КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфкаційна робота

на здобуття ступеня бакалавра

за спеціальністю 121 Інженерія програмного забезпечення

на тему:

ЗАСТОСУВАННЯ ГРАФОВИХ НЕЙРОННИХ МЕРЕЖ ДО ЗАДАЧ КЛАСИФІКАЦІЇ

Виконав студент 4-го курсу
Іван КРАСНОЩОК

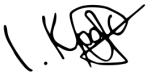
(підпис)

Науковий керівник:
доцент, кандидат фіз.-мат. наук
Ярослав ЛІНДЕР

_____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент



(підпис)

Роботу розглянуто і допущено до захисту на засіданні кафедри інтелектуальних програмних систем
«*25*» *травня* 2022р.,
протокол № *10*
Завідувач кафедри
Олександр ПРОВОТАР

_____
(підпис)

Київ – 2022

РЕФЕРАТ

Обсяг роботи 41 сторінка, 14 ілюстрацій, 5 таблиць, 16 джерел посилань.

ГРАФОВА НЕЙРОННА МЕРЕЖА, ЗГОРТКОВА ГРАФОВА МЕРЕЖА, ГРАФОВА МЕРЕЖА З УВАГОЮ, ГРАФ, ЗАДАЧА КЛАСИФІКАЦІЇ, КЛАСИФІКАЦІЯ ВЕРШИН ГРАФА, МАШИННЕ НАВЧАННЯ, НЕЙРОННА МЕРЕЖА.

Об’єктом роботи є розв’язування задач класифікації вершин гомогенного та геторогенного графів за допомогою графових нейронних мереж та методів машинного навчання. Предметом роботи є графові нейронні мережі, зокрема згорткова графова мережа, графова мережа з увагою.

Метою роботи є побудова моделей графових нейронних мереж для розв’язання задач класифікації вершин гомогенного та гетерогенного графів.

Інструменти розроблення: мова програмування Python 3, бібліотеки машинного навчання Keras, Tensorflow 2, середовище JupyterLab Desktop.

Результати роботи: розглянуто поточні розповсюджені підходи машиинного начання для роботи з графами, описано алгоритм тренування нейронних мереж, розглянуто принцип роботи графових нейронних мереж, розглянуто поширені архітектури графових нейронних мереж, побудовано моделі згорткової графової мережі та графової мережі з увагою, проаналізовано результати роботи побудованих моделей.

Результати роботи демонструють приклад успішного застосування графових нейронних мереж до задач класифікації на графах, а також можуть слугувати підґрунтям для подальших досліджень у напрямку підходів до розв’язання задач класифікації за допомогою графових нейронних мереж.

Сферами застосування графових нейронних мереж є програмні застосунки, що місять дані, які можуть бути представлені у вигляді графів,

наприклад соціальні мережі, бази даних фільмів; транспортні мережі, медицина. Значення роботи полягає у потенційному покращенні результатів задач класифікації на графах, що призводить до покращення досвіду користувача при використанні відповідних програмних застосунків, покращення досвіду подорожей та ін.

ЗМІСТ

ВСТУП

Оцінка сучасного стану об’єкта розробки. Кількість даних, що можуть бути представлені у вигляді графу зростає кожного року. Прикладом, що одразу спадає на думку є соціальні мережі, кількість користувачів яких постійно збільшуєтья, а задачі, що вони виконують ускладнюються. Мережі даних не обмежуються соціальними і включають мережі цитування, біологічні мережі транспортні мережі. Попри те, що ідея графових нейронних мереж існує більш ніж 15 років, вона почала набувати популярності тільки нещодавно.

Актуальність роботи та підстави для її виконання. Класичні нейронні мережі не враховують структуру графів даних і тому не є оптимальними для роботи з подібними даними. Для розв’язання задач машинного навчання на графах, зокрема задачі класифікації вершин, застосовуються графові нейронні мережі. Застосування графових нейронних мереж дозволяє покращити результати у задачах класифікації на графах, наприклад класифікація облікових записів шахраїв у соцмережі, класифікація утворень в організмі людини, за допомогою обробки їх молекул, як шкідливих.

Мета й завдання роботи. Метою кваліфікаційної роботи є побудова моделей графових нейронних мереж для розв’язання задач класифікації на гомогенному та гетерогенному графах. Для досягнення мети поставлено наступні завдання:

- Розглянути принцип роботи нейронних мереж.
- Розглянути проблему вивчення представлень графа.
- Розглянути алгоритми машинного навчання, що вивчають предсталення графа, та можуть слугувати та отримання опорних результатів класифікації.
- Розглянути принцип роботи графових нейронних мереж.
- Розглянути поширені моделі графових нейронних мереж.

– Побудувати моделі графових нейронних мереж для класифікації вершин у гетерогенному та гомогенному графах.

Об’єкт, методи й засоби розроблення. Об’єктом розроблення є розв’язування задач класифікації вершин гомогенного та геторогенного графів за допомогою графових нейронних мереж та методів машинного навчання.

Для розробки було обрано машину з операційною системою macOS. У якості середовища розробки було обрано JupyterLab Desktop — популярний варіант для проведення експериментів з машинного навчання. Для розробки було обрано мову Python 3, яка лідерорм у сфері машинного навчання оскільки є високорівневою, має простий синтаксис і дозволяє використовувати зручні бібліотеки для машинного навчання. У якості основної бібліотеки обрано Tensorflow 2 та Keras, що надає високрівневий API для побудову нейронних мереж. Також було використано наступні бібліотеки: pandas, numpy, gensim.

Можливі сфери застосування. Класифікація вершин графу за допомогою графових нейронних мереж має застосування у соціальних мережах, медицині (аналіз біологічних графів), транспортних мережах та ін.

РОЗДІЛ 1. НЕЙРОННІ МЕРЕЖІ

1.1. Архітектура нейронних мереж

Нейронна мережа — це математична модель, що складається з пов’язаних між собою нейронів (вузлів) подібно до людського мозку.

Одним з найпростіших прикладів нейронної мережі є багатошаровий перцептрон. Дана модель складається з вхідного шару, одного або більше прихованих шарів та вихідного шару. Кожен шар складається з нейронів, усі нейрони сусідніх шарів поєднані між собою. Багатошаровий перцептрон схематично зображений на рисунку 1.1.

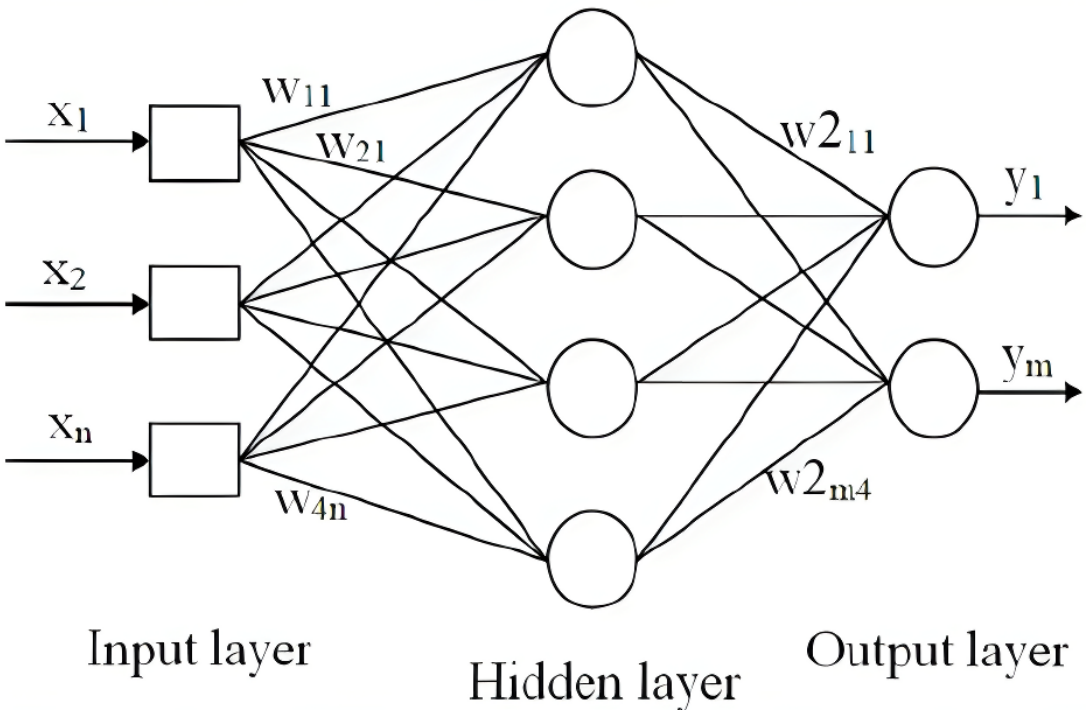


Рисунок 1.1. Багатошаровий перцептрон

З’єднання нейронів мають ваги (позначені як *w*), нейрони мають зміщення (англ. bias). Для кожного нейрона, крім вхідних, розраховується сума ваг, до якої додається зміщення:

$$n_{ij} = \sum w_{ijk} \cdot n_{(i-1)k} + b_{ij},$$

де i — номер шару в мережі, j — номер нейрона в шарі, k — номер нейрона з попереднього шару. Послідовно розраховуючи дані суми для

кожного шару мережі, матимемо результат роботи мережі на вихідному шарі.

Позначимо N_i — вектор нейронів шару i, n_{ij} — j-й елемент вектора, W_i — матриця ваг шару i, w_{ijk} — елемент j-го рядка, k-го стовпчика матриці, B_i — вектор зміщень i, b_{ij} — j-й елемент вектора. Тоді формулу для шару нейронів можна записати наступним чином:

$$N_i = W_i \cdot N_{i-1} + B_i$$

Шари нейронної мережі можуть мати функції активації. Функція активації визначає залежність вихідного сигналу нейрона від вхідного. Бажані властивості функції активації:

- Нелінійність. Якщо функція активації нелінійна, то, двошарову нейронну мережу можна вважати універсальною апроксимацією функцій [1]. Якщо декілька шарів використовують лінійну функцію активації, тоді вся мережа еквівалентна одношаровій моделі
- Неперервна диференційовність — ця властивість бажана для використання методів оптимізації заснованих на градієнті [2].
- Монотонність.

Відомим прикладом функції активації є ReLU (Rectified Linear Unit):

$$f(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

Модифікацією цієї функції є протікаюча (англ. leaky) ReLU, яка робить можливим навчання мережі за умови від’ємних значень ваг:

$$f(x;\alpha) = \begin{cases} x, & x \geq 0 \\ x\alpha, & x < 0 \end{cases}$$

Популярною функцією активації для задач бінарної класифікації є сигмоїда:

$$s(x) = \frac{1}{1 + e^{-x}}$$

Дана функція переводить будь-яке дійсне число в діапазон від нуля до одиниці.

Також для відтворення нелінійної залежності часто обирають функцію активації гіперболічний тангенс tanh:

$$tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$

Популярні функції активації зображено на рисунку 1.2.

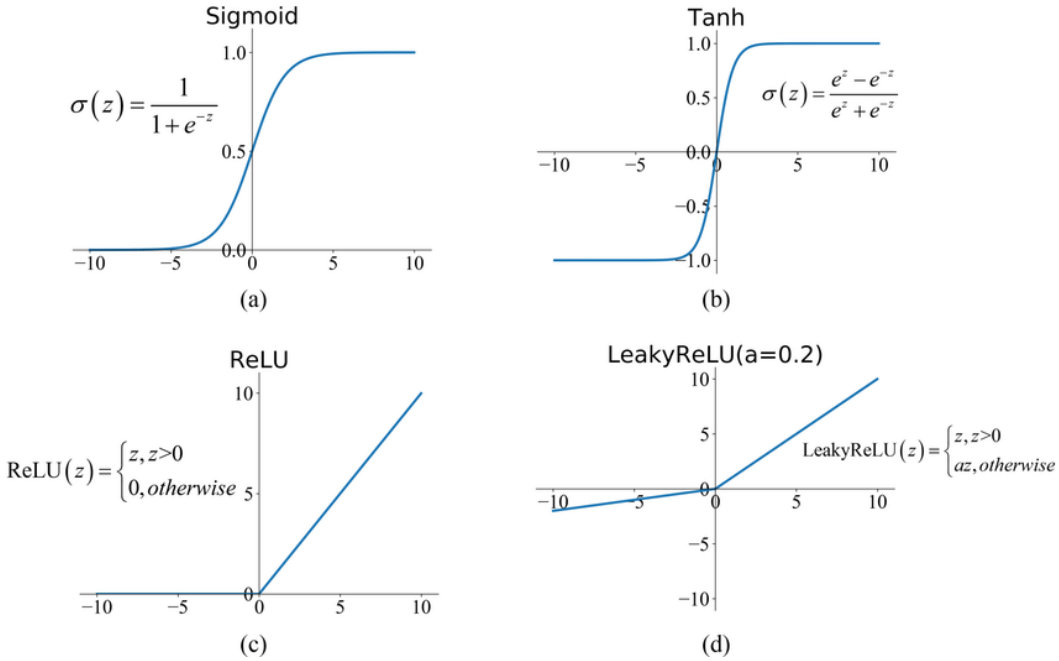


Рисунок 1.2. Популярні функції активації

Перепишемо формулу розрахунку значень для кожного нейрона можна наступним способом:

$$n_{ij} = f_i(\sum w_{ijk} \cdot n_{(i-1)k} + b_{ij})$$

Тоді формула розрахунку вектора значень шару виглядає наступним чином:

$$N_i = f_i(W_i \cdot N_{i-1} + B_i)$$

1.2. Тренування нейронних мереж

Тренування нейронної мережі — це процес знаходження оптимального набору ваг та зміщень для нейронів моделі. Найчастіше використовується метод зворотного поширення помилки (англ. backpropagation). Рухаючись від вихідного до вхідного шару алгоритм оптимізації змінює ваги (на основі їх градієнтів) так, щоб мінімізувати функцію втрат.

Приклад функції помилки для задачі бінарної класифікації — бінарна перехресна ентропія (англ. binary cross entropy):

$$-(y \cdot log(p) + (1 - y) \cdot log(1 - p))$$

Одним з найвідоміших алгоритмів оптимізації є стохастичний градієнтний спуск, який модифікує ваги наступним чином:

$$w = w - \eta \cdot \nabla Q(w)$$

де Q — функція втрат, w — ваги моделі, η — крок градієнтного спуску (англ. learning rate). Параметр η регулює швидкість навчання мережі: при більших значеннях функція втрат мінімізується швидше, але вірогідність досягнення мінімуму зменшується, при менших значеннях навпаки.

Алгоритм стохастичного градієнтного спуску має велику обчислювальну складність, тому на практиці використовують міні-батчевий градієнтний спуск (англ. batch — підвибірка). Набір даних випадковим чином розбивається на підвибірки, після чого відбувається оптимізація функції помилок для кожної з них. Один прохід по набору даних називають епохою. Після завершення епохи дані знову випадковим чином розбиваються на підвибірки.

Градієнтний спуск є найпростішим з методів градієнтної оптимізації. І хоча методи другого та більшиз порядків не є застосовними до нейронних мереж через обчислювальну складність, існує ціла низка числьних алгоритмів першого порядку, що на практиці виявляються кращими за звичнайний градієнтний спуск.

Приведемо як приклад такого алгоритму Adam (Adaptive Momentum) [3]. Його ідеї базуються на алгоритмах RMSprop (Root Mean Square Propagation) та Нестерова (Nesterov accelerated gradient). Adam підраховує як і експоненційно затухаюче середнє градієнтів, як Нестеров, так і експоненційно затухаюче середнє квадратів градієнтів, як RMSprop. Далі відбувається корекція обчислених градієнтів на основі отриманих середніх.

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_1$$
$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) g_t^2,$$

де g_t – вектор градієнту обчисленого на кроці t, β_1 та β_2 – параметри алгоритму. Так як початкові значення середніх m_t та v_t покладаються рівними нулю, робиться корекція їх значень:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

Оновлення параметрів відбувається за правилом:

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t$$

Часто проблемою навчання нейронних мереж є перенавчання (англ. overfitting). У такому випадку ваги мережі стають чутливими до специфічних властивостей тренувального набору даних, але погано узагальнють решту даних. Одним з рішень є застосування шарів виключення (англ. dropout). Цей шар випадковим чином занулює значення деяких нейронів попереднього шару (тільки під час тренування). Таким чином нейронній мережі простіше визначити набір ваг який точніше узагальнює дані.

РОЗДІЛ 2. ВИВЧЕННЯ ПРЕДСТАВЛЕНЬ

2.1. Поняття вивчення представлень

Вивчення представлень (англ. representation learning) — це набір методик машинного навчання, які дозволяють системі виявляти представлення, необхідні для виявлення ознак або класифікації з необроблених даних [4]. Машина може самостійно вивчити представлення застосовні до певної задачі, унаслідок чого зникає потреба в ручному виявленні значущих ознак.

Вивчення представлень працює шляхом скорочення даних великої розмірності до даних малої розмірності, що полегшує виявлення закономірностей і аномалій, а також забезпечує краще розуміння загальної структури даних.

2.2. Вивчення представлень для мереж

У сучасному світі можна знайти все більше прикладів даних у вигляді мережі, починаючи від віртуальних мереж: соціальні мережі, мережі цитування, телекомунікації; до фізичних мереж: транспортні мережі, біологічні мережі. Дані мережі можуть бути представлені як графи, де вершини та ребра характеризують інформацію про мережу. Графи є потужним та гнучким способом представлення даних, так інші типи даних, такі як зображення та тексти, можна розглядити як особливий випадок графу. Наприклад, зображення можна розглядати як мережу вузлів з атрибутами RGB, а текст може бути представлено у вигляді послідовну, деревоподібну або графоподібну інформацію. Тож вивчення представлень для мережі можна розглянути як завдання, що вимагає вдосконалення та узагальнення методів, розроблених для зображень, текстів тощо. На додаток до внутрішньої високої складності мережевих даних, ефективність вивяення представлення в мережах також є важливою проблемою, враховуючи масштабність багатьох мереж реального світу, що варіюються від сотень до мільйонів або навіть мільярдів вершин. Аналіз

інформаційних мереж відіграє вирішальну роль у різноманітних нових програмах у багатьох дисциплінах. Наприклад, у соціальних мережах класифікація користувачів на значущі соціальні групи корисна для багатьох важливих завдань, таких як пошук користувачів, цільова реклама та рекомендації; у комунікаційних мережах виявлення громадських структур може допомогти краще зрозуміти процес поширення чуток. Тим не менш, ефективний аналіз цих мереж значною мірою залежить від вдало сформованого представлення мережі.

Класичне конструювання ознак (англ. feature engineering) мережевих даних зазвичай зосереджується на обчисленні заздалегідь визначених функцій на графі: діаметр, середня довжина шляху, коефіцієнт кластеризації; вузлі: ступінь вузла, центральність. Ця обмежена кількість створених вручну чітко визначених функцій, хоча й описує кілька фундаментальних аспектів графів, відкидає шаблони, які не можуть бути охоплені ними. Більш того, реальні мережі, як правило, дуже складні, вимагають складних, невідомих комбінацій між цими заздалегідь визначеними ознаками або не можуть бути охарактеризовані жодною з наявних ознак. Крім того, традиційне конструювання ознак функцій графів зазвичай включає алгоритмічні обчислення з надлінійною або експоненційною складністю, що часто робить багато завдань аналітичної мережі обчислювально дорогими та важкорозв'язними у великомасштабних мережах. Наприклад, при вирішенні завдання виявлення спільноти класичні методи передбачають обчислення власного розкладу матриці з обчислювальною складністю не менш, ніж $O(n^{2,37})$ (визначається складністю алгоритмну множення матриць [5]), де n - кількість вершин. Ці обчислювальні витрати ускладнюють масштабування алгоритмів для великих мереж з мільйонами вершин.

Вивчення представлення мережі прагне вивчити приховані представлення, що будуть мати малу розмірність, для вершин мережі, зберігаючи структуру топології мережі, вміст вершин та іншу інформацію.

Після вивчення нових представлень вершин можна легко й ефективно виконувати завдання мережевої аналітики, застосовуючи звичайні векторні алгоритми машинного навчання до нового простору представлень. Одним з алгоритмів зменшення розмірності є алгоритм побудови вкладення графу (англ. graph embedding). Вхідними даними є набір незалежних однаково розподілених точок даних, алгоритми побудови вкладення графу спочатку обчислюють подібність між попарними точками даних, щоб побудувати граф спорідненості (англ. affinity graph), наприклад, граф k найближчих сусідів, а потім вкласти граф спорідненості в новий простір меншої розмірності. Однак алгоритми побудови вкладення графів зазвичай мають щонайменше квадратичну тимчасову складність відносно кількості вершин.

Багато алгоритмів навчання репрезентації мережі були запропоновані для побудови вкладень існуючих мереж, демонструючи перспективну продуктивність для різних застосувань. Ці методи створюють вкладення мережі в прихований простір малої розмірності, який зберігає близькість структури та спорідненість атрибутів. Отримані в результаті компактні, низьковимірні векторні представлення можна використати як ознаки для векторних алгоритмів машинного навчання. Це відкриває шлях для легкого та ефективного вирішення широкого кола завдань мережевої аналітики в новому векторному просторі, таких як класифікація вузлів, передбачення зв’язкі, кластеризація, мережевий синтез.

РОЗДІЛ 3. АЛГОРИТМИ МАШИННОГО НАВЧАННЯ

3.1. Випадкова хода

У математиці випадкова хода (англ. random walk) — це випадковий процес, який описує шлях, що складається з послідовності випадкових кроків у деякому математичному просторі. Випадкова хода на графі — це процес, який починається в деякій вершині і на кожному кроці часу

переходить до іншої вершини. Для незваженого графу наступна випадкова вершина обирається серед сусідніх вершин рівноймовірно. Якщо граф зважений, ймовірність обрання сусіда є пропорційною вазі відповідного ребра.

Випадкова хода є однією з основних технік для побудови алгоритмів для роботи з даними, що представлені у вигляді графу. Так застосування випадкової ходи включають виявлення спільноти, ранжування вузлів і ребер, зменшення розмірності даних [6].

3.2. Алгоритм node2vec

Розвитком випадкової ходи є алгоритм node2vec. Задача алгоритму полягає в побудові вкладень (векторів значень; англ. embedding) для кожної вершини графу. Ці вектори можуть бути використані як вхідні дані для різних моделей машинного навчання: лінійна регресія, дерево рішень та інше. Алгоритм складається з двох кроків:

1) За допомогою випадкової ходи побудувати набір послідовностей вершин графа.
2) Застосувати алгоритм побудови вкладень для отриманих послідовностей, наприклад word2vec. Оскільки word2vec зазвичай використовується для обробки тексту, то послідвності, отримані внаслідок випадкової ходи часто називають “реченням”, а вершини послідовності — “словом” або токеном

На відміну від звичайної випадкової ходи, переходи node2vec залежать не тільки від ступеня відвідуваного в даний момент вузла або ваги його ребер, але також від структури локальної мережі та останнього відвідуваного вузла. node2vec можна використовувати в таких завданнях, як виявлення спільноти, класифікація за кількома мітками та передбачення з’єднань у мережі. У node2vec можна налаштувати вагу локального та глобального пошуку мережі шляхом модуляції значень параметрів [7].

node2vec знайшов застосування, наприклад, у прогнозуванні генів, пов’язаних із хворобою Паркінсона, та рекомендаціях фільмів.

Для звичайної випадкової ходи ймовірність вибору наступного вузла розраховується на основі ваг ребер w_{vx}, тобто

$$f_{vx} = \frac{w_{vx}}{\sum_{i \in N(v)} w_{vi}},$$

Де $N(v)$ — множина вершин сусідів v. Проте даний підхід не дозволяє враховувати структуру графа та налаштовувати процес ходи для роботи з різними околами вершин. Крім того, на відміну від класичних стратегій побудови послідовності в ширину та в глибину, які підходять для структурної еквівалентності та гомофілії (прим. гомофілія — ступінь, в якій вершини графа пов’язані з подібними вершинами за певною ознакою). Тому стратегія вибору наступної вершини для ходи node2vec має враховувати той факт, що ці поняття еквівалентності не є конкуруючими або взаємновиключними, а реальні мережі зазвичай демонструють суміш обох.

Визначимо випадкову ходу node2vec, використовуючи параметри p та q. Розглянемо випадкову ходу, яка щойно пройшла ребро (t, v) і зараз знаходиться у вершині v. Для здійснення настпуного кроку визначаютья ймовірності переходу f_{vx} на ребрах (v, x), що виходять з вершини v. Використаємо наступну формулу для визначення ймовірності:

$$f_{vx} = \frac{\alpha(t, x; p; q) \cdot w_{vx}}{Z},$$

де Z — нормалізуюча константа, α визначається наступним чином:

$$\alpha(t, x; p; q) = \begin{cases} \frac{1}{p}, & d_{tx} = 0 \\ 1, & d_{tx} = 1, \\ \frac{1}{q}, & d_{tx} = 2 \end{cases}$$

де d_{tx} — відстань між вузлами t та x. Слід зазначити, що d_{tx} має належати множині {0, 1, 2}, тоді двох параметрів p та q достатньо для визначення α. Значення α для переходів з вершини v у вершину x зображено на рисунку 3.1.

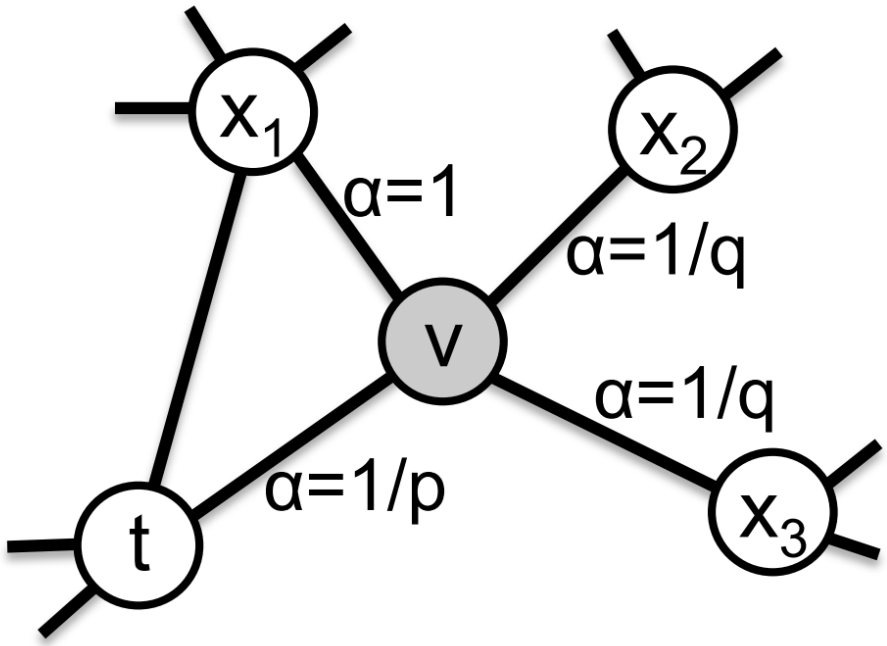


Рисунок 3.1. Переходи і значення α

Параметри p та q контролюють швидкість, з якою node2vec обходить та покидає окіл початкової вершини. Зокрема параметри дозволяють node2vec бути сумішшю процедур побудови послідовності в ширину та в глибину.

Параметер p визначає ймоврністі негайного відвідування попередньої вершини ходи. Встановлення великого значення p, що перевищує $max(q,1)$ зменшує ймовірність обрання вже відвіданої вершини в наступні два кроки ходи (якщо наступна вершина має інших сусідів). Дане значення параметра призводить середньої швидкості обходу і зменшує кількість надлишкових переходів. Якщо ж встановити мале значення p, яке не перевищує $min(q,1)$, то це спричинить збільшення

повернень і обхід буде відбуватись у локальній близькості до початкової вершини.

Параметр q дозволяє ході розрізнити “внутрішні” та “зовнішні” вершини. Якщо значення q перевищує 1, то випадкова хода тяжіє до вершин ближчих до t (рисунок 3.1). Хода такого типу схожа на пошук в ширину, алгоритм обходить вершини, які знаходяться в локальній близькості до початкової вершини. Якщо ж значення параметра q менше за 1, то хода ймовірно відвідає вершини, які знаходяться якомога далі від вершини t. Ця поведінка схожа на пошук в глибину і заохочує “зовнішній” обхід. Важливим фактором є те, що така поведінка досягається за допомогою алгоритму випадкової ходи.

Перевагаою випадкової ходи над пошуком в глибину та в ширину є часова складність, оскільки алгоритм ходи дозволяє повторне використання підпослідовностей для різних початкових вершин ходи. Симулюючи випадкову ходу довжини $l > k$, можна згенерувати k послідовностей довжини $l - k$ [7]. Таким чином складінсть генерації однієї послідовності:

$$O\left(\frac{l}{k \cdot (l-k)}\right)$$

Після генерації послідовностей для них будують вкладення за допомогою word2vec. Для утворення вкладень даний алгоритм може використовувати наступні моделі: неперервна торба слів (англ. continuous bag of words), скіп-грама (прим. n-грама з пропусками). Обидві моделі є нейромережами, які використовують шари вкладення (англ. embedding layer) для кодування слів. Після тренування моделі з шарів вкладення отримуємо вектори для вершин графа.

Користь node2vec полягає в тому, що отримані представлення вершин можна використати в різних моделях машинного навчання. Використовуючи представлення отримані в результаті роботи node2vec,

модель логістичної регресії досягає точності класифікації понад 73% на наборі даних CORA (прим. набір даних складається з 2708 вершин, які розбито на 7 класів наукових публікацій, та 5429 ребер — посилання на публікації) [8]. Слід зазначити, що більш складні моделі досягають точності до 90% на цьому наборі даних.

3.3. Розповсюдження міток

Розповсюдження міток (англ. label propagation) — алгоритм машинного навчання на графі, який розпосюджує мітки від вершин графа, що відмічені (класифіковані), до вершин, що не мають мітки. Для вершин, що мають мітку на початку роботи алгоритму, мітка фіксується. Ці вершини виступають як “джерела”, що просувають мітки по вершинам графа, які не класифіковано [9].

Нехай $(v_1, y_1), \ldots, (v_l, y_l)$ — набір вершин з мітками, де $Y_L = \{y_1, \ldots, y_l\}$ — мітки класів. C — кількість класів, усі класи наявні у відмічених даних. Нехай $(v_{l+1}, y_{l+1}), \ldots, (v_{l+u}, y_{l+u})$ — набір вершин без міток, $Y_U = \{y_{l+1}, \ldots, y_{l+u}\}$ не встановлено; зазвичай l значно менше за u. Нехай $V = \{v_1, \ldots, v_{l+u}\}$. Задача алгоритму полягає у встановленні Y_U з V та Y_L.

Мітки вершин розповсюджуються по ребрам, чим більша вага ребра, тим швидше просувається мітка. Ймовірнісна матриця переходів T розмірності $(l+u) \times (l+u)$ визначаєтья наступними чином:

$$T_{ij} = P(j \to i) = \frac{w_{ij}}{\sum_{k=1}^{l+u} w_{kj}},$$

де T_{ij} — ймовірність стрибку з вершини j до i. Також визначимо матрицю міток Y розмірності $(l+u) \times C$, де i-й рядок представляє розподіл ймовірностей міток для вершини v_i.

Алгоритм складається з наступних кроків:

1) розповсюдити мітки $Y \leftarrow TY$;
2) нормалізувати рядки матриці Y;
3) для вершин $\{v_1, \ldots, v_l\}$ повернути значення рядків матриці Y у початковий стан;
4) повторнутися до кроку 1, поки значення Y не збіглися.

Роботу алгоритму схематично проілюстровано на рисунку 3.2.

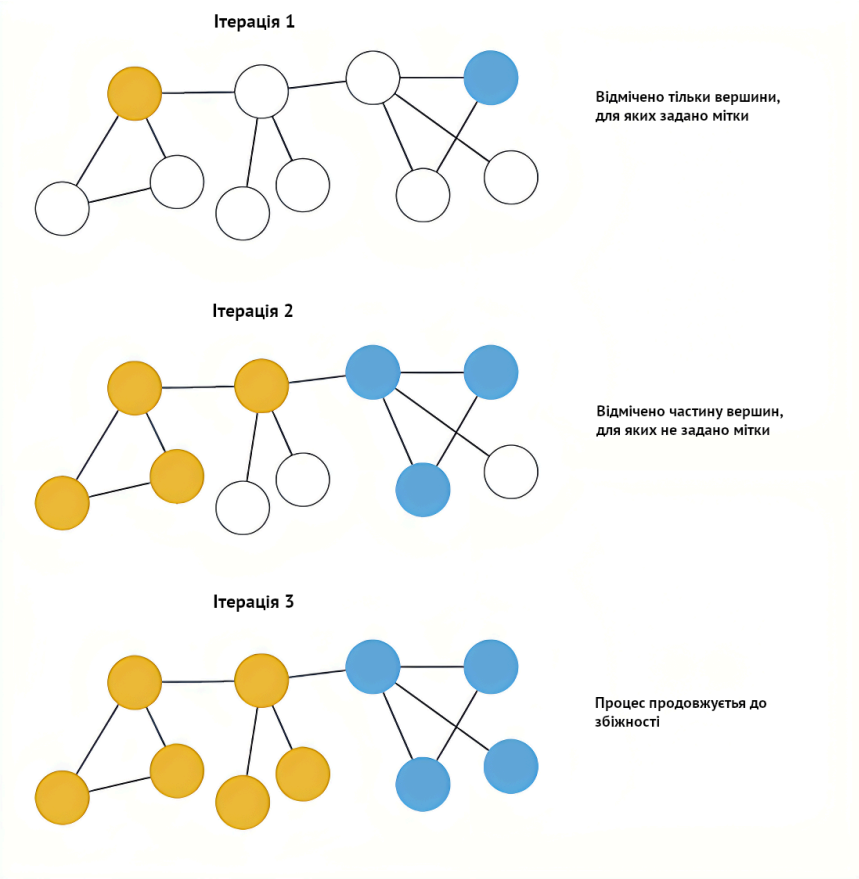


Рисунок 3.2. Розповсюдження міток

Даний алгоритм є прикладом ідеї “обміну повідомлень” (англ. message passing) між сусідніми вершинами графа, яка застосовується в графових нейронних мережах. Утім даний алгоритм передає значення міток і ніяким чином не враховує інші характеристики вершин графа.

РОЗДІЛ 4. ГРАФОВІ НЕЙРОННІ МЕРЕЖІ

4.1. Загальний принцип роботи графових нейронних мереж

Основна ідея графових нейронних мереж (ГНМ) полягає в ітеративному оновленні представлень вершин на основі ознак їх сусідів та їх власних ознак. Опишемо загальний принцип роботи графових нейронних мереж. Кожен шар ГНМ використовує [10]:

- Функцію агрегації A, яка агрегує ознаки сусідів для кожної вершини графа. Часто кажуть, що функція агреграції приймає на вхід повідомлення, які збираються з сусідніх вершин (message passing)

- Функцію об'єднання C, яка оновлює представлення вершини суміщуючи агреговані ознаки сусідів та поточне представлення вершини

У якості простих прикладів для функцій A і C можуть виступати функції середнього та суми.

Математично загальний принцип роботи шару ГНМ з K шарів можна подати наступним чином:

$$\alpha_v^k = A\{H_u^{k-1} \mid u \in N(v)\},$$
$$H_v^k = C\{H_v^{k-1}, \alpha_v^k\},$$

де $N(v)$ — множина сусідів вершини v, $k = 1..K$, H^k — представлення вершини.

Після отримання представлень вершин H^K вони можуть бути використані для різних задач. Наприклад для задачі класифікації клас вершини v може бути визначений за допомогою додаткового лінійного шару, до якого застосовується нормована експоненційна функція (поширена назва — softmax):

$$\hat{y}_v = softmax(WH_v^K + B).$$

Для набору вершин з мітками ГНМ може бути натренована мінімізуючи наступну функцію втрат:

$$L = \frac{1}{n}\sum_{i=1}^{n} \gamma(\hat{y}_i, y_i),$$

де y_i мітка для вершини i, n — кількість вершин, γ — класична функція втрат, наприклад перехресна ентропія.

4.2. Згорткові графові мережі

Згорткові графові мережі (ЗГМ) є найпопулярнішою архітектурою завдяки її простоті та ефективності в різноманітних задачах. Шари ЗГМ оновлюють представлення вершин наступним чином:

$$H^{k+1} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^k W^k) \text{ (1)},$$

де, σ — функція активації, $\tilde{A} = A + I$ — матриця суміжності заданого графа з петлями, що дозволяє врахувати ознаки вершини для якої оновлюється представлення; I — одинична матриця, що додає петлі; $\tilde{D}$ — діагональна матриця елементи, якої визначаються наступним чином:

$$\tilde{D}_{ii} = \sum_j A_{ij}.$$

W^k — матриця ваг, яка тренується під час оптимізації.

Розпишемо рівність (1) і покажемо функції агрегації та об'єднання, що використовуються ЗГМ. Для вершини i, рівність може бути записана наступним чином:

$$H_i^k = \sigma\left(\sum_{j\in N(i)\cup i} \frac{\tilde{A}_{ij}}{\sqrt{\tilde{D}_{ii}\tilde{D}_{jj}}} H_j^{k-1} W^k\right),$$

$$H_i^k = \sigma\left(\sum_{j\in N(i)} \frac{A_{ij}}{\sqrt{\tilde{D}_{ii}\tilde{D}_{jj}}} H_j^{k-1} W^k + \frac{1}{\tilde{D}_{ii}} H_i^{k-1} W^k\right) \text{ (2)},$$

З рівності (2) можна побачити, що функцією агрегації виступає зважене середнє представлень сусідніх вершин. Вага сусіда j дорівнює A_{ij} і нормалізується степенями вершин i та j — $\tilde{D}_{ii}$ та $\tilde{D}_{jj}$ відповідно. Функцією об'єднання виступає сума агрегованих повідомлень та представлення поточної вершини, яке нормалізуєтья її степенем $\tilde{D}_{ii}$.

У шарі ЗГМ обмін повідомленнями відбувається між сусідніми вершинами. Тобто якщо нейронна мережа складається з одного шару $K = 1$, то радіус поширення повідомлень дорівнює одиниці. Для графа зображеного на рисунку 4.1 при $K = 1$ обмін повідомленнями відбудеться лише між сусідніми вершинами: A і B, B і C, B і D. При $K = 2$ обмін повідомленями відбувається між усіма вершинами зображеними на рисунку 4.1, зокрема A і C, A і D.

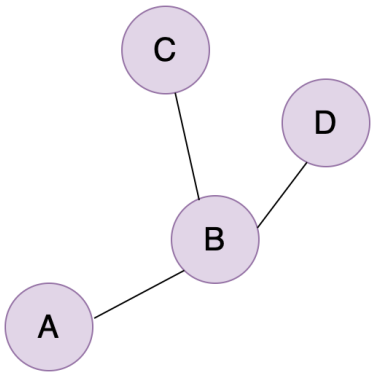


Рисунок 4.1

Розглянемо приклад моделі, яка застосовується для класифікації публікацій з набору даних CORA. Модель класифікатора складаєтья з шару передобробки, двох шарів ЗГМ, шару постобробки та шару лоджітів (значення якого є результатом класифікації). При створенні модель отримує інформацію про граф публікацій. Для прогнозування класу публікації модель отримує список індексів відповідних вершин. Схематично модель зображено на рисунку 4.2.

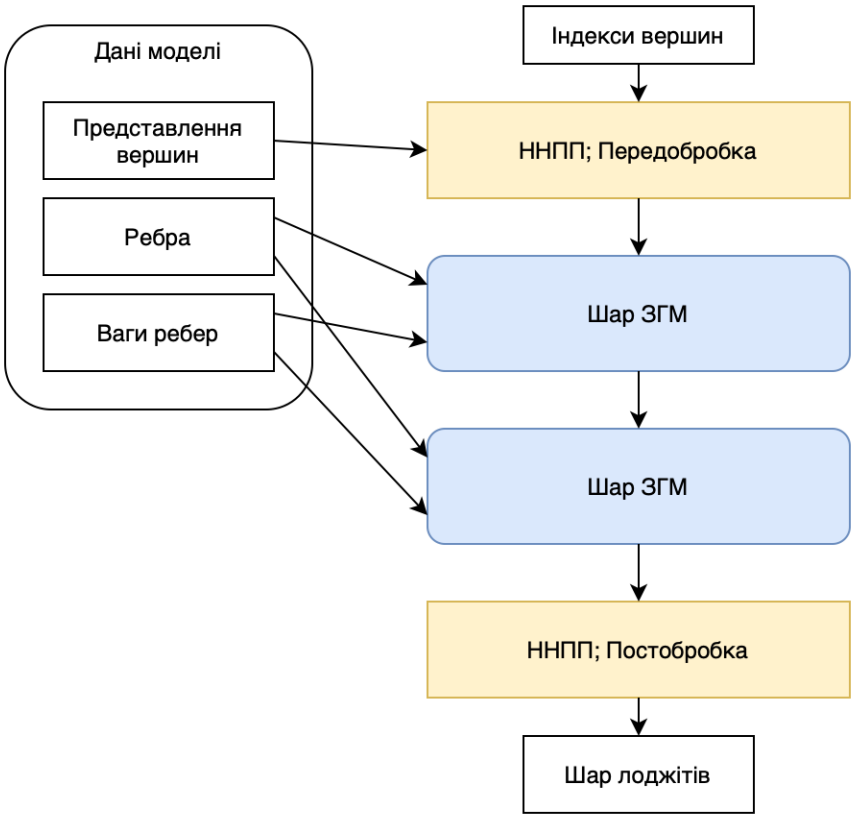


Рисунок 4.2. Класифікатор ЗГМ

Шар моделі ЗГМ отримує в якості вхідних даних представлення вершин, ребра та їх ваги (прим. у наборі даних CORA ваги дорівнюють 1), після чого виконує наступні дії:

- З представлення вершин утворюються повідомлення за допомогою нейронної мережі прямого поширення (НМПП; англ. feedforward neural network).
- Повідомлення агрегуються з урахуванням ваг ребер. Модель може використовувати одну з наступних функцій: середнє, сума, максимальне значення.
- Представлення вершин і агреговані повідомлення об’єднуються за допомогою однієї з функцій: сума, конкатенація після чого до результату застосовуєтья НМПП або пропускаються через вентильний рекурентний вузол (англ. gated recurrent unit; GRU).

Модель з даною архітектурою досягає точності класифікації понад 80% [12].

4.3. Графові мережі з увагою

У ЗГМ для вузла i важливість сусіда j визначається вагою ребра A_{ij} (нормалізованого за ступенем вершини). Однак, на практиці ваги ребер часто не відображають справжній рівень зв'язку двох вершин. Спробою вирішити цю проблему є графові мережі з увагою (ГМУ; англ. graph attention networks, GAT) [13]. У ГМУ важливість кожного сусіда визначається за допомогою механізму уваги. Механізм уваги набув широкого застосування у задачах з обробки природної мови, наприклад переклад, та комп'ютерного зору, наприклад визначення заголовку для зображення (англ. image captioning).

Для того, щоб оновити представлення вершин H^k, шар ГМУ використовує попереднє представлення H^{k-1}, матрицю ваг W (прим. матриця W єдина і застосовується до кожного представлення вершини) і функцію уваги α, яка застосовується до пари вершин наступним чином:

$$e_{ij} = \alpha(WH_i^{k-1}, WH_j^{k-1}),$$

e_{ij} — сила зв'язку між вершинами i та j. Хоча функція уваги і може бути застосована до кожної пари вершин графу, такий підхід ігнорує структуру графу і на практиці функція уваги застосовується лише до суміжних вершин. Для того, щоб зробити значення функції порівнюваними для різних вершин їх нормалізують функцією $softmax$:

$$\alpha_{ij} = softmax_j(\{e_{ij}\}) = \frac{exp(e_{ij})}{\sum_{l \in N(i)} exp(e_{il})}.$$

Звідси можна побачити, що для вершини i визначає поліноміальний розподіл серед сусідніх вершин, що можна розглядати як імовірність переїоду від вершини i до кожного з сусідів.

Після розрахунку значень α_{ij} оновлені представлення розраховуються за формулою (σ — функція активації):

$$H_i^k = \sigma\left(\sum_{j \in N(i)} \alpha_{ij} W H_j^{k-1}\right) \quad (1).$$

На практиці замість використання одного механізму уваги, може використовуватися декілька (англ. multi-head attention), кожен з яких використовує власну функцію схожості вершин графа. Для кожного механізму уваги розраховується власне представлення вершин за рівністю (1), після чого отримані представлення можуть бути сконкатеновані:

$$H_i^k = \Big\|_{t=1}^{T} \sigma\left(\sum_{j \in N(i)} \alpha_{ij}^t W^t H_j^{k-1}\right),$$

де T — кількість механізмів уваги, α_{ij}^t та W^t — значення функції уваги та матриці ваг для t-го механізму уваги відповідно. Замість операції конкатенації можуть бути застосовані інші операції, наприклад середнє:

$$H_i^k = \sigma\left(\frac{1}{T}\sum_{t=1}^{T}\sum_{j \in N(i)} \alpha_{ij}^t W^t H_j^{k-1}\right).$$

Розглянемо приклад моделі, яка застосовується для класифікації публікацій з набору даних CORA. Модель класифікатора складається з лінійного шару, трьох шарів ГМУ та шару лоджітів. При створенні модель отримує інформацію про граф публікацій. Схематично модель зображено на рисунку 4.3.

Шар моделі ГМУ отримує в якості вхідних даних представлення вершин та ребра, після чого виконує наступні дії:

– Обраховує значення функції уваги для вершин графа для кожної функції уваги.
– Нормалізує отримані значення використовуючи $softmax$.

– Агрегує нормалізовані значення разом із представленнями вершин.
– Конкатенує агреговані результати від кожної з функцій уваги.

Модель з даною архітектурою здатна досягти точності класифікації близько 83% [14].

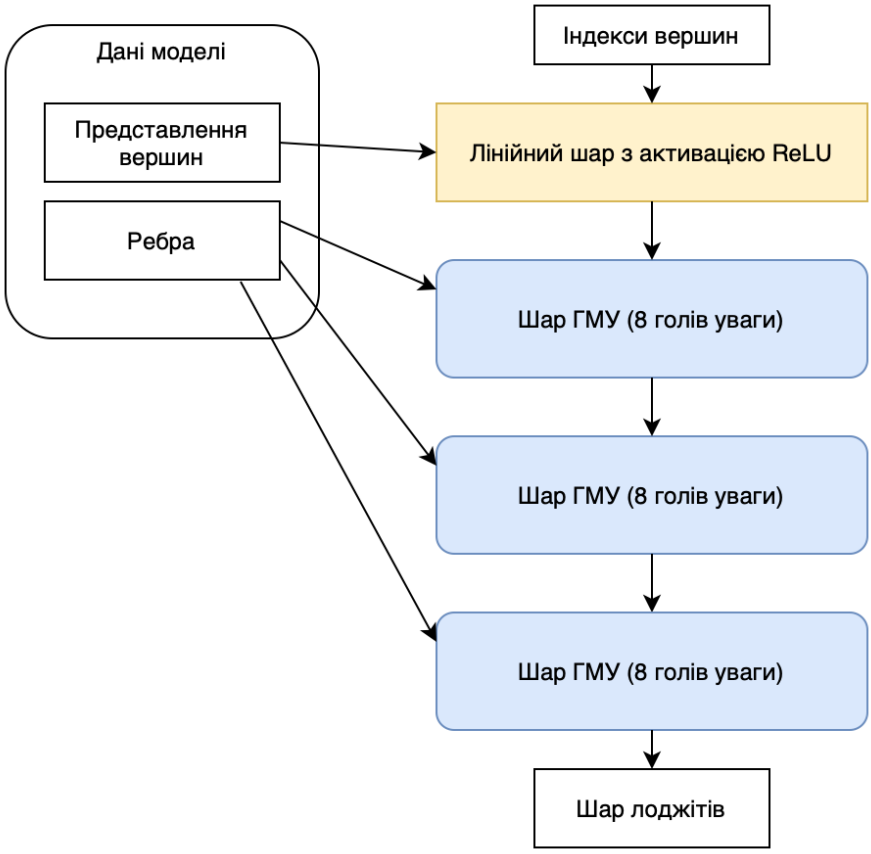


Рисунок 4.3. Класифікатор ГМУ

4.4. Нейронні мережі обміну повідомленнями

Нейронні мережі обміну повідомленнями (НМОП; англ. message passing neural networks) є одним з видів графових нейронних мереж. Для простоти опишемо НМОП, що працюють на неорієнтованих графах G з ознаками вершин x_i та ознаками ребер e_{ij}. Цей опис можна легко розширити до орієнтованих графів.

Пряме поширення (англ. forward pass) у НМОП складається з двох фаз [15]:

– Фаза обміну повідомленнями

– Фаза зчитування повідомлень

Фаза обміну повідомленнями має K кроків і визначається функцією повідомлень M_k та функцією оновлення вершин U_k. Під час фази обмну повідомленнями, приховані стани H^k для кожної вершини графа оновлюються на основі повідомлень m_i^k наступним чином:

$$m_i^k = \sum_{w \in N(i)} M_k(H_i^{k-1}, H_j^{k-1}, e_{ij}) \quad (1)$$

$$H_i^k = U_k(H_i^{k-1}, m_i^k) \quad (2)$$

де $N(i)$ — множина сусідів вершини v у графі.

Під час фази зчитування розраховується вектор ознак для графа використовуючи функцію зчитування R визначену наступним чином:

$$\hat{y} = R(\{H_i^K \mid i \in G\})$$

Функції обміну повідомленнями M_k, функції оновлення вершин U_k і функція зчитування R – все це диференційовані функції, які навчаються. R працює з набором станів вершин і має бути інваріантною до перестановок станів вершин, щоб НМОП була інваріантною щодо ізоморфізму графа. Дана модель може бути використана і для ребер графа: для кожного реба визначається прихований стан $H_{e_{ij}}^k$ і оновлюється аналогічно до рівнянь (1) та (2).

Розглянемо приклад моделі, яка застосовується для класифікації молекул за проникністю гематоенцефалічний бар'єру. Дана модель працює з молекулами, які розглядаються як граф (рисунок 4.4) з атомів — вершин та хімічних зв’язків — ребер.

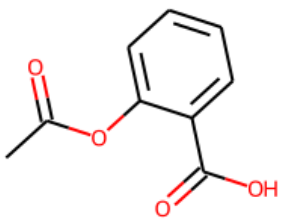


Рисунок 4.4. Граф молекули

Атоми та хімічні зв’язки мають вектори ознак, зв’язки між атомами представлені списком ребер. Модель приймає на вхід вибірки з молекул, що представляють собою граф і опрацьовуються шаром обміну повідмоленнями та індикаторами молекул, які передаються до шару зчитування повідомлень разом з результатом роботи шару обміну повідомлень. Схематично модель зображено на рисунку 4.5.

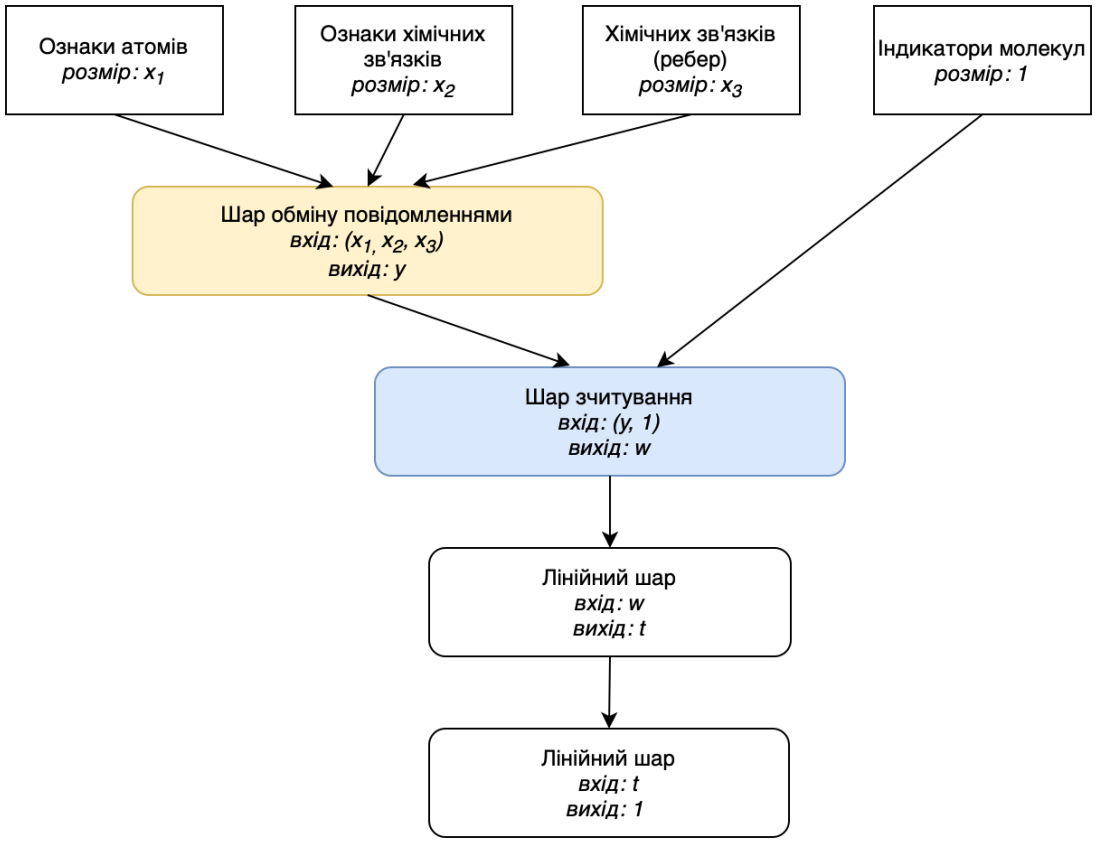


Рисунок 4.5. Класифікатор НМОП

Шар обміну повідомленнями приймає параметр кількості кроків K, який визначає радіус пошерення повідомлення від вершини.

Крок k у шарі обміну повідомленнями складається з двох частин:

– Передача ознак між сусідніми вершинами і отримання агрегованих ознак та прихованого стану вершини на момент k.
– Використання вентильного рекурентного вузла (англ. gated recurrent unit; GRU) який оновлює агреровані ознаки та прихований стан

вершини на момент часу t, використовуючи попередні стани вершини на моменти часу $1..k-1$.

Шар зчитування, отримавши на вхід результат роботи шару обміну повідомленнями та індикатори молекул, виконує наступні кроки:

- Агреговані ознаки і стани вершин розбиваються на підграфи (відповідно до кожної молекул) за допомогою індикаторів молекул.
- Граф кожної молекули доповнюється до графу з найбільшою кількістю вершин.
- Для графів створюється маска, роль якої прибрати вплив від доповнених вершин.
- Отримується вкладення, шляхом застосування трансформера і субдискретизації.

Модель з такою архітектурою досягає значення метрики AUC понад 0.92 [16].

РОЗДІЛ 5. ПОБУДОВА МОДЕЛЕЙ ТА ОЦІНКА РЕЗУЛЬТАТІВ

5.1. Опис набору даних

Для оцінки результатів моделі обрано набір даних ACM. Набір даних представляє собою мережу цитування, у якій наявні наступні типи вершин:

- “публікація”
- “автор публікації”

Вершини типу “публікація” мають вектор характеристик, що представляє список слів, вживаних у публікації. Публікації поділяються на 3 категорії (класи): добування даних (англ. data mining), бази даних і комунікації. Вершини авторів не мають векторів характеристих та міток класів. Схематично набір даних зображено на рисунку 5.1.

Вершини пов’язані між собою наступними зв’язками: “публікація – публікація”, “публікація – автор”. Тип зв’язку “публікація – автор” буде використано для демонстрації роботи графових мереж з гетерогенними

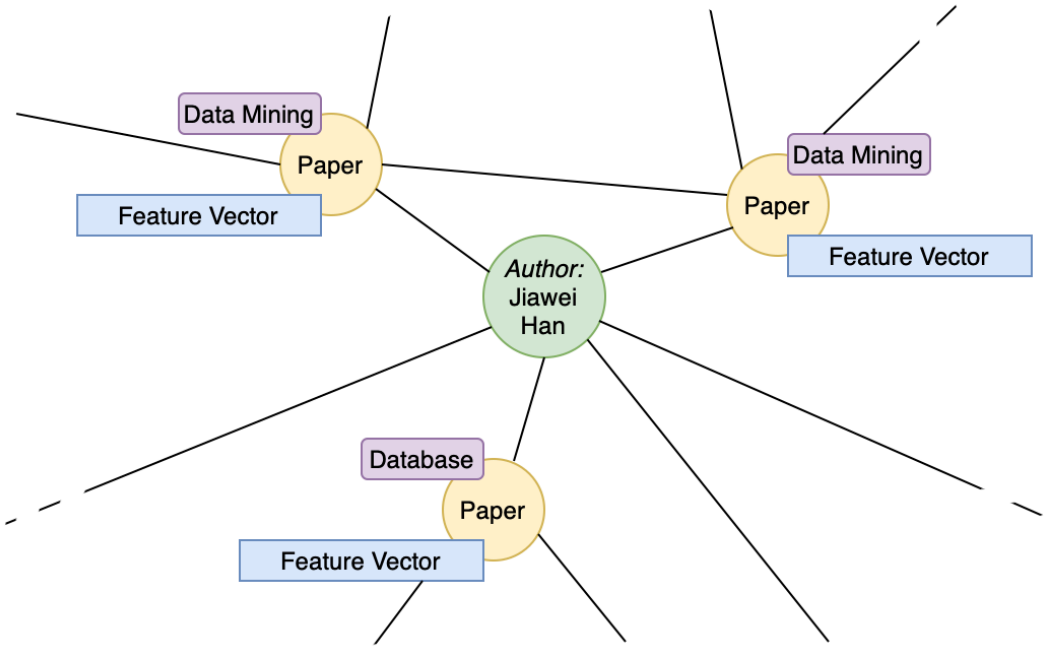


Рисунок 5.1. Схематичне зображення набору даних ACM

графами. Задача для побудованих моделей ГНМ полягає в класифікації вершин типу “публікація”.

У випадку гомогенного графу — вид графу, що має вершини та ребра одного типу, кількість вершин типу “публікація” складає 3360, кількість неорієнтованих ребер — 9579. Класи вершин розподілені наступними чином:

– добування даних — 876 публікацій
– бази даних — 1724 публікації
– комунікації — 760 публікацій

У випадку гетерогенного графу — вид графу, що має вершини та ребра різних типів, кількість вершин типу “публікація” складає 3442, кількість вершин типу “автор” — 2064, кількість ребер “публікація – публікація” — 8753, “публікація – автор” — 8304. Класи вершин типу публікація розподілені наступним чином:

– добування даних — 903 публікацій
– бази даних — 1820 публікації
– комунікації — 719 публікацій

5.2. Розрахунок опорних результатів

Спочатку застосуємо моделі node2vec та НМПП на гомогенному наборі даних ACM для отримання опорних результатів (англ. baseline), з яким можна порівняти результати роботи ЗГМ та оцінити її ефективність.

Застосуємо алгоритм node2vec для отримання представлень вершин типу “публікація” та застосуємо до них НМПП. Результати точності класифікації, влучності (англ. precision), повноти (англ. recall) та F1 на тестовій вибірці (20% від набору даних) наведено у таблиці 5.1.

Таблиця 5.1. Опорний рівень результатів

Клас	Влучність	Повнота	F1	К-сть тестових зразків
Добування даних	0.89	0.67	0.77	185
Бази даних	0.82	0.95	0.88	346
Комунікації	0.92	0.84	0.88	141
Точність класифікації: 85%				

5.3. Побудова і оцінка класифікатора ЗГМ

Класифікатор ЗГМ здійснює модифікацію представлення вершин, використовуючи матрицю суміжності, що включає петлі і нормована за степенями вершин $A_{norm} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$. Дана матриця використовується для створення об’єкту моделі.

Згорткові шари моделі виконують агрегацію та комбінацію повідомлень від вершин графа за допомогою їх множення на матрицю A_{norm}, після додатково застосовуються повнозв’язні шари з функцією активації PReLU — параметричний різновид протікаючої ReLU з “тренованим” параметром α. Схематично шар ЗГМ зображено на рисунку 5.2.

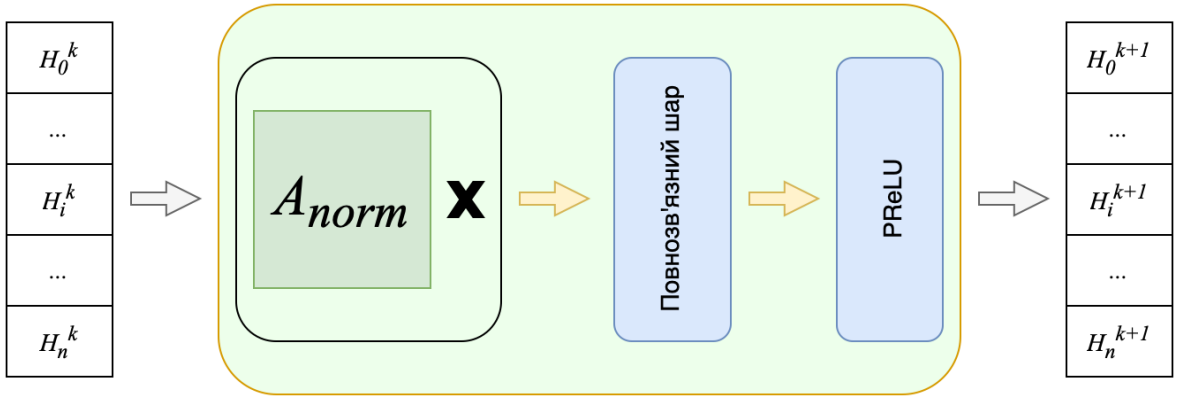


Рисунок 5.2. Шар ЗГМ

Побудована модель використовує:

– Повнозв’язний шар з функцією активації PReLU з розмірністю 512 нейронів для передобробки.
– Три згорткові шари, що використовують пари повнозв’язних шарів розмірностей 384 та 256, 256 та 192, 192 та 96.
– Повнозв’язний шар з функцією активації PReLU з розмірністю 64 для постобробки.

Для тренування використовується алгоритм оптимізації Adam. Результати класифікації моделі наведено у таблиці 5.2.

Таблиця 5.2. Результати моделі ЗГМ

Клас	Влучність	Повнота	F1	К-сть тестових зразків
Добування даних	0.89	0.94	0.91	185
Бази даних	0.95	0.94	0.94	346
Комунікації	0.99	0.96	0.97	141
Точність класифікації: 94%				

Отримані результати первищують опорні, а отже свідчать про те, що модель вдало виконала задачу.

5.4. Побудова і оцінка класифікатора ГМУ

Для класифікатора ГМУ на відміну від ЗГМ, де використовується матриця замість матриці A_{norm}, будемо будувати матрицю для обміну повідомленнями A'_{α} за допомогою функції уваги. Для цього обирається функція α, яка обраховує значення матриці для сусідніх вершин на основі їх представлень до яких застосовано повнозв'язний шар. Схематично шар ГМУ зображено на рисунку 5.3.

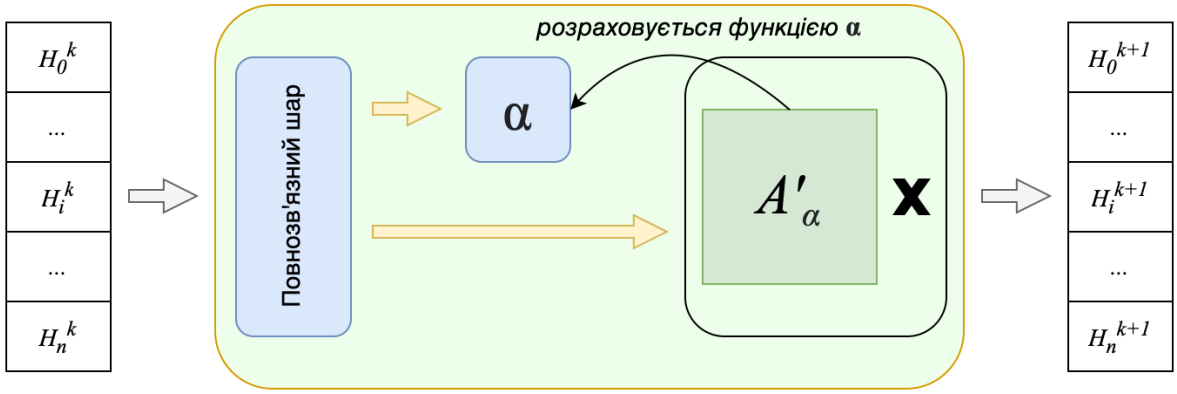


Рисунок 5.3. Шар ГМУ

Використаємо декілька функцій α: PReLU, swish (або $x \cdot sigmoid(x)$) та tanh — популярні функції активації, які добре підходять для охоплення складних нелінійних залежностей. Архітектура моделі аналогічна до архітектури класифікатора ЗГМ, замість шарів згортки використовуються шари ГМУ з зазначеними функціями α.

Результати класифікації моделі наведено у таблиці 5.3.

Таблиця 5.3. Результати моделі ГМУ

Клас	Влучність	Повнота	F1	К-сть тестових зразків
Добування даних	0.87	0.94	0.91	185
Бази даних	0.95	0.94	0.94	346
Комунікації	0.99	0.96	0.97	141

Точність класифікації: 93%

Отримані результати перевищують опорні, отже можна вважати, що модель вдало виконала задачу.

Слід зазначити, що результати моделі дещо гірші, ніж у аналогічної моделі ЗГМ. З чого можна зробити висновок, що модель ЗГМ, яка використовує нормовану матрицю A_{norm}, є оптимальнішою для даної задачі.

5.5. Побудова і оцінка гетерогенного класифікатора ЗГМ

Гетерогенний набір даних має декілька матриць суміжності, зокрема матриці суміжності вершин різного типу, наприклад матриця A^{pa}, елементи якої розначають ребра “публікація – автор”. Пронормуємо дану матрицю суміжності за степенями вершин типу “публікація” та “автор” за формулою:

$$A^{pa}_{norm} = (D^p)^{-\frac{1}{2}} A^{pa} (D^a)^{-\frac{1}{2}},$$

де D^p та D^a — діагональні матриці степенів вершин типу “публікація” та “автор” відповідно. Отриману матрицю A^{pa}_{norm} використовуємо для передачі повідомлень від вершин типу “автор” до вершин типу “публікація”. Для передачі повідомлень у зворотньому напрямі використовуємо матрицю $A^{ap}_{norm} = (A^{pa}_{norm})^T$.

Оскільки в наборі даних, що розглядається автори сполучені ребрами тільки з публікаціями, то для передачі повідомлень від вершини “публікація” до вершини “автор” застосовується згортковий шар ідентичний до шару зображеного на рисунку 5.2, де $A_{norm} = A^{ap}_{norm}$.

Шар, що передає повідомлення до вершин типу “публікація” від сусідніх вершин типу “публікація” та “автор” множить відповідні представлення вершин на матриці A^{pp}_{norm} (прим. до даної матриці додаються петлі) та A^{pa}_{norm} застосовує до них повнозв’язні шари (однакової

розмірності), отримані вектори сумуються і до них застосовується функція активації PReLU. Схематично шар гетерогенний згортковий шар зображено на рисунку 5.4.

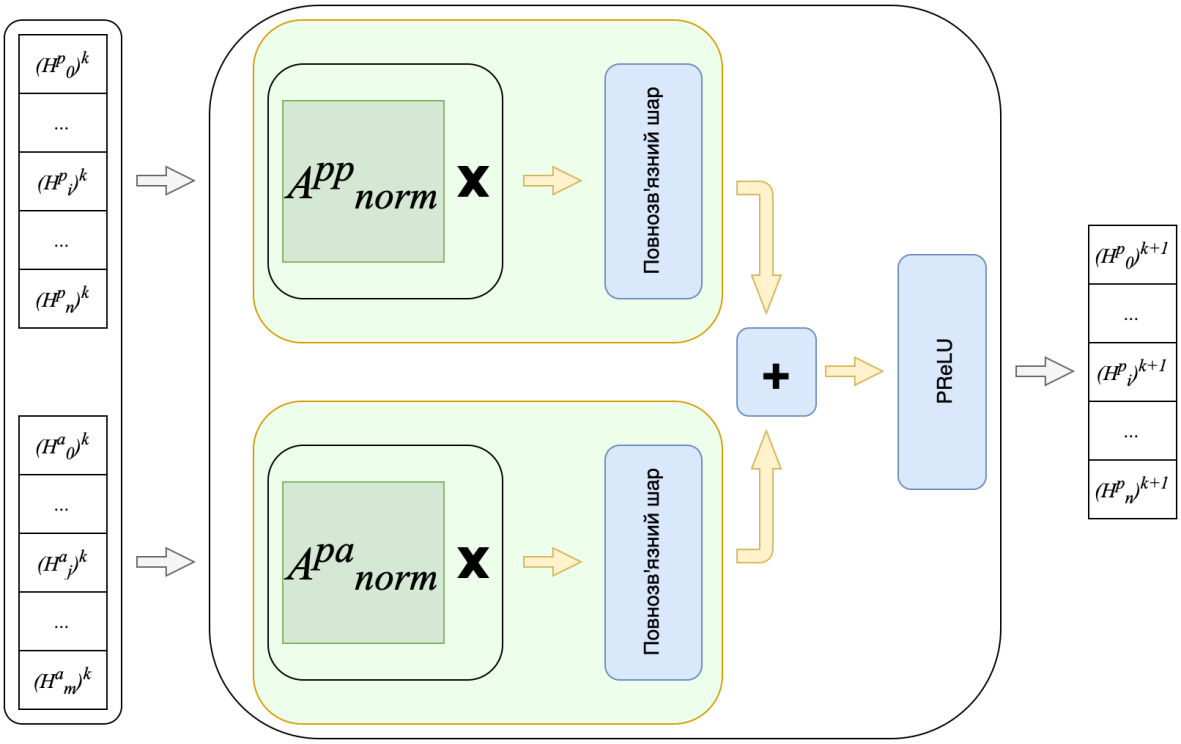


Рисунок 5.4. Шар гетерогенної ЗГМ

Слід зазначити, що оскільки в наборі даних немає ознак для вершин типу “автор” (H_i^a), для їх отримання застосовується шар вкладень.

Побудована модель використовує:

1) повнозв’язний шар для передоброки ознак вершин типу “публікація” розмірності 128 нейронів, шар вкладення для ознак вершин типу “автор” розмірності 128;
2) гетерогенні згорткові шари для передачі повідомлень
   I) шар передачі повідомлень “публікація – публікація”, “автор – публікація”, що використовують по два повнозв’язних шари розмірності 128;
   II) шар передачі повідомлень “публікація – автор”, що використовує повнозв’язні шари розмірності 128 та 96;

III) шар передачі повідомлень “публікація – публікація”, “автор – публікація”, що використовують по одному шару розмірності 96;

3) повнозв’язний шар для постобробки розмірності 64 з функцією активації PReLU.

Результати класифікації моделі наведено в таблиці 5.4.

Таблиця 5.4. Результати гетерогенної моделі ЗГМ

Клас	Влучність	Повнота	F1	К-сть тестових зразків
Добування даних	0.9	0.91	0.9	172
Бази даних	0.96	0.95	0.95	363
Комунікації	0.99	0.99	0.99	154
Точність класифікації: 95%				

Отримані результати перевищують опорні та покращують результати гомогенної моделі ЗГМ, а отже модель вдало виконала задачу.

5.6. Побудова і оцінка гетерогенного класифікатора ГМУ

Гетерогенна модель ГМУ будується аналогічно до моделі ЗГМ описаної в попередньому розділі. Замість матриць A_{norm}^{pp} та A_{norm}^{pa} модель обчислює матриці $(A'_\alpha)^{pp}$ та $(A'_\alpha)^{pa}$ за допомогою ознак вершин між, якими відправляється повідомлення.

Шар, що розраховує $(A'_\alpha)^{pp} = A'_\alpha$, ідентичний шару зображеному на рисунку 5.3. Шар, який розраховує матрицю $(A'_\alpha)^{pa}$, на відміну від гомогенного випадку використовує додаткову матрицю ознак. Його роботу схематично зображено на рисунку 5.5.

Результати класифікації моделі наведено у таблиці 5.5.

Таблиця 5.5. Результати гетерогенної моделі ГМУ

Клас	Влучність	Повнота	F1	К-сть тестових зразків
Добування даних	0.91	0.87	0.89	172

Бази даних	0.92	0.95	0.94	363
Комунікації	0.99	0.97	0.98	154
Точність класифікації: 93%				

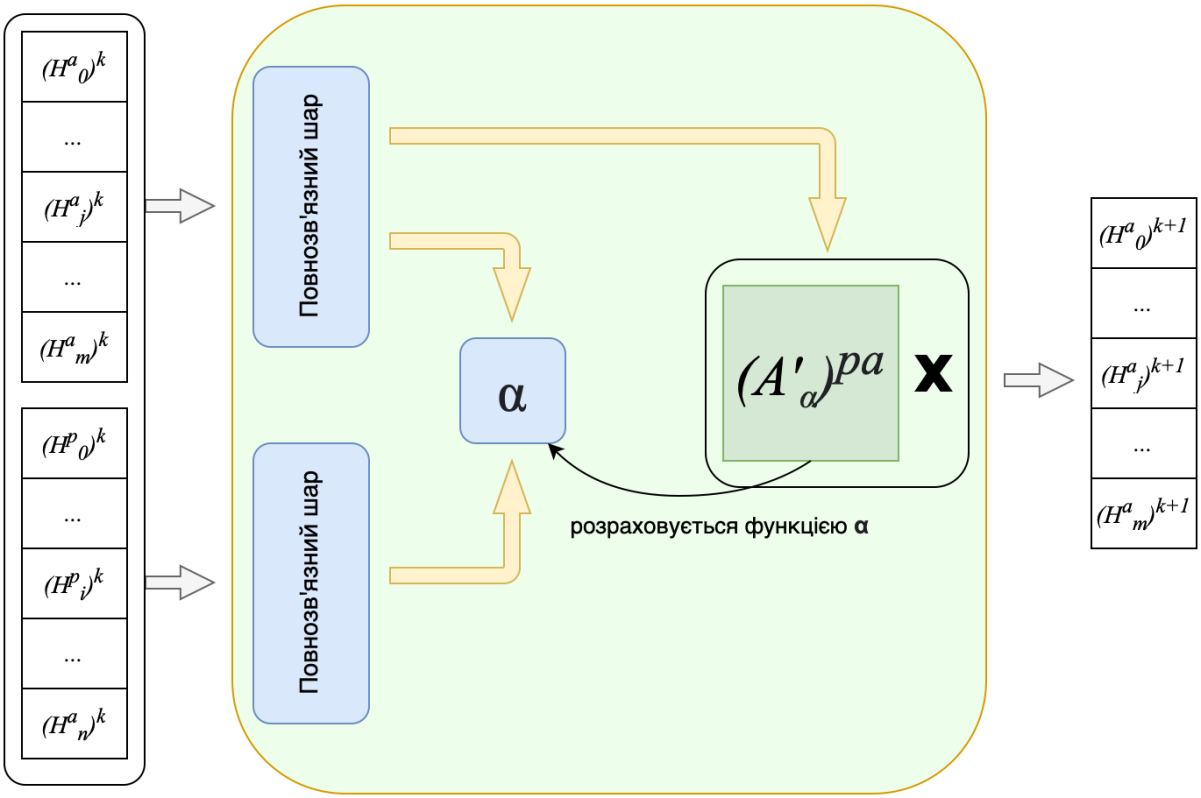


Рисунок 5.5. Шар для передачі повідомлень “автор – публікація”

Отримані результати перевищують опорні, але гірші, ніж результати гетерогенної моделі ЗГМ, що свідчить про оптимальність ЗГМ для даної задачі.

ВИСНОВКИ

У роботі описано базові принципи побудови та тренування нейронних мереж на прикладі багатошарового перцептрону, які застовуються зокрема й при тренуванні графових нейронних мереж.

Розглянуто проблему вивчення представлень вершин графу та алгоритм машиного навчання node2vec, який використовується для розв’язання цієї проблеми. Реалізація цього алгоритму була застосована для отримання опорних результатів, що слугували для оцінки моделей побудованих під час експерименту.

Розглянути загальний принцип роботи графових нейронних мереж, для ілюстрації ідеї “обміну повідомленнями” наведено опис алгоритму розповсюдження міток. Розглянуто архітектуру поширених моделей графових нейронних мереж: згорткові графові мережі, графові мережі з увагою, нейронні мережі обміну повідомленнями та наведено приклади їх роботи.

Проведено експеримент, що полягає у вирішенні задачі класифікації вершин гомогенного та гетерогенного графів. У результаті екперименту для обох типів графів було побудовано моделі згорткової графової мережі та графової мережі з увагою. Модель згорткової графової мережі досягла точності класифікації 94% для гомогенного та 95% для гетерогенного графів. Модель графової мережі з увагою, попри більш складну реалізацію, досягла точності 93% на відповідних графах. Обидві моделі впоралися з задачею, модель згорткової графової мережі виявилась оптимальною на розглянутому наборі даних.

Дана робота виступає теоретичним підґрунтям для вирішення проблеми класифікації вершин графу та пропонує сучасні та ефективні методи її розв’язання за допомогою графових нейронних мереж.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Cybenko G. Approximation by Superpositions of a Sigmoidal function // Mathematics of Control, Signals and Systems 2. – Нью-Йорк: Springer International, 1989. – С. 303–314.
2. Snyman J. Practical Mathematical Optimization / J. Snyman, D. Wilke. – Нью-Йорк: Springer International, 2005. – ISBN 978-0-387-24348-1.
3. Kingma D. Adam: A Method for Stochastic Optimization [Електронний ресурс] / D. Kingma, J. Ba. – arXiv, 2014. – Режим доступу до препринту: https://arxiv.org/abs/1412.6980
4. Bengio Y. Representation Learning: A Review and New Perspectives [Електронний ресурс] / Y. Bengio, A. Courville, P. Vincent. – arXiv, 2013 . – Режим доступу до препринту: https://arxiv.org/abs/1206.5538
5. Alman J. A Refined Laser Method and Faster Matrix Multiplication [Електронний ресурс] / J. Alman, V. Williams. – arXiv, 2020. – Режим доступу до препринту: https://arxiv.org/abs/2010.05846
6. Masuda N. Random walks and diffusion on networks [Електронний ресурс] / N. Masuda, M. Porter, R. Lambriotte. – 2017 . – Режим доступу до журналу: https://www.sciencedirect.com/journal/physics-reports/vol/716
7. Grover A. node2vec: Scalable Feature Learning for Networks [Електронний ресурс] / A. Grover, J. Leskovec. – arXiv, 2016. – Режим доступу до препринту: https://arxiv.org/abs/1607.00653
8. Node classification with Node2Vec using Stellargraph components. [Електронний ресурс]. – 2020. – Режим доступу до статті: https://stellargraph.readthedocs.io/en/stable/demos/node-classification/keras-node2vec-node-classification.html
9. Zhu X. Learning From Labeled and Unlabeled Data With Label Propagation [Електронний ресурс] / X. Zhu, Z. Ghahramani. – 2002. – Режим доступу до препринту: http://mlg.eng.cam.ac.uk/zoubin/papers/CMU-CALD-02-107.pdf

10. Xu K. How Powerful are Graph Neural Networks [Електронний ресурс] / K. Xu, W. Hu, J. Leskovec, S. Jagelka. – arXiv, 2018. – Режим доступу до препринту: https://arxiv.org/abs/1810.00826
11. Kipf T. Semi-Supervised Classification with Graph Convolutional Networks [Електронний ресурс] / T. Kipf, M. Welling. – arXiv, 2016. – Режим доступу до препринту: https://arxiv.org/abs/1609.02907
12. Node Classification with Graph Neural Networks [Електронний ресурс] / K. Salama. – 2021. – Режим доступу до статті: https://keras.io/examples/graph/gnn_citations/
13. Graph Attention Networks [Електронний ресурс] / [P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio]. – arXiv, 2017. – Режим доступу до препринту: https://arxiv.org/abs/1710.10903
14. Graph attention network (GAT) for node classification [Електронний ресурс] / A. Kensert. – 2021. – Режим доступу до статті: https://keras.io/examples/graph/gat_node_classification/
15. Neural Message Passing for Quantum Chemistry [Електронний ресурс] / [J. Gilmer, S. Schoenholz, P. Riley, O. Vinyals, G. Dahl]. – arXiv, 2017 . – Режим доступу до препринту: https://arxiv.org/abs/1704.01212
16. Message-passing neural network (MPNN) for molecular property prediction [Електронний ресурс] / A. Kensert. – 2021. – Режим доступу до статті: https://keras.io/examples/graph/mpnn-molecular-graphs/