Міністерство освіти і науки України

Національний університет “Києво-Могилянська академія”

Факультет інформатики

Кафедра математики

Курсова робота

освітній ступінь — магістр

на тему: **“Нейронні мережі на графах”**

Виконав: студент 1-го року навчання,
освітньо-наукової програми
“Прикладна математика”, 113
Гапоненко Владислав Олександрович

Керівник Козеренко С.О.,
Кандидат фіз.-мат. наук

Курсова робота захищена
з оцінкою _____

Секретар ЕК _____
“_____”_____ 20__р.

Київ – 20__

Зміст

Вступ

Графи є узагальненою мовою для опису та аналізу об'єктів та структур зі взаємозв'язками. З цього випливає підхід до сприйняття навколишнього світу в рамках певним чином пов'язаних між собою об'єктів. Існує багато типів даних, що можуть бути представленими як граф. Серед таких даних можна назвати графи подій, що описують послідовності подій. Також, комп'ютерні мережі, схема станцій метро, соціальні мережі, комунікаційні мережі, мережа цитувань або інтернет можуть бути виражені природнім чином у вигляді графа. З іншого боку графи здатні варажати набагато складніші об'єкти. Чудовим прикладом є граф сцени, що представляє об'єкти на фотографії або зображенні у вигляді дерева. Також, програмний код може бути представленим у вигляді дерева лексичних одиниць.

В історії розвитку нейронних мереж було багато спроб знайти спосіб, що дозволив би справлятися з даними у вигляді графів з довільною структурою. Ранні роботи Пауло Фрасконі та Алесандро Спердутті висвітлювали рекурсивні нейронні мережі для обробки даних у вигляді напрямлених ациклічних графів. Нейронні мережі на графах вперше були представлені в роботах Марко Горі та Франко Скарсетті як узагальнення рекурсивних нейронних мереж, що здатне працювати з більш широким класом графів.

Звичані нейронні мережі здатні ефективно працювати тільки з типами даних, що можуть бути виражені як послідовності або вектори, тому постала необхідність в знаходженні підходу, що здатний виражати складніші структури даних. Графи є природнім типом даних для нейронних мереж на графах, тож вони володіють більшим експресивним потенціалом аніж класичні нейронні мережі. З цього також випливає необхідність їх дослідження.

Метою роботи є дослідження та опис нейронних мереж на графах та підходів до розв'язку задач за їх допомогою.

Мета завдання зумовила наступне наукове завдання:

1. Дослідити задачі, що розв'язуються за допомогою нейронних мереж на графах.

2. Проаналізувати структуру різних видів нейронних мереж на графах.

3. Порівняти успішність різних видів нейронних мереж на графах в контексті проаналізованих задач.

Об'єктом дослідження є нейронні мережі на графах.

Методами дослідження є аналіз наукової літератури.

Оскільки спосіб роботи нейронної мережі є неінтерпретовним, межі застосування різних типів нейронних мереж окреслюються за допомогою емпіричного досвіду. Тож, дана робота може використовуватися як довідник по використанню різних типів нейронних мереж на графах.

Основні поняття

Для опису графів, що здатні репрезентувати щонайбільшу кількість об'єктів ми використовуємо узагальне поняття графів. Графом G ми називаємо пару (V, E) де V є множиною вершин, а E є множиною дуг. Множину вершин, що з'єднані з $v \in V$ дугою ми позначаємо $ne(v)$, а множину дуг, що з'єднані з v ми позначаємо $co(v)$. Також, вершини та ребра можуть мати мітки у вигляді дійсних векторів. Мітки вершини $v \in V$ та ребра $(v_1, v_2) \in E$ позначаємо $l_v \in R^{l_V}$ та $l_{(v_1,v_2)} \in R^{l_E}$ відповідно. Визначимо l як вектор отриманий в результаті конкатенації всіх міток графа. Визначимо більш узагальнену позначку міток l_S як вектор отриманий в результаті конкатенації міток множини $S \subset V$ або $S \subset E$. Таким чином $l_{ne(v)}$ означатиме вектор, що є конкатенацією всіх міток сусідів $v \in V$. Мітки зазвичай є носіями інформації об'єктів або зв'язків між ними. Наприклад, виражаючи зображення у вигляді графа ми використовуємо граф сцени, тобто граф вершини якого є репрезентацію об'єктів сцени, а ребра можуть виражати відстань між відповідними об'єктами сцени. З цього випливає, що ребра такого графа матимуть мітки у вигляді одновимірних векторів. Ми також допускаємо співіснування напрямлених та ненапрямлених векторів у графі одночасно. Їх можна буде розрізняти за допомогою сигнальних міток, що будуть слугувати виключно індикаторами напрямленості. Також, ми припускаємо, що графи можуть бути позиційними або непозиційними. Позиційні графи відрізнятимуться від інших тим, що всі сусіди $ne(v)$ для $\forall v \in V$ матимуть логічну позицію відносно v. Тобто, для $\forall v \in V$ визначено ін'єктивну функцію $\phi_v : ne(v) \to N$ яка присвоює кожному сусіду $u \in ne(v)$ позицію $\phi_v(u)$.

Загальний опис нейронних мереж на графах

Задача тренування нейронної мережі як задача оптимізації

Підходи то тренування нейронних мереж можуть бути розділені на декілька типів як навчання із вчителем, навчання з підкріпленням та інші. В роботі розглядаються підходи, що належать до класу навчання із вчителем. Характерною особливістю цього підходу є використання тестових даних для тренування мережі. Тестові дані є множиною пар об'єктів з домену над яким працює нейронна мережа та відповідними їм мітками. Наприклад для задачі класифікації тяжкості урагану тестовими даними була б множина характеристик відомих ураганів і поставлена їм у відповідність оцінка тяжкості. Тобто, результатом тренування нейронної мережі є функція f, що ставить у відповідність вхідним даним x мітку y, де x може бути вектором, послідовністю символів, матрицею чи графом. Фактично, це є задачею оптимізації $min_{\Theta} L(y, f(x))$, де Θ є множиною параметрів функції f, що оптимізуються, а L є функцією втрат, що показує різницю між бажаним результатом y та функцією $f(x)$.

Процес навчання звичайної нейронної мережі

Процес навчання нейронної мережі є ітеративним процесом корекції шуканих параметрів Θ. Як правило, на початку навчання значення параметрів ініціалізуються довільними значеннями і потім корегуються завдяки напрямленій похідній $\nabla_{\Theta} L = (\frac{\partial L}{\partial \Theta_1}, \frac{\partial L}{\partial \Theta_2}, ...)$. Даний процес називається градієнтним спуском. Значенням напрямленої похідної є вектор що вказує напрямок найшвидшого зростання функції L, відповідно значення параметрів має змінюватися на обернене значення напрямленої похідної оскільки ми зацікавленні у спаданні L. Ітеративне знаходження похідної і корекція параметрів відбуваються доки L не досягне глобального чи локального мінімуму, тобто доки $\frac{\partial L}{\partial \Theta} = 0$. На практиці ця умова виконується не завжди, тому навчання зупиняється, коли L перестає покращуватися. Одною з головних проблем цього підходу є необхідність обрахунку функції втрат L для всіх вхідних даних x. Проте, це можна виправити за допомогою стохастичного градієнтного спуску. Сенс цього способу полягає в розділенні множини вхідних даних на неперетинні підмножини та обрахунку $\frac{\partial L}{\partial \Theta}$ і подальшої корекції параметрів Θ для кожної ітерації на основі нової підмножини те-

стових даних. Одна така підмножина називається *minibatch*, а об'єднання *minibatch*-ів, що покривають всю множину тестових даних називають епохою. Епоха має скінченну кількість елементів, тому після ітерації через всі minibath-и та при не виконанні умови $\frac{\partial L}{\partial \Theta} = 0$, створюється нова епоха і продовжується процес ітерації.

Як вже було зазначено, метою навчання є $min_{\Theta} L(y, f(x))$. Функція f може бути дуже складною, проте для початку ознайомимось з лінійним варіантом $f(x) = Wx$, де $\Theta = \{W\}$. У випадку, якщо $f(x) \in R$ W є вектором і тоді в процесі градієнтного спуску обраховується $\nabla_w f = (\frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}, ...)$. Якщо ж $f(x) \in R^n$, то W є матрицею і знаходженням напрямленої похідної фактично є знаходження якобіану $\nabla_w f = W^T$. Розглянемо складніший приклад $f(x) = W_2(W_1 x)$, де $\Theta = \{W_1, W_2\}$. Тоді функцію f можна представити як композицію функцій $f(x) = g \circ h$, де $h(x) = W_1 x$ та $g(z) = W_2 z$, тоді за правилами диференціювання складної функції напрямлена похідна f виглядатиме $\nabla_x f(x) = \frac{\partial f}{\partial (W_1 x)} \frac{\partial (W_1 x)}{\partial x}$. Процес послідовного знаходження похідних для обрахунку $\nabla_x f(x)$ для параметрів Θ називають методом зворотнього поширення помилки. Повертаючись до минулої функції $f(x) = W_2(W_1 x)$, проілюструємо метод зворотнього поширення помилки. Отримавши L в ході однієї ітерації, починаємо корекцію параметрів $\Theta = \{W_1, W_2\}$ за допомогою знаходження напрямленої похідної. Спочатку знаходимо $\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial f} \frac{\partial f}{\partial W_2}$ і таким чином отримуємо напрямок зміни параметрів W_2. Далі знаходимо $\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial f} \frac{\partial f}{\partial W_2} \frac{\partial W_2}{\partial W_1}$, використовуючи значення $\frac{\partial L}{\partial W_2}$ з минулого кроку, і таким чином отримуєм необхідні значення для корекції W_1.

Узагальнення звичайних нейронних мереж на нейронні мережі на графах

Нейронні мережі на графах можуть працювати з більш широким класом даних з різними властивостями. Оскільки граф можна задати матрицею суміжності, то виникає природнє питання про релевантність використання звичайних нейронних мереж, адже для цього достатньо перетворити матрицю у вектор. Насправді існує декілька важливих недоліків звичайних нейронних мереж, що не дозволяють їм ефективно використовувати дані представлені у вигляді графа. По-перше, звичайні нейронні мережі працюють з вхідними даними фіксованої довжини. Наприклад хімічні сполуки для завдання класифікації молекул природньо представляються у вигля-

ді графа, проте можуть суттєво відрізнятися кількістю вершин та ребер, що унеможливлює єдиноразмірне представлення для всіх вхідних даних і як наслідок неможливість використання класичних нейронних мереж. По-друге, дані виражені у вигляді непозиційних графів є інваріантними відносно порядку нумерації вершин графів, що їх репрезентує. Проте, класичні нейронні мережі, що працюють з векторами, або послідовностями сприйматимуть один граф з двома варіантами нумерації вершин як два різніх графи.

В звичайних нейронних мережах метою навчання є знаходження набору параметрів Θ функції f, що має перетворювати вхідні дані x у необхідний результат. Наприклад, для задачі класифікації пухлин, отримавши інформацію про її розміри чи форму x, в результаті $f_\Theta(x)$ очікується отримати двовимірний нормалізований вектор, що може інтерпретуватися як вектор ймовірностей належності пухлини до злоякісних або доброякісних пухлин. В нейронних мережах на графах потрібно також використовувати інформацію про суміжність вершин або ребер. Підходом що ефективно може це використовувати виявилася ітеративна агрегацію міток суміжних вершин та ребер. Тобто, після кожної ітерації мітка кожної вершини або дуги змінюється на результат агрегації міток суміжних елементів. Метою такого процесу є знаходження вектора репрезентації вершин, дуг або навіть самого графа для подальшої її інтерпретації відносно поставленої задачі. Спосіб агрегації може відрізнятися на кожній ітерації. Агрегативні функції є параметризовані, тож можемо вважати метою навчання нейронних мереж на графах знаходження параметрів агрегативних функцій, що трансформують мітки початкового графа для отримання їх чисельної репрезентації. Формалізуємо процес знаходження репрезентації для вершин графа, що має вершини з мітками, та агрегативної функції, що є спільною для всіх ітерацій. Позначатимемо номер ітерації t, а номер кінцевої ітерації як T. Позначатимемо вкладення вершини v та ребра e на t-тій ітерації l_v^t та l_e^t. Покладемо агрегативну функцію для кожної вершини для t-тої ітерації $f = \sigma(W_t \sum_{u \in ne(v)} \frac{l_u^t}{|ne(u)|} + B_t l_v^t)$, де W_t та B_t є параметрами f на t-тій ітерації, а σ є нелінійною функцією. В цьому прикладі f агрегує середнє значення міток сусідів із власною міткою вершини v для $\forall v \in V(G)$. Тоді процес трансформації міток кожної вершини можна описати як

$$l_v^0 = l_v$$

$$l_v^{t+1} = \sigma(W_t \sum_{u \in ne(v)} \frac{l_u^t}{|ne(u)|} + B_t l_v^t), \forall v \in V(G), \forall t \in \{0, ..., T-1\}$$

$$r_v = l_v^T$$

Де r_v позначає репрезентацію вершини v після всіх агрегацій. Функція f відповідає за спосіб отримання вершиною v інформації від сусідніх вершин, тому f також називають транзиційною функцією. Не важко побачити, що кількість ітерацій відповідає відстані з якої v отримує інформацію.

Навчання нейронних мереж на графах

В попередньому розділі згадувалося, що задача визначає необхідний тип репрезентації елементів графа. Так для задачі класифікації графа в ході навчання необхідно знайти чисельну репрезентацію графа, для задачі класифікації вершин графа необхідно знайти репрезентацію кожної вершини, а для задачі передбачення дуг потрібно знайти попарну репрезентацію всіх вершин на основі чого можна було б робити висновок на рахунок можливості існування дуги між вершинами.

Для знаходження репрезентації y'_v вершин графа G можна використовувати r_v $\forall v \in V(G)$ отриману в результаті процесу застосування транзиційної функції f. Проте для задачі класифікації, як правило, необхідно використовувати чисельну репрезентацію вимірність якої дорівнює кількості класів, тому d-вимірний вектор r_v трансформують в $y'_v = Wr_v$, де $W \in R^{k \times d}$.

Для задачі передбачення дуг в графі необхідне знаходження репрезентації всіх пар вершин графа. Тобто $y'_{vu} = Transform(r_v, r_u)$, де $Transform$ є функцією для отримання репрезентації пари вершин u та v. Такою функцією може бути конкатенація векторів і їх подальше перетворення у k-вимірний вектор $y'_v = W\binom{r_v}{r_u}$, де $W \in R^{k \times 2d}$. Також можливо використати просто множення векторів $y'_v = r_v^T r_u$, де одновимірний y'_v підійде для задачі передбачення дуг. Проте, для задачі класифікації дуг з k класами необхідно отримати k-вимірний y'_v, тому більш узагальнений підхід полягає в знаходженні кожної компоненти окремо та їх подальша конкатенація. Знаходження i-тої компоненти виглядає як $y_v^{'(i)} = r_v^T W^i r_u$, де W^i параметр, що визначається в результаті навчання.

Задача знаходження репрезентації графа G потребує агрегації міток всіх об'єктів графа. Чисельну репрезентацію графа отриману в результаті T ітерацій позначатимемо y'_G. Для графів з поміченими вершинами можливими агрегативними функціями може бути знаходження середнього значення репрезентації всіх вершин $y'_G = \sum_{v \in V(G} \frac{r_v}{|V(G)|}$, знаходження

найбільшої чисельної репрезентації вершини $y'_G = Max(r_v) \forall v \in V_G$, або знаходження суми всіх репрезентацій вершин $y'_G = \sum_{v \in V(G)} r_v$. Проте, при знаходженні суми всіх вкладень ми можемо втратити інформацію. Наприклад розглянемо два графа G_1 та G_2. Для графа G позначатимемо S_G множину всіх чисельних репрезентацій вершин після T ітерацій. Тоді покладемо $S_{G_1} = \{-2, -1, 0, 1, 2\}$ та $S_{G_2} = \{-20, -10, 0, 10, 20\}$. Тоді репрезентації цих двох графів будуть однаковими $y'_{G_1} = y'_{G_2} = 0$, проте вкладення їх вершин суттєво відрізняються. Вирішенням цієї проблеми є ієрархічна агрегація, коли множина вкладень ділиться на неперетинні підмножини, знаходиться репрезентація кожної підмножини окремо з використанням нелінійної функції та потім знаходиться репрезентація всього графа в результаті агрегації репрезентацій всіх підмножин. Роглядаючи минулий приклад, поділимо S_{G_1} та S_{G_2} на дві підмножини $S^{(1)}_{G_1} = \{-2, -1, 0\}$, $S^{(2)}_{G_1} = \{1, 2\}$ та $S^{(1)}_{G_2} = \{-20, -10, 0\}$, $S^{(2)}_{G_2} = \{10, 20\}$ відповідно. Агрегувати будемо за допомогою звичайної суми вкладеною в нелінійну функцію $ReLU(x) = max(0, x)$, тобто за допомогою $ReLU(SUM(S))$. Тоді отримаємо наступні репрезентації підмножин $y'_{S_{G^{(1)}_1}} = 0$, $y'_{S_{G^{(2)}_1}} = 3$, $y'_{S_{G^{(1)}_2}} = 0$ та $y'_{S_{G^{(2)}_2}} = 30$ і відповідні репрезентації графів $y'_{G_1} = 3$ та $y'_{G_2} = 30$, що краще описують різницю між графами.

Задачі знаходження репрезентацій можна умовно розділити на задачі класифікації, де необхідно отримати дискретне значення зі скінченної множини, та задачі регресії, де передбачається неперервне значення. Кожна з цих задач потребує різні функції втрат. Для задачі класифікації зазвичай використовується перехресна ентропія $CE(y^{(i)}, y^{'(i)}) = -\sum_{j=1}^{k} y_j^{(i)} log(y_j^{'(i)})$, де k є кількістю класів, $y^{(i)} \in R^k$ є унітарним вектором, $y^{'(i)} \in R^k$ є нормалізованим вектором, а i відповідає елементу тренувальних даних. Загальна функція втрат для всіх N елементів тренувальних даних виглядає як $L = \sum_{i=1}^{N} CE(y^{(i), y^{'(i)}})$. Для задачі регресії зазвичай використовується середня квадратична помилка. Для k-вимірних елементів вона виглядає як $MSE(y^{(i)}, y^{'(i)}) = -\sum_{j=1}^{k} (y_j^{(i)} - y_j^{'(i)})^2$, де $y^{(i)}$ є шуканим k-вимірними дійсним вектором, а $y^{'(i)}$ є k-вимірним дійсним вектором, що був передбачений. Тоді загальна середня квадратична помилка для всіх N тренувальних даних вираховується $L = \sum_{i=1}^{N} MSE(y^{(i), y^{'(i)}})$.

Подальший процес тренування нейронної мережі є ідентичним до описаного процесу градієнтного спуску для звичайних нейронних мереж. Тобто, після кожної спроби знаходження потрібної репрезентації в результаті

ітеративного застосування транзиційної функції, вираховується помилка L а параметри транзиційної функції корегуються за допомогою зворотнього поширення помилки. Цей процес повторюється доки результативність нейронної мережі не перестане покращуватися.

Види нейронних мереж на графах

Основна відмінність різних нейронних мереж на графах полягає у використанні різних транзиційних функцій. В даному розділі буде розглянуто основу будь-якої транзиційної функції, особливості різних нейронних мереж на графах, мету їх створення та найпопулярніші архітектури.

Основою більшості транзиційних функцій є лапласіан L, який дозволяє працювати з вершиною графа як з пікселем працює конволюційна нейронна мережа. Розглянемо довільний ненапрямлений граф G, що $|V(G)| = n$, та його матрицю суміжності A. Зафіксуємо довільний порядок вершин графа. Означимо D як діагональну матрицю степенів вершин графа G, тобто $D_v = \sum_u A_{vu}$, де $v \in V(G)$ та $u \in ne(v)$, а A_{vu} є елементом матриці суміжності що знаходиться в рядку, що відповідає вершині v та стовпцику, що відповідає вершині u. Тоді лапласіаном L буде $n \times n$ матриця $L = D - A$. Тепер означимо поліном у формі

$$p_w(L) = w_0 I_n + w_1 L + w_2 L^2 + ... + w_k L^k = \sum_{i=0}^{k} w_i L_i$$

Такий поліном однозначно визначається вектором коефіцієнтів $w = [w_0, ..., w_k]$, а сам поліном $p_w(L)$ є $n \times n$ матрицею. Такі поліноми є еквівалентом фільтрів у ковнолюційній нейронній мережі. Покладемо $l_v \in R$ для $\forall v \in V(G)$ та вектор $x \in R^n$, що його i-тий елемент відповідає мітці вершини, що відповідає i-тому рядку матриці A. Тобто, $x_i = l_i$. Результат множення поліноміального фільтра $p_w(L)$ на x називатимемо конволюцією поліноміального фільтра p_w та позначатимемо його як $x' = p_w(L)x$. Розглянемо роботу фільтра з вектором параметрів $w = [1, 0, ..., 0]$. В цьому випадку

$$x' = p_w(L)x = \sum_{i=0}^{k} w_i L^i x = w_0 I_n x = x$$

Розглянемо поліном більшої степені поклавши $w_1 = 1$, тоді $x' = Lx$, тож

$$l'_v = (Lx)_v = L_v x = \sum_{u \in G} L_{uv} l_u = \sum_{u \in G} (D_{vu} - A_{vu}) l_u = D_v l_v - \sum_{u \in ne(v)} l_u$$

З цього можемо зробити висновок, що мітка відповідної вершини поєднується з мітками суміжних вершин. Також, справджується твердження, що степінь полінома p_w також є відстанню, що визначає окіл для кожної вершини $v \in V(G)$ з якого v агрегує мітки в результаті дії конволюційного фільтра p_w.

$$l'_v = (p_w(L)x)_v = (p_w(L))_v x = \sum_{i=0}^{k} w_i L_v^i x = \sum_{i=0}^{k} w_i \sum_{u \in G} L_{vu}^i l_u = \sum_{i=0}^{k} w_i \sum_{\substack{u \in G \\ dist_G(v,u) \leq i}} L_{vu}^i l_u$$

Такий поліноміальний фільтр є інваріантним щодо порядку вершин.

В роботі M. Defferrard та X. Bresson був запропонований покращений підхід в контексті нейронної мережі на графах *ChebNet*. Тут $p_w(L) = \sum_{i=1}^{k} w_i T_i(L\ )$, де T_i є поліномом Чебишева першого типу, а $L\ $ є нормалізованим лапласіаном що визначається як $L\ = \frac{2L}{\lambda_{max}(L)} - I_n$. Даний підхід спрощує обрахунки, оскільки якщо $\lambda_{max}(L) > 1$, то значення L починають швидко збільшуватися. Нормалізований лапласіан $L\ $ гарантовано має $\lambda_{max} \in [-1, 1]$.

Сучасні нейронні мережі на графах використовують видозмінені транзиційні функції конволюційного поліному першої степені $(Lx)_v = D_v l_v - \sum_{u \in ne(n)} l_u$. Його можна логічно декомпозитувати на етап агрегації міток суміжних вершин v та її подальше комбінування з l_v. Оскільки такий підхід до агрегації є інваріантним до порядку вершин графа, результуюча нейронна мережа є інваріантною також. Одна дія такої транзиційної функції на кожну вершину зумовлює агрегацію інформації всіх вершин на відстані 1 від кожної вершини.

Graph Convolutional Networks

Вперше конволюційні нейронні мережі на графах були представлені в роботі T. Kipf та M. Welling. Конволюційні нейронні мережі на графах були однією з перших спроб узагальнення звичайних нейронних мереж для роботи з графами, що використовує конволюційний поліном першої степені.

Транзиційна функція Graph Convolutional Networks виглядає наступним чином

$$f(W \times \frac{\sum_{u \in ne(v)} l_u}{|ne(v)|} + B \times l_v)$$

де f є параметризованою функцією або іншою нейронної мережею чиї параметри визначаються у вході навчання, W та B є також параметрами, що визначаються у вході навчання.

Graph Attention Networks

Graph Attention Networks були представлені в роботі Petar Veličković, Guillem Cucurull та інших. Транзиційна функція цих графів використовує attention mechanism, що дозволяє запам'ятовувати великі послідовності інформації та використовувати її для знаходження репрезентації об'єктів графа в процесі послідовних застосувань транзиційної функції.

Транзиційна функція Graph Attention Networks виглядає наступним чином

$$f(W[\sum_{u \in ne(u)} \times a_{vu} \times l_u + a_{vv} \times l_v])$$

де f є параметризованою функцією або іншою нейронної мережею чиї параметри визначаються у вході навчання, W є параметром, що визначається у вході навчання, a є вагою, що генерує механізм уваги нейронної мережі A таким чином їх сума для всіх сусідів дорівнювала 1

$$a_{vu} = \frac{A(l_v, l_u)}{\sum_{u \in ne(v)} A(l_v, l_u)}$$

Graph Sample and Aggregate

William L. Hamilton, Rex Ying та Jure Leskovec представили GraphSAGE в роботі "*Inductive Representation Learning on Large Graphs*". Транзиційна функція данної нейронної мережі дозволяє продукувати маловимірні вектори, що дозволяє ефективно працювати з велики графами або графами з вершинами, що мають багатовимірні мітки. Це досягається завдяки використанню рекурентної нейронної мережі для агрегації міток із сусідніх вершин для кожної вершини.

Транзиційна функція Graph Sample and Aggregate виглядає як

$$f(W \times RNN[l_v : v \in ne(u)] + B \times l_u)$$

де f є параметризованою функцією або іншою нейронної мережею чиї параметри визначаються у вході навчання, а RNN є рекурсивною нейронною мережею, що приймає на вхід мітки сусідніх вершин цільової вершини.

Graph Isomorphism Network

Вперше Graph Isomorphism Network були представлені в роботі Xu Keyulu та інших як спосіб альтернативний до *Weisfeiler-Lehman* тесту для визначення ізоморфних графів. Основною особливістю агрегативної функції в *GIN* є її ін'єктивність, тобто кожний образ мітки асоційованої з вершиною має унікальний праобраз. Така особливість дозволяє досягти високої дискримінатизації. В контексті задачі визначення ізоморфних графів *GIN* не постунається вже згаданому *Weisfeiler-Lehman* тесту.

Транзиційна функція Graph Isomorphism Network виглядає наступним чином

$$f((1+r) \times l_v + \sum_{u \in ne(v)} \times l_u)$$

де f є параметризованою функцією або іншою нейронної мережею чиї параметри визначаються у вході навчання, а r є параметром. В роботі Xu Keyulu було запропоновано використовувати багатошаровий перцептрон як f.

Задача розпізнавання написаних цифр

Класифікація зображень є фундаментальним завданням комп'ютерного зору. Наприклад, алгоритм, що виносить рішення про присутність якогось об'єкта на фотографі, належить до алгоритмів комп'ютерного зору. Розпізнавання об'єктів є тривіальним завданням для людей, проте надійний алгоритм класифікації зображень досі є не легкою задачею комп'ютерного зору.

Однією з початкових задач у класифікації зображень із використання нейронних мереж на графах є представлення зображення у вигляді графа. Існує велика кількість підходів кожний з яких вмотивований специфічними потребами для виконання тієї чи іншої задачі. Розглянемо два найбільш загальних підходи. Перший підхід називають королівським графом. Назва цього підходу зумовлена способом побудови цього графа за зображенням, що має прозору аналогії з шахами. Вершини графа ми вважаємо квадратами шахової дошки, а суміжними вершинами для випадкової вершини $v \in V$ ми вважаємо всі вершини до яких може дійти король за один хід. Тобто кожна звершин графа, що не знаходиться зкраю, буде з'єднана з усіма 8 вершинами дугами. Також, кожна з вершин матиме мітку або вкладення у вигляді одномірного вектора у випадку чорно-білоного зображення, або 3-вимірного вектора у випадку кольорового зображення.

Також існує підхід з використанням стиснутого графа, що також оптимізує роботу із зображеннями високого розширення оскільки кількість пікселів навіть для зображень низького розширення є відносно великою. Проілюструємо процес створення стиснутого графа з попередньо згаданого королівського графа. Для цього кожній дузі графа G $r_{ij} \in E(G)$ призначається вага що залежить від міток вершин $v_i, v_j \in V$ наступним чином $w_{ij} = 1 - ||l_{v_i} - l_{v_j}||$. Також ми вважаємо, що значення у вкладеннях вершин є $0 \leq l_v \leq 1, \forall v \in V$. До цього графа застосовується метод Лувена для об'єднання схожих між собою вершин у спільноти і отримується стиснутий граф з'єднаних між собою спільнот вершин. Кожна спільнота вершин у стиснутому графі є вершиною з міткою ще є середнім значенням міток всіх вершин спільноти. Дуга між двома вершинами такого графа присутня тільки тоді, коли присутня хоч одна дуга між вершинами цих спільнот, напрямок зберігається.

Для виконання завдання розпізнавання зображень в роботі G. Nikolentos, T. Michalis та R. Rivera було розглянуто три архітектури нейронних мереж на графах для знаходження репрезентації графа, а також досліджена їх

успішність.

Першою було розглянуто архітектуру, що складається з декількох послідовних застосувань транзиційної функції та подальшої агрегації міток всіх вершин. Кожний застосунок транзиційної функції буде змінювати або ж оновлювати мітки всіх вершин на основі вкладень їх сусідів. Після застосувань транзиційних для знаходження репрезентації зображення застосовуватиметься агрегація всіх міток.

Друга архітектура це декілька послідовних застосувань транзиційної функції та конволюційна нейронна мережа. Процедуру застосувань транзиційної функції можна розглядати як метод оновлення вкладень для всіх вершин. Після застосування функції отримаємо щось, що структурно можна розглядати як зображення. Тому такий об'єкт можна передати конволюційній нейронній мережі для знаходження репрезентації графа. Зверніть увагу, що модель такого типу може застосовуватися тільки для завдання класифікації зображень, що представлені у вигляді королівського графа і не може бути застосована до графав загального вигляду, адже конволюційна нейронна мережа потребує чіткої табличної структури вхідних даних.

Останньою було розглянуто архітектуру, що складається з декількох застосувань транзиційної функції, процесу кластеризації та агрегації. В результаті кластеризації певна підмножина вершин заміняється однією вершиною, що нагадує створення скороченого графу. Вкладення таких нових вершин вираховується завдяки функції, що інваріантна до перестановок вершин як сума, знаходження середнього чи максимального значення серед кластеризованих вершин. В Процесі кластеризації вдічі зменшуємо кількість вершин. Також кластеризація може чередуватися із застосуванням транзиційної функції. Після всіх застосувань транзиційної функції та кластеризацій, значення всіх міток вершин агрегуються для знаходження репрезентації графа.

Розглянемо агрегативні функції, що використовуються для виведення репрезентації графа. Нагадаємо, що дані функції мають бути інваріантними відносно порядку вершин. Однією з таких функцій є звичайна сума всіх міток вершин. Також, використовуватиметься середнє значення міток всіх вершин графа та знаходження вектора кожний елемент якого є максимальним значенням серед відповідних елементів міток всіх вершин. Також використовується *SortPool*. Ця агрегативна функція спочатку сортує репрезентації всіх вершин та повертає перші k вершин. Якщо вершин менше ніж k, вільне місце заповнюється нульовими векторами. Вершини сортуються відносно останнього елементу вектору репрезентації вершини за спаданням

значення.

Нейронні мережі тренуються та перевіряються на основі даних з набору зображень вручну написаних цифр *MNIST*. Набір даних поділяється на тренувальний та тестовий по 60000 та 10000 забражень відповідно. Кожне зображення в цьому наборі даних представлене у вигляді 28×28 матриці пікселів. Кожне фото зображує цифру $i \in \{0, ..., 9\}$. Тобто, в контексті задачі класифікації, присутньо 10 класів. Кожне зображення потім представляється у вигляді королівського графа, дугам призначається вага за попередньо розглянутим алгоритмом та створюється скорочений граф.

Для подальшого тренування моделей і оцінки їх успішності було створено неперетинні тренувальні та валідаційні множини зображень шляхом випадкового відбору 10000 та 1000 зображень з тренувального набору даних *MNIST* відповідно. Моделі навчалися на основі тренувального набору з 10000 зображень і потім визначалася їх успішність на валідаційному набору даних. Як метрика успішності використовувалася точність, що є відношенням кількості вірно класифікованих зображень до кількості всіх зображень. Репрезентація графа отримана в результаті роботи різних моделей передавалася до багатошарового перцептрону з двома прихованими шарами. Результатом роботи багатошарового перцептрону є нормований 10-ти вимірний вектор завдяки функції $Softmax$. Всі можливі кофігурації моделей тренувалися 100 епох з *minibatch*-ами по 64 зображення. Активаційною функцією для всіх транзиційних функцій є $ReLU$.

Транзиційна функція	**Sum**	**Max**	**Mean**	**SortPool**
GAT	80.2	73.8	60.9	41.3
GCN	76.4	66.8	52.0	32.7
GraphSAGE	79.5	54.6	56.9	33.0

Табл. 1: Результати тренування нейронних мереж з першою архітектурою для королівських графів

Транзиційна функція	Точність
GAT	96.6
GCN	96.4
GraphSAGE	97.1

Табл. 2: Результати тренування нейронних мереж з другою архітектурою для королівських графів

Транзиційна функція	Sum	Max	Mean	SortPool
GAT	93.3	91.3	91.0	94.6
GCN	93.3	92.8	93.3	93.2
GraphSAGE	92.6	91.9	93.3	90.9

Табл. 3: Результати тренування нейронних мереж з третьою архітектурою для королівських графів

В ході дослідження було виявлено, що використання репрезентації зображень у вигляді королівського графа призводить до кращих результатів. У всіх випадках є суттєва різниця в точності. В роботі "*Image Classification Using Graph-Based Representations and Graph Neural Networks*" це пов'язується з втратою інформації в результаті створення скороченого графа. Найбільш успішнішою моделлю серед попередньозгаданих виявилася *Graph Attention Network*. Найгіршою агрегаційною функцією виявилася *SortPool*. Це пов'язується зтим фактом, що вона ігнорує велику кількість міток вершин. У випадку королівського графа найкращою агрегаційною функцією є сума. Проте, для скороченого графа, найкращою агрегаційною функцією є знаходження вектора кожний елемент якого є максимальним значенням серед відповідних елементів міток всіх вершин.

Найкращою архітектурою для класифікації зображень за результатами дослідження є декілька послідовних застосувань транзиційної функції після чого отриманий граф проходить через конволюційну нейронну мережу для отримання його репрезентації. В роботі G. Nikolentzon-а це пояснюється тим, що дана архітектура бере найкраще з нейронних мереж на графах на конволюційних нейронних мереж. Завдяки використанню транзиційних функцій нейронні мережі на графах дають чітку репрезентацію кожної ноди графа, а конволюційні нейронні мережі природньо добре оброблюють дані, що структурно є зображеннями. Видно, що така архітектура дозволяє отримати високу точність класифікації незалежно від використаної нейрон-

Транзиційна функція	**Sum**	**Max**	**Mean**	**SortPool**
GAT	65.9	66.5	65.0	49.8
GCN	61.4	60.9	61.4	46.3
GraphSAGE	59.0	59.1	55.6	42.8

Табл. 4: Результати тренування нейронних мереж з першою архітектурою для скорочених графів

ної мережі на графах навідміну від архітектури де після послідовного застосування транзиційних функцій одразу застосовується агрегативна функція. Також, варто зауважити, що така архітектура може застосовуватися тільки до зображень репрезентованих у вигляді королівського графа.

Ще однією архітектурою, що оперує із зображеннями у вигляді королівського графа є архітектура де перед агрегацією використовується кластеризація. Такий підхід покращує всі результати архітектури без агрегації. З цього випливає, що кластеризація пікселів зображення є дуже ефективною.

Задача передбачення дуг

Задача передбачення дуг вважається найменш дослідженим типом задач на графах. Метою задачі є знаходження пропущених дуг або ребер між вершинами. Розв'язування цієї задачі лежить в основі багатьох прикладних застосунків як рекомендаційні системи в соціальних мережах та онлайн кінотеатрах, або передбачення результату взаємодії протеїнів.

Існує багато традиційних підходів до розв'язування задачі передбачення ребер, деякі з них базуються на евристиках, деякі пов'язані з аналізом околів вершин та подальшим оцінюванням їх подібності. Загалом існує два основні підходи до розв'язування данної задачі за допомогою нейронних мереж на графах. Одним з таких є *Graph autoencoder* або *GAE* метод. Сенс цього підходу полягає у вивченні репрезентацій вершин графа за допомогою послідовного застосування транзиційних функцій відповідної нейронної мережі та попарній агрегації репрезентацій вершин для подальшого передбачення можливості їх поєднаності ребром. Інший підхід називають *SEAL* методом. Сенс цього методу полягає в створенні спеціального підграфа вершин з n-околу кожної пари нод графа. Кожна вершина такого підграфа позначається відповідним лейблом, що залежить від відстані даної вершини до нод з околів яких був створений підграф. Далі завдяки нейронній мережі на графах отримується репрезентація даного підграфа, що вважається репрезентацією можливого ребра на основі якої визначається можливість поєднання цих нод ребром. З попередньо розглянутими нейронними мережами на графах використовують перший *GAE* підхід.

Graph Autoencoder метод спочатку використовує певну кількість разів транзиційну функцію певної нейронної мережі на графах, і таким чином отримує репрезентацію l_v для кожної вершини $v \in V$. Потім використовується $f(l_v, l_u)$ для передбачення ребра або дуги між вершинами $v, u \in V$. В оригінальному *Graph Autoencoder* представленому в роботі T. Kipf та M. Welling використовувалося два застосування транзиційної функції конволюційної нейронної мережі на графах та багатошаровий перцептрон замість $f(l_v, l_u)$ для передбачення ребер. У багатьох застосунках замість перцептрона використовується середнє значення репрезентацій вершин або сума. В подальшому, згадуючи *GAE* метод, матимемо на увазі узагальнену структуру методу без прив'язки до використання конктретної нейронної мережі на графах або функції для передбачення ребер f.

Успішність моделей буде досліджуватися на основі набору даних *ogbl-citation*. Даний набір даних є графом з 2900000 вершин та 30600000 ребер,

де вершини репрезентують статті, а ребра між ними означають цитування однієї із статей в іншій. Для оцінювання нейронних мереж використовується метрика *MRR* (Mean Reciprocal rank).

Архітектура всіх нейронних мереж складається з три разового застосування транзиційної функції та застосування багатошарового перцептрону до всіх пар вершин для передбачення існування ребра між ними.

Транзиційна функція	**Валідаційна точність**	**Тестувальна точність**
GCN	84.49	84.56
GraphSAGE	82.17	82.28

Табл. 5: Результати тренування нейронних мереж

За результатами, що наведені в таблиці, конволюційна нейронна мережа на графах є найкращим варіантом серед методів, що імплементують *GAE* метод. Її успішність майже досягає результату *SEAL* підходу на даному наборі даних, що, як правло, краще розрізняє структуру графа і тому володіє більшою дискримінативною здатністю.

Задача класифікації вершин

Задача класифікації вершин належить до категорії задач часткового навчання. Загальний підхід до розв'язування таких задач полягає в тренуванні нейронних мереж на графах для отримання правильної репрезентації вершин на основі міток вершин та структури їх околів. Нейронну мережу на графах можна розглядати як параметризовану функцію $f(X, A)$, що залежить від матриці міток вершин X та матриці суміжності A. В ході навчання нейронної мережі кожна вершина агрегує інформацію із свого околу. Результат можна вважати репрезентацією вершини. До матриці репрезентацій вершин отриману в результаті роботи f по рядково застосовують функцію $Softmax$ або ж нормовану експотенційну функцію. Після цього вираховується перехресно ентропійна помилка та шляхом градієнтного спуску корегуються параметри f.

Дослідження успішності різних нейронних мереж на графах виконане на основі роботи "*A pipeline for fair comparison of graph neural networks in node classification tasks*". Для точного порівнянння різних моделей використовувалася стала архітектура для всіх нейронних мереж. Узагальнена архітектура складається з видозміни вхідних даних для вимог конкретної нейронної мережі на графах, послідовного застосування 4 або 16 транзиційних функцій та подальша інтерпретація отриманих репрезентацій вершин за доповогою багатошарового перцептрону.

Для задачі класифікації вершин набори даних поділяються на два типи. першим типом є набори даних, що складаються з одного великого графа. Таким набором даних може бути мережа статей, що природньо виражається графом з вершинами, що репрезентують статті, та дугами, що відповідають за цитування. Набори даних, що складають з багатьох графів належать до другого типу. Тип набору даних впливає тільки на підхід до тренування нейронної мережі. Так набори першого типу не можуть бути поділені на підмножини, адже вони складаються тільки з одного графа, тому під час тренування не використовуються *minibatch*-и. В цій роботі ми розглянемо успішність нейронних мереж на графах на наборі даних *Cora*, що є мережою статей. Завданням для цього набору даних є класифікація теми статей на основі структури та інформації їх околу. Він складається з оного графу, 2708 вершин кожна з яких має 1433-и вимірну мітку, 5278 ненапрямлених ребер. Вершини цього графа належать до одного з 7-и класів.

Для тренування моделей використовується методика *10-fold cross -validation*. Вона полягає в умовному розділенні набору даних на 10 рівних

частин. Далі кожна така частина розділяється на тренувальну, тестову та валідаційну частини, причому відношення кількості класів у кожній частині однакове. Модель тренується окремо на кожній з 10-ти частин. Процес тренування для кожної частини проводиться тричі після чого вираховується середня точність для кожної такої частини. Фінальною точністю моделі вважається середнє значення точностей моделі на раніше згаданих 10-ти частинах. Кожна модель повторює цей процес чотири рази, причому кожного разу розподіл набору данних на 10 частин відбувається випадково. Потім, середнє значення фінальних точностей моделі в результаті чотирьох *10-fold cross-validation*-ів вважається точністю моделі.

Транзиційна функція	К-ть т. ф.	**Тренувальна точність**	**Валідаційна точність**	**точність**
GCN	4	99.9996	85.4674	84.9894
	16	99.9604	85.5166	84.6536
GraphSage	4	100.0	85.9472	85.515
	16	100.0	86.3684	85.6969
GAT	4	99.9907	85.5658	85.1954
	16	99.9569	85.981	85.1492
GIN	4	100.0	87.0388	86.7398
	16	99.9919	87.0541	87.0323

Табл. 6: Результати тренування нейронних мереж

За результатами проведених тренувань нейронних мереж на графах, всі нейронні мережі мають приблизно однакову успішність. Проте, найкращі результати має *Graph Isomorphism Network*, а *Graph Convolutional Network* справилася найгірше.

Висновки

У ході виконання магістерської роботи, було проаналізовано загальні відомості про звичайні нейронні мережі та узагальнений підхід до їх навчання. Було проаналізовано загальні відомості про нейронні мережі на графах та окремі реалізації моделей. На основі інших робіт було досліджено успішність розглянутих моделей в контексті найпоширеніших задач для нейронних мереж на графах та дані рекомендації щодо їх застосування.

Література

Paolo Frasconi, Marco Gori, and Alessandro Sperduti. A general framework for adaptive processing of data structures. IEEE transactions on Neural Networks, 9(5):768–786, 1998.

A. Sperduti and A. Starita. Supervised neural networks for the classification of structures. Trans. Neur. Netw., 8(3):714–735, May 1997. ISSN 1045-9227. doi: 10.1109/72.572108.

Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In IEEE International Joint Conference on Neural Networks, pp. 729734, 2005.

Nikolentzos, Giannis, Thomas, Michalis and Ramirez Rivera, Adin and Vazirgiannis, Michalis. (2021). Image Classification Using Graph-Based Representations and Graph Neural Networks. 10.1007/978-3-030-65351-4_12.

Xu, K., Hu, W., Leskovec, J. and Jegelka, S. (2018). How Powerful are Graph Neural Networks?. CoRR, abs/1810.00826.

Daigavane, et al., "Understanding Convolutions on Graphs Distill, 2021.

Hamilton, W. L., Ying, R., and Leskovec, J. (2017). Inductive Representation Learning on Large Graphs. arXiv. https://doi.org/10.48550/ARXIV.1706.02216

Veličkovič, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2017). Graph Attention Networks. arXiv. https://doi.org/10.48550/ARXIV.1710.10903

Kipf, T. N., and Welling, M. (2016). Semi-Supervised Classification with Graph Convolutional Networks. arXiv. https://doi.org/10.48550/ARXIV.1609.02907

Heindl, C. (2020). Graph Neural Networks for Node-Level Predictions. arXiv. https://doi.org/10.48550/ARXIV.2007.08649

Maurya, S. K., Liu, X., and Murata, T. (2021). Simplifying approach to Node Classification in Graph Neural Networks. arXiv. https://doi.org/10.48550/ARXIV.2111.06748

Yu, Z., Wang, H., Liu, Y., Böhm, C., and Shao, J. (2020, November). Community Attention Network for Semi-supervised Node Classification. In 2020 IEEE International Conference on Data Mining (ICDM) (pp. 1382-1387). IEEE.

Zhao, W., Zhou, D., Qiu, X., and Jiang, W. (2020). A pipeline for fair comparison of graph neural networks in node classification tasks. arXiv preprint arXiv:2012.10619.

Zhang, M., Li, P., Xia, Y., Wang, K., and Jin, L. (2020). Revisiting graph neural networks for link prediction.

Diao, Z. Link Prediction with Enclosing Subgraph.

Kipf, T. N., and Welling, M. (2016). Variational graph auto-encoders. arXiv preprint arXiv:1611.07308.

Zhang, M., Cui, Z., Neumann, M., and Chen, Y. (2018, April). An end-to-end deep learning architecture for graph classification. In Thirty-second AAAI conference on artificial intelligence.

Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. Advances in neural information processing systems, 29.