосувань вкладення графів є візуалізація графа. Експерименти проводяться з набором даних SYN з відомою структурою спільноти. Застосовується t-SNE [Maaten and Hinton, 2008] до вкладення, створеного DynGEM на кожному часовому кроці, щоб побудувати 2D-вкладення. Щоб уникнути нестабільності візуалізації протягом часових кроків, t-SNE ініціалізується ідентичним випадковим станом для всіх часових кроків. Рисунок

2 ілюструє результати двовимірної візуалізації 100-вимірного вкладення для набору даних SYN, коли вузли змінюють спільноти за один часовий крок.



(a) DynGEM time step with 5 nodes jumping out of 1000

(b) DynGEM time step with 300 nodes jumping out of 1000

Рисунок 2. 2D-візуалізація 100-вимірних вставок для набору даних SYN.

Ліва (права) діаграма на кожному підрисунку показує вкладення до (після), коли вузли змінять свої спільноти. Точка на будь-якому графіку представляє вкладення вузла в граф, колір якого вказує на спільноту вузлів. Малі (великі) точки є вузлами, які не змінили спільноту. Кожна велика точка забарвлюється відповідно до кольору спільноти. Можна спостерігати, що DynGEM вкладення вузлів, які змінили спільноту, точно слідують змінам у структурі спільноти, не порушуючи вкладення інших вузлів, навіть якщо частка таких вузлів дуже висока (див. рисунок 2b, де 30% вузлів змінюють спільноту). Це яскраво демонструє стійксть запропонованого алгоритму.

5.5. Застосування для виявлення аномалій

Виявлення аномалій є важливою задачею для виявлення шкідливої діяльності в мережах. DynGEM застосовано до набору даних Enron, щоб виявити аномалії та порівняти отримані результати із подіями, що відбуваються з компанією за спостереженнями [Sun et al., 2007].

Δt визначається як зміна вкладення між часовими кроками t і t-1: $\Delta t = ||F_{t+1}(V_t) - F_t(V_t)||_F$, цю величину можна встановити для виявлення аномалій. Графік Δt з часом у наборі даних Enron зображений на рисунку 3.

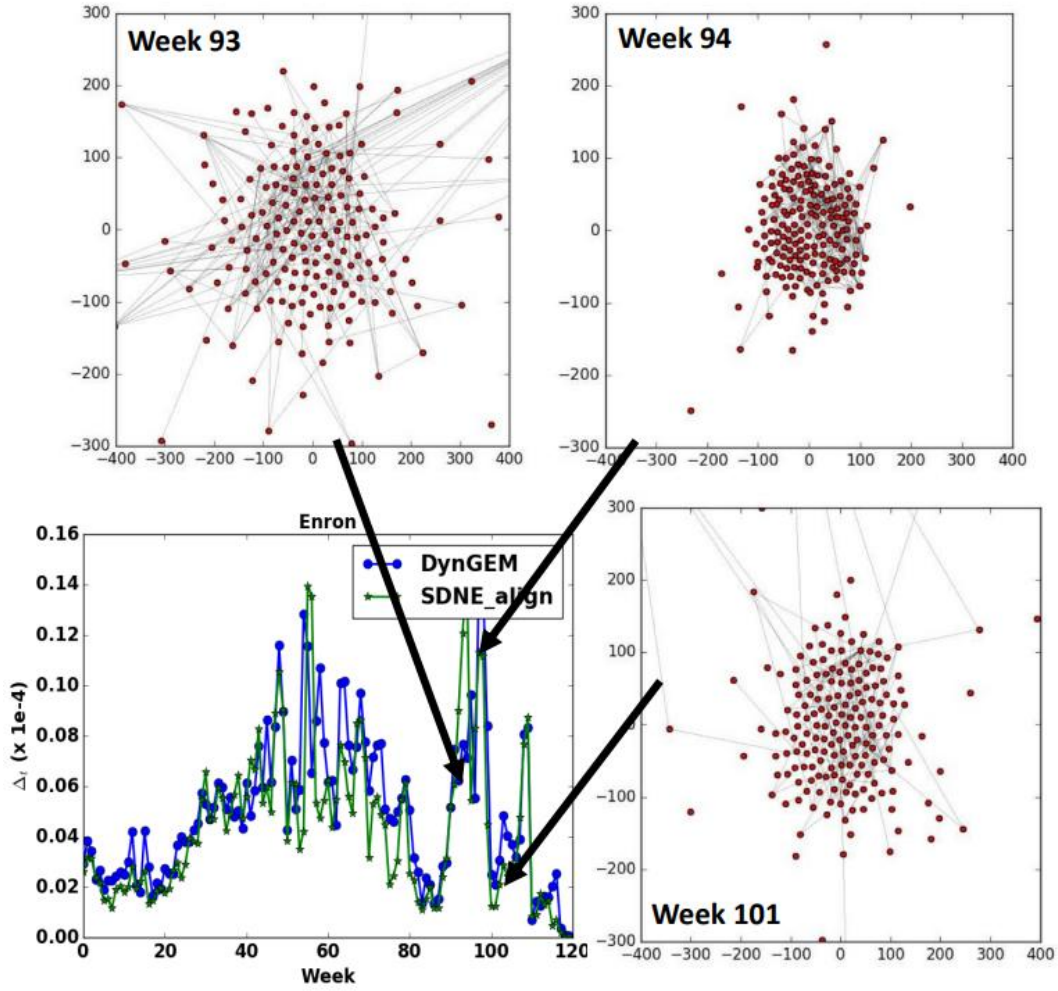


Рисунок 3. Результати виявлення аномалій на Enron та візуалізація вкладень за 93, 94 та 101 тижні з набору даних Enron

На рисунку бачимо три значні стрибки приблизно в 45, 55 та 94 тижні, що відповідають лютому 2001, червню 2001 та січню 2002 року. Ці місяці були пов’язані з наступними подіями: Джеффрі Скілінг став генеральним директором у лютому 2001 року; Роув продав акції на енергію в червні 2001 року, а у січні 2002 року ФБР розпочало відставку генерального директора та розслідування злочинів. Також спостерігаємо деякі піки, що вказують на початок цих подій. На рисунку 3 зображені візуалізації вкладень на 94 тижні. Розширення вкладень можна спостерігати протягом 93 і 101 тижнів, що відповідає низькій комунікації між співробітниками. І навпаки, обсяг комунікації значно зріс на 94-му тижні (показано дуже компактним вкладенням).

5.6. Ефект розширення шару

Оцінимо вплив розширення шару на набір даних HEP-TH. Для цього запускаємо модель DynGEM з розширенням шарів і без нього. Можна помітити, що без розширення шару модель досягає середнього значення MAP 0,46 і 0,19 для реконструкції графа та передбачення зв'язку відповідно. Варто зауважити, що це значно нижче, ніж значення продуктивності DynGEM з розширенням шарів 0,491 і 0,26 для відповідних задач. Також слід зазначити, що для SDNE та $SDNE_{align}$ обираємо найкращу модель на кожному часовому кроці. З використанням евристичного PropSize ця потреба усувається, а розмір нейронної мережі для наступних часових кроків підбирається автоматично.

5.7. Масштабованість

Тепер порівняймо час, необхідний для вивчення різних вбудованих моделей. З таблиці 6 ми бачимо, що DynGEM значно швидший за $SDNE_{align}$. Ми не порівнюємо його з методами, заснованими на факторизації графів, тому що, незважаючи на їх швидкість, моделі на основі глибокого автокодувальника значно перевершують їх.

Таблиця 6. Час обчислення для методів вкладення на перших сорока часових кроках по кожному набору даних. $Speedup_{exp}$ – очікуване прискорення на основі параметрів моделі

	SYN	HEP-TH	AS	ENRON
$SDNE_{align}$	56.6 min	71.4 min	210 min	7.69 min
DynGEM	13.8 min	25.4 min	80.2 min	3.48 min
Speedup	4.1	2.81	2.62	2.21
$Speedup_{exp}$	4.8	3	3	3

Припускаючи, що n_s ітерацій для вивчення вкладення одного зображення з нуля та n_i ітерацій для вивчення вкладення під час ініціалізації на попередньому кроці вкладення, очікуване прискорення для динамічного графа довжиною T

визначається як: $\frac{Tn_s}{n_s+(T-1)n_i}$ (ігноруючи інші накладні обчислення). Порівняймо спостережене прискорення з очікуваним. У таблиці 7 показано, що запропонована модель досягає прискорення, ближчого до очікуваного, так як кількість зображень графа збільшується через зменшення ефекту накладних обчислень (наприклад, збереження, завантаження, розширення та ініціалізація моделі).

Таблиця 7: Час обчислення для методів вкладення на SYN при зміні тривалості часового ряду T

	T=5	T=10	T=20	T=40
$SDNE_{align}$	6.99 min	14.24 min	26.6 min	56.6 min
DynGEM	2.63 min	4.3 min	7.21 min	13.8 min
Speedup	2.66	3.31	3.69	4.1

Результати експерименту показують, що DynGEM забезпечує стабільну швидкість в різних мережах.

6. Висновок

У цій статті запропоновано DynGEM, швидкий та ефективний алгоритм для побудови стійких вкладень динамічних графів. Він використовує глибокий автокодувальник, що динамічно розширюється аби фіксувати нелінійні наближення першого та другого порядків вузлів графа. Крім того, запропонована модель використовує інформацію з попередніх часових кроків для того, щоб прискорити процес навчання, поступово навчаючи вкладення на кожному часовому кроці. Проведені експерименти демонструють стійкість алгоритму в часі та доводять, що він зберігає конкурентоспроможність у всіх задачах, наприклад, реконструкції графа, передбачення зв’язків та візуалізації графа. Показано, що DynGEM точно зберігає структуру спільноти, навіть коли велика частка вузлів (~ 30%) змінює спільноту на часових кроках. Також алгоритм застосовано для виявлення аномалій, що є новим застосуванням динамічного

вкладення графів. DynGEM демонструє великий потенціал в багатьох інших задачах виведення графів, таких як класифікація вузлів, кластеризація тощо, які залишено на майбутнє.

Є кілька напрямків роботи в майбутньому. Запропонований алгоритм забезпечує стійкість шляхом ініціалізації ваг, навчених на попередньому часовому кроці. Планується його розширення, для включення метрики стійкості з модифікаціями, що забезпечують гарну роботу при виявленні аномалій. Також планується надати теоретичне уявлення про модель і визначити межі її продуктивності.

Подяка

Ця робота була частково підтримана дослідницьким грантом NSF IIS1254206 та IIS-1619458. Твердження та висновки належать авторам, їх не слід інтерпретувати як відтворення офіціальної політики фінансового агентства чи уряду США. Робота була також частково підтримана стипендією USC Viterbi Graduate PhD.

Список джерел

[Ahmed et al., 2013] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J Smola. Distributed large-scale natural graph factorization. In 22nd Intl. World Wide Web Conference, pages 37–48, 2013.

[Belkin and Niyogi, 2001] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In Proc. 13th Advances in Neural Information Processing Systems, pages 585–591, 2001.

[Bengio et al., 2013] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. IEEE transactions on pattern analysis and machine intelligence, 35(8):1798–1828, 2013.

[Cao et al., 2015] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In Proc. 21st Intl. Conf. on Knowledge Discovery and Data Mining, pages 891–900, 2015.

[Cao et al., 2016] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In AAAI, pages 1145–1152, 2016.

[Chang et al., 2015] Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C Aggarwal, and Thomas S Huang. Heterogeneous network embedding via deep architectures. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 119–128. ACM, 2015.

[Chen et al., 2015] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. arXiv preprint arXiv:1511.05641, 2015.

[Dai et al., 2017] Hanjun Dai, Yichen Wang, Rakshit Trivedi, and Le Song. Deep coevolutionary network: Embedding user and item features for recommendation, 2017.

[Gehrke et al., 2003] Johannes Gehrke, Paul Ginsparg, and Jon Kleinberg. Overview of the 2003 kdd cup. ACM SIGKDD Explorations Newsletter, 5(2):149–151, 2003.

[Glorot et al., 2011] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Proc. 14th Intl. Conf. on Artificial Intelligence and Statistics, page 275, 2011.

[Goyal and Ferrara, 2017] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. arXiv preprint arXiv:1705.02801, 2017.

[Grover and Leskovec, 2016] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In Proc. 22nd Intl. Conf. on Knowledge Discovery and Data Mining, pages 855–864, 2016.

[Hamilton et al., 2016] William L Hamilton, Jure Leskovec, and Dan Jurafsky. Diachronic word embeddings reveal statistical laws of semantic change. arXiv preprint arXiv:1605.09096, 2016.

[Huang et al., 2017a] Xiao Huang, Jundong Li, and Xia Hu. Accelerated attributed network embedding. In Proceedings of the 2017 SIAM International Conference on Data Mining, pages 633–641. SIAM, 2017.

[Huang et al., 2017b] Xiao Huang, Jundong Li, and Xia Hu. Label informed attributed network embedding. In Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, pages 731–739. ACM, 2017.

[Klimt and Yang, 2004] Bryan Klimt and Yiming Yang. The enron corpus: A new dataset for email classification research. In European Conference on Machine Learning, pages 217–226, 2004.

[Kulkarni et al., 2015] Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. Statistically significant detection of linguistic change. In 24th Intl. World Wide Web Conference, pages 625–635. ACM, 2015.

[Leskovec and Krevl, 2014] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection, 2014.

[Leskovec et al., 2007] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. ACM Transactions on Knowledge Discovery from Data (TKDD), 1(1):2, 2007.

[Maaten and Hinton, 2008] Laurens van der Maaten and Geoffrey Hinton. Visualizing high-dimensional data using t-sne. Journal of Machine Learning Research, 9(Nov): 2579–2605, 2008.

[Niepert et al., 2016] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In International Conference on Machine Learning, pages 2014–2023, 2016.

[Ou et al., 2016] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In Proc. 22nd Intl. Conf. on Knowledge Discovery and Data Mining, pages 1105–1114, 2016.

[Park et al., 2009] Youngser Park, C Priebe, D Marchette, and Abdou Youssef. Anomaly detection using scan statistics on time series hypergraphs. In Link Analysis, Counterterrorism and Security (LACTS) Conference, page 9, 2009.

[Perozzi et al., 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In Proc. 20th Intl. Conf. on Knowledge Discovery and Data Mining, pages 701–710, 2014.

[Roweis and Saul, 2000] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. science, 290(5500):2323–2326, 2000.

[Sun et al., 2007] Jimeng Sun, Christos Faloutsos, SpirosPapadimitriou, and Philip S Yu. Graphscope: parameterfree mining of large time-evolving graphs. In Proc. 13th Intl. Conf. on Knowledge Discovery and Data Mining, pages 687–696, 2007.

[Sutskever et al., 2013] Ilya Sutskever, James Martens, George E Dahl, and Geoffrey E Hinton. On the importance of initialization and momentum in deep learning. In Proc. 30th Intl. Conf. on Machine Learning, pages 1139–1147, 2013.

[Tang et al., 2015] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Largescale information network embedding. In 24th Intl. World Wide Web Conference, pages 1067–1077, 2015.

[Tenenbaum et al., 2000] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. science, 290(5500):2319–2323, 2000.

[Wang and Wong, 1987] Yuchung J Wang and George Y Wong. Stochastic blockmodels for directed graphs. Journal of the American Statistical Association, 82(397):8–19, 1987.

[Wang et al., 2016] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In Proc. 22nd Intl. Conf. on Knowledge Discovery and Data Mining, pages 1225–1234, 2016.

[Zhu et al., 2016] Linhong Zhu, Dong Guo, Junming Yin, Greg Ver Steeg, and Aram Galstyan. Scalable temporal latent space inference for link prediction in dynamic social networks. IEEE Transactions on Knowledge and Data Engineering, 28(10):2765–2777, 2016.