НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

«До захисту допущено»
В.о. завідувача кафедри
_____ Микола ГРАЙВОРОНСЬКИЙ
«__» _____ 2021 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Математичні методи моделювання, розпізнавання образів та безпеки даних»
спеціальності: 113 «Прикладна математика»

на тему: «Застосування дифузіїних карт до розв'язання задач напівавтоматичного навчання»

Виконав: здобувач вищої освіти **IV** курсу, групи ФІ-72
Попригo Леонід Леонідович

Керівник: звання, степінь, посада Литвинова Т.В. _____

Консультант: с.н.с Семенова Є.В. _____

Рецензент: звання, степінь, посада Прізвище І.П. _____

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

Здобувач вищої освіти _____

Київ — 2021

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти — перший (бакалаврський)
Спеціальність (освітня програма) — 113 Прикладна математика,
Освітньо-професійна програма «Математичні методи моделювання, розпізнавання образів та безпеки даних»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Микола ГРАЙВОРОНСЬКИЙ
«__» _____ 2021 р.

ЗАВДАННЯ
на дипломну роботу здобувача вищої освіти

Попригo Леонід Леонідович

1. Тема роботи: Застосування дифузіїних карт до розв'язання задач напівавтоматичного навчання,

керівник роботи: звання, степінь, посада Литвинова Т.В.,

затверджені наказом по університету №__ від «__» _____ 2021р.

3. Вихідні дані до роботи: *літературні джерела за тематикою роботи, біологічний та іграшковий("два місяці") набори даних, реалізовані існуючі методи розв'язання задач напівавтоматичного навчання*

4. Зміст роботи:

– *розробити метод, заснований на дифузійних картах, для розв'язання задач напівавтоматичного навчання;*

– *навести числовий алгоритми для здійснення експериментальної перевірки результатів;*

– *комп'ютерна реалізація алгоритму, що використовувався у дослідженні;*

– *порівняти та результати з існуючими алгоритмами та проаналізувати їх.*

5. Перелік ілюстративного матеріалу: *Презентація доповіді*

6. Дата видачі завдання: 10 вересня 2020 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання	Примітка
1	Узгодження теми роботи із науковим керівником	01-15 вересня 2020 р.	Виконано
2	Огляд опублікованих джерел за тематикою дослідження	Вересень-жовтень 2020 р.	Виконано
3	Написання літературного огляду	01 листопада - 25 грудня 2020 р.	Виконано
4	Формулювання і доведення поставленої задачі	25 грудня - 1 лютого 2021 р.	Виконано
5	Програмна реалізація класифікатору	1 лютого - 19 березня 2021 р.	Виконано
6	Написання першої частини дипломної роботи	19 березня - 19 квітня 2021 р.	Виконано
7	Написання другої частини дипломної роботи	19 квітня - 1 травня 2021 р.	Виконано
8	Написання третьої частини дипломної роботи	1-10 травня 2021 р.	Виконано
9	Написання висновків	10-19 травня 2021 р.	Виконано
10	Підготовка тез	19-28 травня 2021 р.	Виконано
11	Подання роботи на рецензування	8 червня 2021 р.	Виконано
12	Підготовка доповіді та презентації	4-10 червня 2021 р.	Виконано
13	Подання роботи до захисту	17 червня 2021 р.	Виконано

Студент _____ Леонід ПОПРИГО

Керівник _____ Тетяна ЛИТВИНОВА

РЕФЕРАТ

Кваліфікаційна робота містить: 64 стор., 12 рисунки, 0 таблиць, 12 джерел посилань.

Завданням роботи є дослідження існуючих класифікаторів, що розв'язують задачу напівавтоматичного навчання на невеликій кількості маркованих даних та реалізація власного класифікатору за допомогою дифузійних карт.

Метою цієї роботи є правильне передбачення міток всіх немаркованих точок. В процесі навчання використовуються як марковані, так і немарковані точки. Ефективність алгоритму вимірюється коефіцієнтом помилок лише в немаркованих точках.

Об'єктом дослідження є методи розв'язання задач напівавтоматичного навчання.

Предметом дослідження є дифузійні карти графу Лапласіану.

Актуальність роботи зумовлюється тим, що на сьогодні все більше й більше зростає простір бітів, збільшуються обчислювальні потужності, тим не менш, дані, які необхідно оброблювати, потребують маркування. Часто це вимагає дорогої людської праці, тоді як немарковані дані отримати набагато простіше. Напівавтоматичне навчання використовує не тільки марковані, а й немарковані дані, що покращує оцінки якості класифікації.

Наукова новизна отриманих результатів полягає у застосуванні дифузійних карт до розробки нового методу розв'язання задачі напівавтоматичного навчання, що базуються на використанні розрідженого графу Лапласіану, як апроксимації оператору Лапласа-Бельтрамі.

Практичне значення результатів полягає у покращенні основних метрик оцінки якості класифікації в задачах машинного навчання, в котрих використовується невелика частина маркованих та велика частина

немаркованих даних.

В роботі були імплементовані існуючі методи розв'зяання задачі напівавтоматичного навчання, а також вдосконалено існуючий метод класифікації за рахунок оптимізації прорахунку розрідженого графу Лапласіану замість повного графу.

Практично перевірено результати класифікації на даних біологічної активності хімічних сполук під певну мішень, використовуючи розроблений метод напівавтоматичного навчання з використанням розрідженого графу Лапласіану.

КЛЮЧОВІ СЛОВА: НАПІВАВТОМАТИЧНЕ НАВЧАННЯ, МАШИННЕ НАВЧАННЯ, ПРОГНОЗУВАННЯ, ВІДТВОРЮЮЧЕ ЯДРО ГІЛЬБЕРТОВОГО ПРОСТОРУ, РЕГУЛЯРИЗАЦІЯ, ОПЕРАТОР ЛАПЛАСА-БЕЛЬТРАМІ, ХМАРНИЙ ГРАФ ЛАПЛАСІАН, СПЕКТРАЛЬНА ЗБІЖНІСТЬ.

ABSTRACT

The qualification work contains: 64 pages, 12 drawings, 0 tables, 12 names of bibliographic sources.

The task of the work is to study the existing classifiers, which solve the problem of semi-supervised learning on a small amount of labeled data and the implementation of its classifier with the help of diffusion maps.

The purpose of this work is to correctly predict the labels of all unlabeled data. Both labeled and unlabeled points are used in the learning process. The efficiency of the algorithm is measured by the error rate only at unlabeled points.

The object of research is methods of solving problems of semi-supervised learning.

The subject of the study is the diffusion maps of Count Laplacian.

The urgency of the work is since today more and more bit space is growing, computing power is increasing, however, the data that needs to be processed need labeling. This often requires expensive human labor, while unlabeled data is much easier to obtain. Semi-supervised learning uses not only labeled but also unlabeled data, which improves the assessment of the quality of classification.

The scientific novelty of the obtained results lies in the application of diffusion maps to the development of a new method for solving the problem of semi-supervised learning, based on the use of a sparse Laplacian graph as an approximation of the Laplace-Beltrami operator.

The practical significance of the results is to improve the basic metrics for assessing the quality of classification in machine learning tasks, which use a small part of the labeled and most of the unlabeled data.

Existing methods for solving the problem of semi-supervised learning have been implemented, as well as the existing method of classification has been improved by optimizing the calculation of the sparse Laplacian graph instead

of the complete graph.

The results of classification of biological activity of chemical compounds under a certain target are practically checked, using the developed method of semi-supervised learning using a sparse Laplacian graph. The comparative analysis of models is carried out, the received results are processed.

KEYWORDS: SEMI-SUPERVISED LEARNING, MACHINE LEARNING, PREDICTING, REPRODUCING KERNEL HILBERT SPACE, REGULARIZATION, LAPLACE-BELTRAMI OPERATOR, POINT CLOUD LAPLACIAN, SPECTRAL CONVERGENCE.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

RKHS - Reproducing kernel Hilbert space, відтворююче ядро в Гільбертовому просторі.

Label Propagation - метод поширення міток.

Label Spreading - метод розподілення міток.

Ridge regression - гребнева регресія.

RBF - Radial basis function, Радіальна базисна функція

SVM - Support-vector machine, Метод опорних векторів

kNN - k-nearest neighbors algorithm, Метод k-найближчих сусідів

ERM - Empirical risk minimization, Мінімізація емпіричного ризику

Mercer Kernel - ядро Мерсера

XGBoost - eXtreme Gradient Boosting, екстремальне градієнтне підсилювання.

ВСТУП

Кажуть, що проблема розуміння інтелекту є найбільшою проблемою науки сьогодні та проблемою цього століття. Останні кілька років спостерігається значна активність у створенні геометрично-вмотивованих підходів до аналізу даних та машинного навчання.

Актуальність дослідження. Маркування даних часто вимагає дорогої людської праці, тоді як непомічені дані отримати набагато простіше. Актуальність даного дослідження полягає у тому, що напівавтоматичне навчання є дуже корисним у багатьох реальних проблемах і останнім часом привертає увагу значної кількості дослідників.

Метою дослідження є правильне передбачення міток всіх немаркованих точок. В процесі навчання використовуються як марковані, так і немарковані точки. Ефективність алгоритму вимірюється коефіцієнтом помилок лише в немаркованих точках. Для досягнення мети необхідно розв'язати **задачу дослідження**, яка полягає у пошуку такої f, що мінімізує помилку апроксимації функції на вихідних даних. Для розв'язання задачі необхідно вирішити такі завдання:

1) провести огляд опублікованих джерел за тематикою дослідження;

2) аналіз можливих шляхів вирішення задачі напівавтоматичного навчання;

3) розробити метод, заснований на дифузійних картах, для розв'язання задач напівавтоматичного навчання;

4) навести числовий алгоритми для здійснення експериментальної перевірки результатів;

5) комп'ютерна реалізація алгоритму, що використовувався у дослідженні;

6) порівняти та проаналізувати отримані результати.

Об'єктом дослідження є методи розв'язання задач

напівавтоматичного навчання.

Предметом дослідження є дифузійні карти графу Лапласіану.

При розв'язанні поставлених завдань використовувались такі *методи дослідження*: методи лінійної та абстрактної алгебри, математичного та функціонального аналізу, теорії імовірностей, математичної статистики, теорії складності алгоритмів, методи комп'ютерного та статистичного моделювання.

Наукова новизна отриманих результатів полягає у застосуванні дифузійних карт до розробки нового методу розв'язання задачі напівавтоматичного навчання, що базується на використанні розрідженого графу Лапласіану, як апроксимації оператору Лапласа-Бельтрамі.

Практичне значення результатів полягає у покращенні основних метрик оцінки якості класифікації в задачах машинного навчання, в котрих використовується невелика частина маркованих та велика частина немарковани даних.

1 ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ

Проблема навчання на маркованих та немаркованих даних привертає значну увагу в останні роки.

Розглянемо загальну проблему навчання на маркованих та немаркованих даних. Дано набір точок $X = x_1, ..., x_l, x_{l+1}, ..., x_n$ та набір міток $L = 1,..., c$, перші l точок мають позначки $y_1, ..., y_l \in L$, а решта точок не позначені.

Означення 1.1. Навчання під наглядом - метод машинного навчання, який у процесі навчання використовує як і марковані, так і немарковані точки для підвищення ефективності алгоритму.

Означення 1.2. Функція $k : X \times X \to \mathbb{R}$ **ядром Мерсера (Mercer kernel)** якщо вона є неперервною, симетричною та невід'ємно визначеною функцією.

Алгоритми напівавтоматичного навчання використовують додаткові немарковані дані для кращого зображення форми основного розподілу даних та кращого їх узагальнення для нових зразків. Такі алгоритми використовують невелику кількість маркованих даних та велику кількість немаркованих даних для завдань класифікації.

Зауваження. Напівавтоматичні алгоритми повинні робити припущення щодо розподілу набору даних, щоб досягти підвищення продуктивності.

1.1 Класифікація задач машинного навчання

Кажуть, що проблема розуміння інтелекту є найбільшою проблемою науки сьогодні та проблемою цього століття - оскільки розшифровка генетичного коду була у другій половині минулого. Можна стверджувати, що проблема навчання являє собою ворота для розуміння інтелекту в

мозку та машинах, для виявлення того, як працює людський мозок, та створення інтелектуальних машин, які навчаються на досвіді та покращують свої компетенції, як це роблять діти. В інженерії способи навчання дозволяють розробляти програмне забезпечення, яке можна швидко налаштувати для обробки обсягу інформації та потоку даних навколо нас, що постійно зростає.

Прикладів безліч. Протягом останніх десятиліть експерименти з фізики частинок дали дуже велику кількість даних. Секвенування геному робить те саме в біології. Інтернет - це величезне сховище різнорідної інформації, яка швидко змінюється та зростає в геометричній прогресії.

Ми віримо, що набір методів, заснованих на новій галузі науки та техніки, яка є відомою як "контрольоване навчання буде ключовою технологією для вилучення інформації з океану бітів навколо нас та осмислення її.

Навчання під наглядом, або навчання на прикладах, належать до систем, які тренуються, а не програмуються, із набором прикладів, тобто набором пар вхід-вихід. Системи, які могли б навчитися на прикладі виконувати конкретне завдання, мали б багато застосувань. Банк може використовувати програму для перевірки заявок на позики та затвердження “добрих”. Така система буде навчена набором даних із попередніх заявок на позику та досвідом роботи за замовчуванням. У цьому прикладі заявка на позику - це точка у багатовимірному просторі змінних, що характеризують її властивості; пов’язаний з ним результат є двійковою міткою "добре"чи "погано".

В іншому прикладі, виробник автомобілів може захотіти мати у своїх моделях систему для виявлення пішоходів, які можуть перейти дорогу, щоб попередити водія про можливу небезпеку під час руху в центрі міста. Таку систему можна навчити на позитивних і негативних прикладах: зображення пішоходів та зображень без людей.

Були розроблені алгоритми, які дозволяють діагностувати тип раку на основі набору вимірів рівня експресії багатьох тисяч людських генів

під час біопсії пухлини, виміряної за допомогою мікрочипу кДНК, що містить зонди для декількох генів. Знову ж таки, програмне забезпечення знаходить правило класифікації на безлічі прикладів, тобто на прикладах патернів експресії у пацієнтів з відомими діагнозами. Проблема в цьому випадку полягає у великій розмірності вхідного простору - близько 20 000 генів - і невеликій кількості прикладів, доступних для навчання - близько 50.

У наведених вище прикладах ми припускаємо, що це машина, яка навчена, а не запрограмована, виконувати завдання з урахуванням даних форми $(x_i, y_i)_{i=1}^m$. Навчання означає синтез функції, яка найкраще показує відношення між входами x_i та відповідними виходами y_i. Центральним питанням теорії навчання є те, наскільки добре ця функція узагальнює, тобто наскільки добре вона оцінює результати для невидимих раніше входів.

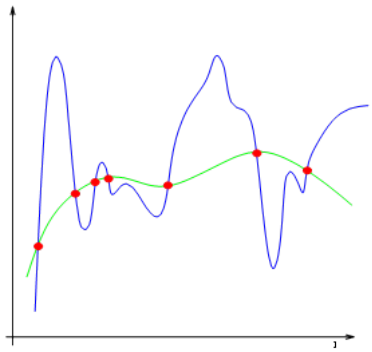

Рисунок 1.1 – Як ми можемо знайти функцію, яка здатна до узагальнення - серед багатьох функцій, які однаково добре підходять для тренувальних прикладів (тут m = 7)?

Як ми можемо підігнати навчальний набір даних $S_m = (x_i, y_i)_{i=1}^m$ з функцією прийняття рішень $f : X \to Y$ - де X - підмножина $\mathbb{R}^n$ та $Y \subset \mathbb{R}$? Ось алгоритм, який робить саме це, і який є майже магічним для простоти та ефективності:

1) Почніть з даних $(x_i, y_i)_{i=1}^m$

2) Виберіть симетричну, позитивно визначену функцію:

$K_x(x') = K(x,x')$, неперервну на $X \times X$.

Ядро $K(t,s)$ позитивно визначене, якщо:

$$\sum_{i,j=1}^{n} c_i c_j K(t_i,t_j) \geqslant 0$$

для будь-якого $n \in \mathbb{N}$ та вибору $t_1, ..., t_n \in X$ та $c_1, ..., c_n \in \mathbb{R}$.

Прикладом такого додатньо-визначеного ядра є гауссіана:

$$K(x,x') = e^{-\frac{\|x-x'\|^2}{2\sigma^2}} \tag{1.1}$$

обмежена на $X \times X$.

3) Знайдіть $f : X \to Y$:

$$f(x) = \sum_{i=1}^{m} c_i \mathbf{K}_{x_i}(x) \tag{1.2}$$

де $c = (c_1, ..., c_m)$ та

$$(m\gamma\mathbf{I} + \mathbf{K})c = y \tag{1.3}$$

де $\mathbf{I}$ - одинична матриця, $\mathbf{K}$ - квадратно позитивно визначена матриця з елементами $K_{i,j} = K(x_i, x_j)$, а y - вектор з координатами y_i. Параметр γ є додатним, дійсним числом.

Лінійна система рівнянь 1.3 у m змінних є добре складеною, оскільки $\mathbf{K}$ є невід'ємним, а $(m\gamma\mathbf{I} + \mathbf{K})$ додатним. Цей тип рівняння вивчався з часів Гаусса, і алгоритми його ефективного вирішення представляє одну з найбільш розвинених областей чисельного та обчислювального аналізу.

Що говорить рівняння 1.2? У випадку ядра Гауса рівняння апроксимує невідому функцію зваженою суперпозицією гаусових куль, кожна з яких розташована в місці x_i одного з m прикладів. Вага c_i кожного гаусової кулі така, щоб мінімізувати регуляризовану емпіричну помилку, тобто помилку на навчальному наборі. Параметри σ та γ контролюють ступінь згладжування, толерантність до шуму та узагальнення. Зауважимо, що для ядра Гаусса з $\sigma \to 0$ подання рівняння

1.2 фактично стає таблицею пошуку, яка не може узагальнювати (вона забезпечує правильний $y = y_i$ лише тоді, коли $x = x_i$, а в іншому випадку виводить 0).

Алгоритм добре працює в ряді додатків, що включають регресію, а також двійкову класифікацію. В останньому випадку y_i навчального набору $(x_i, y_i)_{i=1}^m$ приймають значення $\{-1, +1\}$; передбачена мітка тоді $\{-1, +1\}$, залежно від знака функції f рівняння 1.2.

Програми регресії є найдавнішими. Як правило, вони включали дані про навчання в невеликій кількості вимірів. Зовсім недавно вони також включали типові навчальні дані, іноді з дуже великими розмірами. Одним із прикладів є використання алгоритмів у комп'ютерній графіці для синтезу нових зображень та відео. Інша успішна програма - обернена проблема оцінки вираження обличчя та пози об'єкта на зображенні. Ще один випадок - це управління механічними руками. Є також додатки у фінансах, як, наприклад, оцінка ціни похідних цінних паперів, таких як опціони на акції. У цьому випадку алгоритм замінює класичне рівняння Блека-Скоулза (похідне від перших принципів) шляхом вивчення карти з вхідного простору (волатильність, основна ціна акції, час до закінчення строку дії опції тощо) на вихідний простір (ціна варіанту) з історичних даних.

Бінарних класифікаційних програм існує безліч. Алгоритм був використаний для двійкової класифікації ряду задач. Він також використовувався для здійснення розпізнавання візуального об'єкта незалежно від зору, зокрема, розпізнавання обличчя та категоризації. Інші додатки охоплюють біоінформатику для класифікації раку людини за даними мікрочипів, узагальнення тексту, класифікація звуку. Дуже тісно пов'язаний класифікатор Support Vector Machine був використаний для того самого сімейства програм, зокрема для біоінформатики та розпізнавання облич та виявлення автомобілів та пішоходів.

Дивно, але зовсім недавно було зрозуміло, що той самий лінійний алгоритм не тільки добре працює, але й повністю порівнянний у задачах

двійкової класифікації з найпопулярнішими класифікаторами сьогодні.

Описаний алгоритм можна отримати з регуляризації Тихонова. Щоб знайти мінімізатор помилки, ми можемо спробувати розв'язати проблему - звану емпіричною мінімізацією ризику - пошуку функції в $\mathcal{H}$, яка мінімізує

$$\frac{1}{m}\sum(f(x_i)-y_i)^2$$

що в цілому неправильно сформовано, залежно від вибору простору гіпотез. Згідно з Тихоновим, ми замість цього замість помилки на просторі гіпотез $\mathcal{H}_\mathcal{K}$ мінімізуємо для фіксованого додатного параметра γ регуляризований функціонал

$$\frac{1}{m}\sum(f(x_i)-y_i)^2+\gamma\|f\|_K^2 \tag{1.4}$$

де $\|f\|_K^2$ - норма в $\mathcal{H}_\mathcal{K}$ - Гільбертів простір з відтворювальним ядром(RKHS), визначений ядром K.

Останній доданок у рівнянні 1.4 - званий регуляризатор - зумовлює плавність та унікальність рішення.

Визначимо спочатку норму $\|f\|_K^2$. Розглянемо простір лінійної оболонки $K_{\overline{x_j}}$. Ми використовуємо $\overline{x_j}$, щоб підкреслити, що елементи з X, використані в цій конструкції, взагалі не мають нічого спільного з навчальним набором $(x_i)_{i=1}^m$. Визначимо внутрішній скалярний добуток у цьому просторі, обравши $\langle K_x, K_{\overline{x_j}}\rangle = K(x,\overline{x_j})$ і лінійно продовжуючи до $\sum_{j=1}^r a_j\mathbf{K}_{\overline{\mathbf{x}_\mathbf{j}}}$. Замиканням простору в асоційованій нормі є гільбертів простір з відтворювальним ядром, який є простором Гільберта $\mathcal{H}_\mathcal{K}$ з нормою $\|f\|_K^2$. Зауважимо, що $\langle f, K_x\rangle = f(x)$ для $f\in\mathcal{H}_\mathcal{K}$ (просто нехай $f=K_{\overline{x_j}}$ і продовжується лінійно).

Щоб мінімізувати функціонал у рівнянні 1.4, ми беремо функціональну похідну відносно f, застосовуємо його до елемента $\overline{f}$ RKHS і встановлюємо рівним 0. Отримуємо

$$\frac{1}{m}\sum_{i=1}^m(y_i-f(x_i))\overline{f}(x_i)-\gamma\langle f,\overline{f}\rangle=0. \tag{1.5}$$

Рівняння 1.5 повинно бути дійсним для будь-якого $\overline{f}$. Зокрема, значення $\overline{f} = K_x$ дає

$$f(x) = \sum_{i=1}^{m} c_i \mathbf{K}_{x_i}(x) \tag{1.6}$$

де

$$c_i = \frac{y_i - f(x_i)}{m\gamma} \tag{1.7}$$

оскільки $\langle f, \mathbf{K}_x \rangle = f(x)$. Отримуємо рівняння 1.3, підставивши рівняння 1.6 у рівняння 1.7.

Також зауважимо, що по суті той самий висновок для загальної функції втрат $V(y, f(x))$, замість $(f(x) - y)^2$, дає те саме рівняння 1.6, але рівняння 1.3 тепер інше і, загалом, нелінійне, залежно від форми V. Зокрема, популярні алгоритми регресії SVM та класифікації SVM відповідають спеціальним виборам не квадратичних V, один для забезпечення "надійної"міри похибки, а інший для наближення ідеальної функції втрат, що відповідає двійковій класифікації. В обох випадках рішення все ще має однакову форму рівняння 1.6 для будь-якого вибору ядра $\mathbf{K}$. Коефіцієнти c_i вже не задаються рівняннями 1.7, але їх потрібно знайти, вирішуючи квадратичну задачу програмування.

1.2 Різновид алгоритмів напівавтоматичного навчання

Існує дві основні моделі розповсюдження міток: Label Propagation та Label Spreading. Обидві вони працюють шляхом побудови графу подібності за всіма елементами вхідного набору даних.

Label Propagation та Label Spreading відрізняються модифікаціями матриці схожості, яка описує граф, а також Label Spreading певною мірою може змінювати клас справжніх маркованих даних, тобто змінюється впевненість у розподілі класів в межах заданого відсотка.

Label Propagation використовує необроблену матрицю подібності, побудовану з даних без змін. На відміну від цього, Label Spreading

мінімізує функцію втрат, яка має властивості регуляризації, тому вона часто є більш надійною до шуму. Алгоритм повторює модифіковану версію вихідного графа та нормалізує вагові значення ребра шляхом обчислення нормованої матриці графа Лапласіана.

Моделі розповсюдження міток мають два вбудовані методи ядра. Вибір ядра впливає як на масштабованість, так і на продуктивність алгоритмів. Доступні:

1) rbf ($exp(-\gamma|x-y|^2), \gamma > 0$). γ визначається ключовим словом *гама*.

2) knn ($\mathbb{1}[x' \in \text{kNN}(x)]$). k визначається ключовим словом n neighbors.

Ядро RBF створить повністю зв'язаний граф, який представлений в пам'яті щільною матрицею. Ця матриця може бути дуже великою, і в поєднанні з витратами на виконання розрахунку повного множення матриці для кожної ітерації алгоритму може призвести до надмірно тривалих часів роботи. З іншого боку, ядро KNN буде виробляти набагато зручнішу для пам'яті розріджену матрицю, що може значно зменшити час роботи.

Досліджується використання немаркованих даних, щоб допомогти маркованим даним у класифікації. Розповсюдження міток (Label Propagation) - простий ітераційний алгоритм в основі якого є розповсюдження міток через набір даних уздовж областей високої щільності, визначених немаркованими даними.

Марковані дані необхідні для навчання під наглядом. Однак вони часто доступні лише у невеликих кількостях, тоді як немаркованих даних може бути багато. Використання немаркованих даних разом із маркованими даними представляє як теоретичний, так і практичний інтерес. Метод передбачає, що тісніші точки даних, як правило, мають подібні мітки класів аналогічно методу k найближчих сусідів (kNN) у традиційному контрольованому навчанні.

Як результат, метод поширює мітки через щільні немарковані області даних. Мітки вузла поширюються на всі вузли відповідно до

їхньої близькості. Тим часом фіксуються мітки на маркованих даних. Таким чином, марковані дані діють як джерела, які виштовхують мітки через немарковані дані.

Модель Label Spreading подібна до базового алгоритму розповсюдження міток, але використовує матрицю спорідненості на основі нормалізованого графа Лапласіана. Родзинкою алгоритму Label Spreading є здатність змінювати певною мірою клас справжніх маркованих даних.

Основна ідея методу: надання дозволу кожній точці ітеративно розповсюджувати інформацію про мітку своїм сусідам, доки не буде досягнутий глобальний стабільний стан.

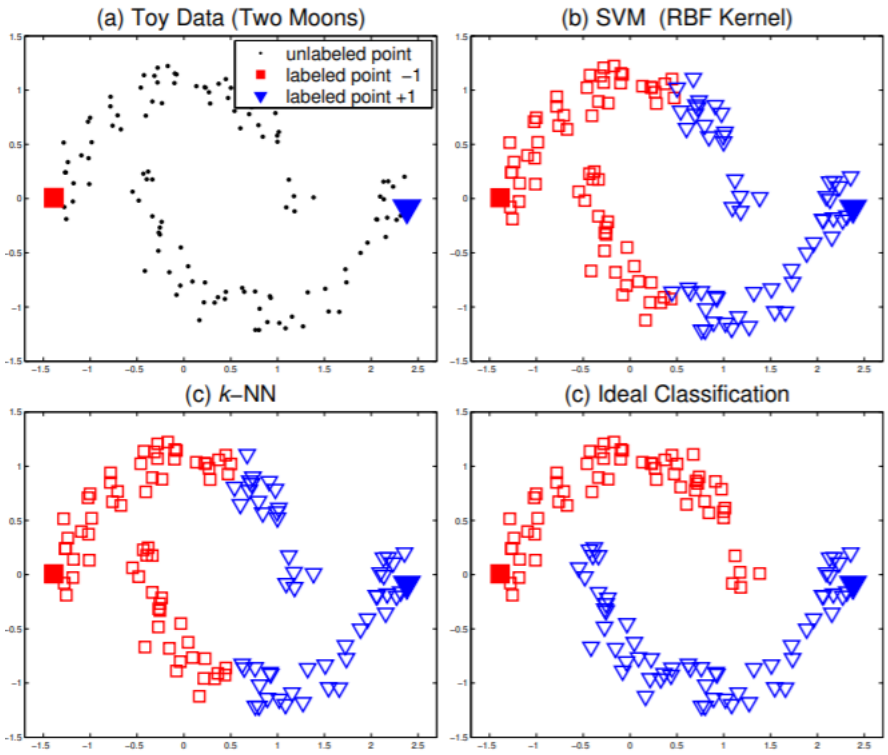

Рисунок 1.2 – Класифікація за зразком двох місяців. (а) іграшковий набір даних з двома позначеними точками; (б) класифікація результату, наданого SVM, із ядром RBF; (в) kNN з k = 1; (г) ідеальна класифікація, яку ми сподіваємось отримати.

Ключем до напівавтоматичних навчальних проблем є попереднє

припущення про послідовність, що означає:

– сусідні точки, ймовірно, матимуть однакову позначку;

– точки на одній і тій же структурі (зазвичай називаються кластером або многовидом), ймовірно, матимуть однакову позначку.

Щоб проілюструвати попереднє припущення про узгодженість, яке лежить в основі напівавтоматичного навчання, розглянемо набір іграшкових даних, створений за зразком двох сплетених місяців на 1.2(а). Кожна точка повинна бути схожа на точки в її локальному сусідстві, і, крім того, точки на одному місяці повинні бути більше схожими один на одного, ніж на точки на другому місяці. Результати класифікації, отримані за допомогою методу опорних векторів (SVM) з ядром RBF та kNN, показані на рисунках 1.2(б) та 1.2(в) відповідно. Однак, згідно з припущенням про узгодженість, два місяці слід класифікувати, як показано на рисунку 1.2(г).

1.3 Збіжність дифузійних карт Лапласіана

Геометрично засновані методи для різних завдань машинного навчання привернули значну увагу за останні кілька років. В роботі [1] продемонстрована збіжність власних векторів хмарного Лапласіана до власних функцій оператора Лапласа-Бельтрамі на навчальному многовиді, тим самим встановлюючи перші результати збіжності для алгоритму зменшення спектральної розмірності в умовах різноманітності.

Об'єднуючою передумовою геометрично-вмотивованих підходів до аналізу даних та машинного навчання є припущення, що багато типів природних даних лежать на багатовимірному многовиді або поблизу нього. У сукупності цей клас алгоритмів навчання часто називають багатогранними алгоритмами навчання.

У роботі [1] наводиться теоретичний аналіз власних карт Лапласіана, основи, що базується на власних векторах графа Лапласіана, пов'язаних із даними хмари точок. Більш конкретно, було доведено, що

за певних умов власні вектори графа Лапласіана збігаються із власною функцією оператора Лапласа-Бельтрамі на підлягаючому многовиді. Зазначимо, що в математиці многовид Лапласіана є класичним добре дослідженим об'єктом диференціальної геометрії. Це один з ключових об'єктів, пов'язаних із загально-диференційованим Рімановим многовидом.

В алгоритмах навчання, що базуються на многовидах, сам основний многовид, як правило, невідомий. Тому дифузійні карти з многовида потрібно оцінювати, використовуючи дані з хмари точок. Загальною стратегією наближення в цих методах є побудова графа суміжності, пов'язаного з хмарою точок. Основна інтуїція полягає в тому, що, оскільки граф є наближенням многовида, то висновок, заснований на структурі графа, відповідає бажаному висновку на основі геометричної структури многовида.

Теоретичні результати для обґрунтування цієї інтуїції були доведені протягом останніх кількох років. Спираючись на останні результати щодо функціональної збіжності наближення для оператора Лапласа-Бельтрамі з використанням теплових ядер та результатів щодо узгодженості власних функцій для емпіричних наближень таких операторів, була доведена збіжність алгоритму власних карт Лапласіана. Зазначимо, що для доведення збіжності спектрального методу потрібно продемонструвати збіжність емпіричних власних значень та власних функцій.

Результати в цій роботі стосуються випадку рівномірного розподілу ймовірностей на многовиді.

Хоча аналогії між геометрією многовидів та геометрією графів добре відомі в спектральній теорії графів та деяких областях диференціальної геометрії [2], точна природа цієї паралелі зазвичай не є точною.

Основним результатом отриманим в роботі [1] є збіжність власних векторів графа Лапласіана, пов'язаних із набором даних хмари точок, із власними функціями оператора Лапласа-Бельтрамі, коли дані відбираються з рівномірного розподілу ймовірностей на вбудованому

многовиді.

Далі ми будемо вважати, що різноманіття $\mathcal{M}$ є компактним нескінченно диференційованим римановим підмноговидом $\mathbb{R}^N$ без межі. Згадаймо тепер, що оператор Лапласа-Бельтрамі Δ на $\mathcal{M}$ є диференціальним оператором $\Delta : C^2 \to L^2$, визначений як

$$\Delta f = -div(\nabla f)$$

де ∇f - векторне градієнтне поле, а div позначає дивергенцію.

Δ є позитивним напіввизначеним самоспряженим оператором і має дискретний спектр на компактному многовиді. Ми, як правило, будемо позначати його i-те найменше власне значення через λ_i, а відповідну власну функцію - через e_i.

Визначимо оператор $\mathbf{L}^\epsilon : \mathcal{L}^2(\mathcal{M}) \to L^2(\mathcal{M})$ наступним чином (μ - стандартна міра):

$$\mathbf{L}^\epsilon(f)(p) = (4\pi\epsilon)^{-\frac{k+2}{2}} \left(\int_{\mathcal{M}} e^{\frac{\|p-q\|^2}{4\epsilon}} f(p) d\mu_q - \int_{\mathcal{M}} e^{\frac{\|p-q\|^2}{4\epsilon}} f(q) d\mu_q \right)$$

Якщо x_i - точки даних, відповідна емпірична версія задається як

$$\hat{\mathbf{L}}^\epsilon_n(f)(p) = \frac{(4\pi\epsilon)^{-\frac{k+2}{2}}}{n} \left(\sum_i e^{\frac{\|p-x_i\|^2}{4\epsilon}} f(p) - \sum_i e^{\frac{\|p-x_i\|^2}{4\epsilon}} f(x_i) \right)$$

Оператор $\hat{\mathbf{L}}^\epsilon_n$ - це *хмара точок Лапласіана*, яка лягає в основу алгоритму власних карт Лапласіана для напівавтоматичного навчання. Неважко помітити, що він діє шляхом множення матриць на функції, обмежені хмарою точок, при цьому матриця є відповідним графом Лапласіана. Будемо вважати, що x_i випадкові, незалежні та однаково розподілені, відібрані з $\mathcal{M}$ відповідно до рівномірного розподілу.

Основна теорема показує, що існує спосіб вибору послідовності ϵ_n, такий, що власні функції емпіричних операторів $\hat{\mathbf{L}}^\epsilon_n$ збігаються з власними функціями оператора Лапласа-Бельтрамі Δ за ймовірністю.

Теорема 1.1. *Нехай $\lambda^\epsilon_{n,i}$ - i-те власне значення $\hat{\mathbf{L}}^\epsilon_n$ та $e^\epsilon_{n,i}$ -*

відповідна власна функція (яка, для кожного фіксованого i, буде показано, що існує для ϵ досить малого). Нехай λ_i та e_i - відповідне власне значення та власна функція Δ відповідно. Тоді існує послідовність $\epsilon_n \to 0$, така що

$$\lim_{n\to\infty} \lambda_{n,i}^{\epsilon_n} = \lambda_i$$

$$\lim_{n\to\infty} \|e_{n,i}^{\epsilon_n}(x) - e_i(x)\|_2 = 0$$

де межі ймовірні.

Доведення основної теореми складається з двох основних частин:

– спектральна збіжність функціонального наближення $\mathbf{L}^\epsilon$ до Δ при $\epsilon \to 0$,

– спектральна збіжність емпіричного наближення $\hat{\mathbf{L}}_n^\epsilon$ до $\mathbf{L}^\epsilon$, оскільки кількість точок даних n прагне до нескінченності.

Потім ці два типи збіжності складаються разом, щоб отримати основну теорему 1.1.

Теорема 1.2. *Нехай $\lambda_i, \lambda_i^\epsilon, e_i, e_i^\epsilon$ - i-е найменше власне значення та відповідні власні функції Δ та $\mathbf{L}^\epsilon$ відповідно. Тоді*

$$\lim_{\epsilon\to 0} |\lambda_i - \lambda_i^\epsilon| = 0$$

$$\lim_{\epsilon\to 0} \|e_i - e_i^\epsilon\|_2 = 0$$

Теорема 1.3. *Для фіксованого досить малого ϵ нехай $\lambda_{n,i}^\epsilon$ та λ_i^ϵ i є i-м власним значенням $\hat{\mathbf{L}}_n^\epsilon$ та $\mathbf{L}^\epsilon$ відповідно. Нехай $e_{n,i}^\epsilon$ та e_i^ϵ - відповідні власні функції. Тоді*

$$\lim_{\epsilon\to 0} \lambda_{n,i}^\epsilon = \lambda_i^\epsilon$$

$$\lim_{\epsilon\to 0} \|e_{n,i}^\epsilon(x) - e_i^\epsilon(x)\|_2 = 0$$

припускаючи, що $\lambda_i^\epsilon \leqslant \frac{1}{2\epsilon}$. Збіжність майже напевно.

Зауваження. Зверніть увагу, що це передбачає збіжність для будь-якого фіксованого i, як тільки ϵ є достатньо малим.

Символічно ці дві теореми можна представити у верхньому рядку наступної діаграми:

$$\mathbf{Eig}\hat{\mathbf{L}}_n^{\epsilon} \xrightarrow[\text{probabilistic}]{n\to\infty} \mathbf{Eig}\mathbf{L}^{\epsilon} \xrightarrow[\text{deterministic}]{\epsilon\to 0} \mathbf{Eig}\boldsymbol{\Delta}$$

Після демонстрації двох типів результатів збіжності у верхньому рядку діаграми простий аргумент показує, що послідовність ϵ_n може бути обрана для гарантування збіжності, як у кінцевій теоремі 2.1.

,а саме: було розглянуто за яких умов і з якою швидкістю спектр графа Лапласіана, побудованого з незалежних однаково розподілених випадкових величин на підмноговиді, збігаються до спектру (зваженого) оператора Лапласа – Бельтрамі на підмноговиді як вибірці розміром $n \to \infty$ і радіусом сусідства $\epsilon \to 0$.

У роботі [3] вивчалася збіжність графа Лапласіана, породженого незалежними та однаково розподіленими випадковими величинами з m-вимірного підмноговиду $\mathcal{M}$ в $\mathbb{R}^d$, коли розмір вибірки n збільшується, а розмір околиця ϵ прагне до нуля. А саме, було розглянуто за яких умов і з якою швидкістю спектр графа Лапласіана збігаються до спектру (зваженого) оператора Лапласа – Бельтрамі на підмноговиді. Показано, що власні значення та власні вектори графа Лапласіана збігаються зі швидкістю $O\left(\left(\frac{\log n}{n}\right)^{\frac{1}{2m}}\right)$ до власних значень та власних функцій зваженого оператора Лапласа Бельтрамі в $\mathcal{M}$. Жодна інформація про підмноговид $\mathcal{M}$ не потрібна для побудови графа або «розширення поза вибіркою» власних векторів.

Висновки до розділу 1

Існує декілька поширених алгоритмів машинного навчання, що розв'язують задачі напівавтоматичного навчання на маркованих та

немаркованих даних.

Основні відмінності між різними напівавтоматичними алгоритмами навчання полягають у способі реалізації припущення про узгодженість. Принциповим підходом до формалізації припущення є розробка функції прийняття рішень, яка є достатньо *гладкою* щодо внутрішньої структури, виявленої відомими маркованими та немаркованими точками.

Наявні методи, що засновані на ядрі, розв'язують задачу напівавтоматичного навчання з використанням попередньо обраних ядер, або їх комбінацій.

Доведена збіжність власних векторів графа Лапласіана, що будується з набору даних хмари точок, із власними функціями оператора Лапласа-Бельтрамі, коли дані відбираються з рівномірного розподілу ймовірностей на вбудованому многовиді.

Наша задача полягає в побудові ядра на всіх вхідних даних і ця задача буде розв'язана в наступних розділах.

2 ТЕОРЕТИЧНІ ВІДОМОСТІ

У цій роботі розглядається розв'язок задач напівавтоматичного навчання з використанням невеликої кількості навчальних зразків.

2.1 Побудова ядер на основі даних для напівавтоматичного навчання

Традиційні методи, засновані на ядрі, використовують або фіксоване ядро, або комбінацію розумно вибраних ядер із фіксованого словника. На відміну від цього, ми створюємо залежне від даних ядро, використовуючи компоненти Mercer різних ядер, побудованих з використанням ідей з дифузійної геометрії, і використовуємо техніку регуляризації цього ядра з підібраними параметрами.

Різноманітні алгоритми машинного навчання використовують схеми єдиного штрафу або множинні штрафи Тихонова для регуляризації з різними підходами до аналізу даних.

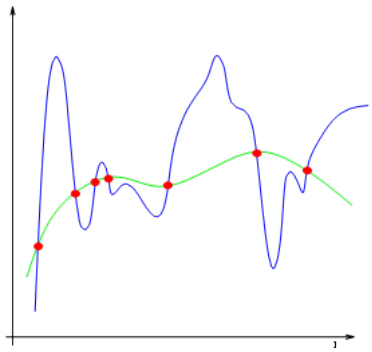

Рисунок 2.1 – Вплив регуляризації на усунення перенавчання моделі.

Більшість заснованих на ядрі алгоритмів забезпечують регуляризацію на основі відтворюючих ядер Гільбертоового простору. Проблема пошуку відповідного ядра для вивчення реальної функції

шляхом регуляризації розглядається, зокрема, в роботах [4, 3], де пропонувались різні підходи. Всі методи, згадані в цих роботах, мають справу з деяким набором ядер, які з'являються в результаті параметризації класичних ядер, своєрідної згортки «внутрішнього» та «зовнішнього» ядер або лінійної комбінації деяких функцій. Такі підходи призводять до проблеми багатоядерного навчання. Таким чином, проблема вибору ядра переноситься на проблему опису словника ядер, на якому виконується багатоядерне навчання.

У цій роботі запропоновано підхід до побудови ядра безпосередньо з спостережуваних даних, а не комбінування ядер із даного словника. У підході використовуються ідеї з дифузійної геометрії[1], де власні вектори графа Лапласіана, пов'язані з немаркованими даними, використовуються для імітації геометрії основного многовиду, яка зазвичай невідома.

Власні карти Лапласіана [1] та дифузійні карти [5] були запропоновані як інструменти для вилучення внутрішньої структури многовида шляхом розгляду власних векторів результуючого ненормалізованого графу Лапласіану; зокрема, власні карти Лапласа використовуються на першому кроці спектральної кластеризації, одного з найпопулярніших методів кластеризації на основі графів. Загалом, загальновідомо, що спектр графа Лапласіана, відповідного оператору Лапласа-Белтрамі фіксує важливі структурні відповідності геометричних властивостей графа [6] відповідного многовиду [7].

2.2 Постановка задачі

Дано незалежні однаково розподілені випадкові величини $X = x_1, ..., x_n$ з міри генерування даних μ в евклідовому просторі $\mathbb{R}^d$, метою більшості завдань машинного навчання та статистики є висновок про властивості μ.

Нехай $\mathcal{M}$ - опуклий компактний зв'язний m-вимірний риманів многовид, вкладений у $\mathbb{R}^d$, з $m \geqslant 2$.

Особливо цікавий випадок, якщо μ має опору на m-вимірному компактному підмноговиді $\mathcal{M}$ в $\mathbb{R}^d$, наприклад через сильну залежність між окремими ознаками. У цьому випадку можна побудувати граф сусідства на даних, з'єднавши всі вершини евклідової відстані менше певної шкали довжини ϵ, і таким чином отримати дискретне наближення невідомого многовида $\mathcal{M}$.

Припустимо, що абсолютне значення кривизни перерізу обмежене K, радіус інжективності i_0 і з досяжністю R. Запишемо $d(x, y)$ для відстані між x та y на многовиді та $|x - y|$ для евклідової відстані в $\mathbb{R}^d$.

Нехай μ - ймовірнісна міра на $\mathcal{M}$, яка має незникаючу липшицеву неперервну щільність p відносно риманового об'єму на $\mathcal{M}$ з константою Ліпшіца L_p. Компактність $\mathcal{M}$ і неперервність p гарантують існування константи $\alpha \geqslant 1$ такої, що

$$\frac{1}{\alpha} \leqslant p(x) \leqslant \alpha \text{ for all } x \in \mathcal{M} \tag{2.1}$$

Нехай $x_1, x_2, ..., x_n, ...$ – це послідовність незалежних однаково розподілених випадкових величин з μ.

Суб'єкт дифузійної геометрії прагне зрозуміти геометрію даних $\{x_i\}_{i=1}^n \subset \mathbb{R}^D$, випадково взятих з невідомого розподілу ймовірностей μ, де D, як правило, є великим розміром навколишнього середовища. Передбачається, що μ є гладким підмновидом $\mathbb{R}^D$. Оскільки многовид невідомий, його потрібно наблизити оператор Лапласа-Бельтрамі за допомогою використання графу Лапласіану.

Для того, щоб використати геометрію $\mathcal{M}$ з даних, ми будуємо граф із набором вершин $X := x_1, ..., x_n$. У найпростіших умовах для кожного $n \in \mathbb{N}$ вибираємо параметр сусідства $\epsilon = \epsilon_n$ і ставимо ребро від x_i до x_j та від x_j до x_i (і записуємо $x_i \sim x_j$) за умови, що $|x_i - x_j| \leqslant \epsilon$; нехай $E = (i, j) \in 1, ..., n^2 : x_i \sim x_j$ - множина таких ребер. В загальному ми розглядаємо зважені графи з вагами, які залежать від відстані між зв'язаними ними вершинами. З цією метою розглянемо спадаючу

функцію $\eta : [0, \infty) \to [0, \infty)$ з підтримкою на інтервалі $[0, 1]$ такою, що обмеження η на $[0, 1]$ є липшицевим безперервним. Нормалізація η за необхідності дозволяє з цього припустити, що

$$\int_{\mathbb{R}^m} \eta(|x|)dx = 1 \tag{2.2}$$

Для зручності припустимо, що $\eta(1/2) > 0$. Позначимо через

$$\sigma_\eta := \int_{\mathcal{R}^m} |y_1|^2 \eta(|y|)dy, \tag{2.3}$$

поверхневий натяг η, де y_1 являє собою першу координату вектора $y \in \mathbb{R}^m$.

Кожному даному ребру $(i, j) \in E$ присвоюємо вагу $w_{i,j} = 0$ де

$$w_{i,j} = \frac{1}{n\epsilon^m} \eta \left(\frac{|x_i - x_j|}{\epsilon} \right) \tag{2.4}$$

і ми розглянемо зважений граф (X, w) з $w_{i,j}$ як у 2.4 для кожного (i, j). Насправді зауважимо, що якщо точки x_i, x_j не з'єднані ребром в E, тоді $w_{i,j} = 0$.

Зауваження. Функцію η можна вибрати як $c\mathbf{1}_{[0,1]}$, а також таку гладку функцію, як

$$\eta(\epsilon) := c \begin{cases} exp\left(\frac{1}{\epsilon - 1}\right), & 0 \leqslant \epsilon < 1 \\ 0, & \epsilon \geqslant 1, \end{cases}.$$

де c - відповідна константа, що забезпечує нормалізацію.

Зауваження. Ми припустили, що $\eta : [0, 1] \to \mathbb{R}$ зменшується, а $\eta(1/2) > 0$, що означатиме, що $\eta(0) > 0$. Тим не менш, зауважимо, що жоден з результатів, представлених у цій статті, не зміниться, якщо ми змінимо значення $\eta(0)$. Зокрема, при бажанні ми допускаємо $\eta(0) = 0$, і ми можемо просто припустити, що η зменшується, а Ліпшиц у $(0, 1)$ (тоді умова $\eta(0) > 0$ змінюється на $\eta(0+) > 0$).

Це спостереження є важливим для того, щоб врахувати графи, де вершини не мають ребер із собою.

Для $\epsilon > 0$ та $x, y \in \mathbb{R}^D$, нехай:

Візьмемо $w_{i,j} = W^\epsilon(x, y)$

$$W^\epsilon(x, y) := exp\left(-\frac{\|x - y\|^2}{4\epsilon}\right). \tag{2.5}$$

Точки $\{x_i\}_{i=1}^n$ ми розглядаємо як вершини неорієнтованого графа з вагою ребра між x_i та x_j, заданими $W^\epsilon(x_i, x_j)$, визначаючи тим самим зважену матрицю суміжності, що позначається $\mathbf{W}^\epsilon$. Визначимо D^ϵ діагональною матрицею з i-м входом на діагоналі, заданим $\sum_{j=1}^n W^\epsilon(x_i, x_j)$. Граф Лапласіан визначається матрицею:

$$\mathbf{L}^\epsilon = \frac{1}{n}\{\mathbf{D}^\epsilon - \mathbf{W}^\epsilon\}. \tag{2.6}$$

Зауважимо, що для будь-яких дійсних чисел $a_1, \cdots, a_n$,

$$\sum_{i,j=1}^n a_i a_j L_{i,j}^\epsilon = \frac{1}{2n}\sum_{i,j=1}^n W_{i,j}^\epsilon \left(a_i - a_j\right)^2.$$

Усі власні значення $\mathbf{L}^\epsilon$ є дійсними та невід'ємними, і тому їх можна упорядкувати як:

$$0 = \lambda_1^\epsilon \leqslant \lambda_2^\epsilon \leqslant \cdots \leqslant \lambda_n^\epsilon \tag{2.7}$$

Зручно вважати власний вектор, що відповідає λ_k^ϵ, функцією на $\{x_j\}_{j=1}^n$, а не вектором у $\mathbb{R}^n$, і позначати його ϕ_k^ϵ, таким чином:

$$\lambda_k^\epsilon \phi_k^\epsilon(x_i) = \sum_{j=1}^n L_{i,j}^\epsilon \phi_k^\epsilon(x_j) = \frac{1}{n}\left(\phi_k^\epsilon(x_i)\sum_{j=1}^n W^\epsilon(x_i, x_j) - \sum_{j=1}^n W^\epsilon(x_i, x_j)\phi_k^\epsilon(x_j)\right), \quad i = 1, ..., n \tag{2.8}$$

Оскільки функція W^ϵ визначена на всьому навколишньому просторі, можна поширити функцію ϕ_k^ϵ на весь навколишній простір, використовуючи 2.8. Позначаючи цю розширену функцію через Φ_k^ϵ,

маємо:

$$\lambda_k^\epsilon \Phi_k^\epsilon(x) = \frac{1}{n}\left(\Phi_k^\epsilon(x)\sum_{j=1}^{n} W^\epsilon(x,x_j) - \sum_{j=1}^{n} W^\epsilon(x,x_j)\phi_k^\epsilon(x_j)\right), x \in \mathbb{R}^D. \quad (2.9)$$

Отримуємо:

$$\Phi_k^\epsilon(x) = \frac{\sum_{j=1}^{n} W^\epsilon(x,x_j)\phi_k^\epsilon(x_j)}{\sum_{j=1}^{n} W^\epsilon(x,x_j) - n\lambda_k^\epsilon} \quad (2.10)$$

для всіх $x \in \mathbb{R}^D$, для яких знаменник не дорівнює 0. Умову, що знаменник 2.10 не дорівнює 0 для будь-якого x, легко перевірити для будь-якого даного ϵ.

Порушення цієї умови для конкретного k можна розглядати як ознаку того, що для заданого обсягу n даних наближення власного значення λ_k відповідного оператора Лапласа-Бельтрамі власними значеннями λ_k^ϵ не може бути гарантоване з достатньою точністю.

Теорема щодо збіжності розширених власних функцій Φ_k^ϵ, обмежених до гладкого ноговиду X, до власних функцій оператора Лапласа-Бельтрамі на X.

Зауваження. Кожна Φ_k^ϵ є побудована з випадково вибраних даних $\{x_i\}_{i=1}^n$ з якогось невідомого многовиду X, і тому сама є випадковою.

Теорема 2.1. *Нехай X - гладкий, компактний многовид з розмірністю d, а μ - риманова міра на X, нормована на ймовірнісну міру. Нехай $x_i{}_{i=1}^n$ вибрано випадковим чином з μ, Φ_k^ϵ буде таким, як у (2.10), а Φ_k - власна функція оператора Лапласа-Белтрамі на X, що має той самий номер упорядкування, що і k, що відповідає власному значенню λ_k. Тоді існує послідовність $\epsilon_n \to 0$ така, що:*

$$\lim_{n\to\infty} \frac{1}{\epsilon^{1+d/2}} |\lambda_k^{\epsilon_n} - \lambda_k| = 0, \quad (2.11)$$

та

$$\lim_{n\to\infty} \|\Phi_k^{\epsilon_n} - \Phi_k\| = 0, \quad (2.12)$$

де норма є нормою L^2, а межі приймаються за ймовірністю, породженою μ.

Ми розглядаємо загальну проблему використання як маркованих, так і немаркованих даних для підвищення точності класифікації. Виходячи з припущення, що дані лежать на многовиді у багатовимірному просторі, ми розробляємо алгоритмічну структуру для класифікації частково маркованого набору даних.

Центральна ідея підходу полягає в тому, що класифікаційні функції, природно, визначаються лише на розглянутому підмноговиді, а не на загальному навколишньому просторі.

Використовуючи оператор Лапласа-Бельтрамі, створюється основа (власні карти Лапласа) для Гільбертового простору квадратично-інтегрованих функцій на підмноговиді. Щоб відновити таку основу, потрібні лише немарковані приклади. Отримавши таку основу, навчання можна проводити, використовуючи маркований набір даних. Наш алгоритм моделює многовид, використовуючи граф суміжності для даних, і апроксимує оператор Лапласа-Бельтрамі за допомогою графа Лапласіана.

Ми надаємо деталі алгоритму, його теоретичне обґрунтування та декілька практичних застосувань для класифікації активності біологічних даних.

У багатьох практичних програмах класифікації даних та видобутку даних можна знайти безліч легкодоступних не маркованих прикладів, тоді як збір маркованих прикладів може коштувати дорого і довго.

Класичні приклади включають розпізнавання об'єктів на зображеннях, розпізнавання мови, класифікацію статей новин за темами тощо. Останнім часом генетика також надає величезні обсяги легкодоступних даних. Однак класифікація цих даних передбачає експериментування і може бути дуже ресурсомісткою.

Отже, цікавим є розробка алгоритмів, які можуть використовувати як марковані, так і немарковані дані для класифікації та інших цілей.

Зокрема, ми використовуємо внутрішню структуру даних, щоб поліпшити класифікацію на немаркрованих прикладах, припускаючи, що дані знаходяться на багатовимірному многовиді. У деяких випадках здається обгрунтованим припущення, що дані лежать у багатовимірному многовиді або близько до нього. Наприклад, рукописну цифру 0 можна досить точно представити у вигляді еліпса, який повністю визначається координатами його фокусів і сумою відстаней від фокусів до будь-якої точки. Таким чином, простір еліпсів є п'ятивимірним многовидом. Фактичне рукописне значення 0 вимагає більшої кількості параметрів, але, можливо, не більше 15 або 20. З іншого боку, розмірність зовнішнього простору представлення - це кількість пікселів, яке, як правило, набагато вище.

Що стосується інших типів даних, питання про структуру многовиду видається значно більш залученим. Наприклад, у текстовій категорізації документи зазвичай представлені векторами, елементи яких (іноді зважуються) - кількість слів / термінів, що з'являються в документі. Далеко не ясно, чому простір документів повинен бути різноманітним. Однак безсумнівно, що він має складну внутрішню структуру і займає лише крихітну частину простору подання, який, як правило, є дуже багатовимірним, розмірність перевищує 1000. Ми показуємо, що навіть не маючи переконливих доказів для різноманітної структури все ще може використовувати наші методи з хорошими результатами. Важливо також зазначити, що, хоча об'єкти, як правило, представлені векторами в $\mathbb{R}^n$, природна відстань часто відрізняється від відстані, викликаної навколишнім простором $\mathbb{R}^n$.

Ми спостерігаємо, що для того, щоб оцінити многовид, все, що потрібно, - це немарковані дані (x'). Після того, як оцінено многовид, тоді оператор Лапласа-Бельтрамі може бути використаний для забезпечення основи для дифузійних карт, які є суттєво визначеними на цьому многовиді, а потім відповідний класифікатор оцінюється на основі маркованих прикладів. Таким чином, ми маємо природні рамки для

навчання з частково позначеними прикладами.

2.3 Приклад

Розглянемо спочатку бінарну класифікаційну задачу з класами C_1, C_2 та простором X, елементи яких слід класифікувати. Ймовірнісна модель для цієї задачі включала б два основних інгредієнти, щільність ймовірності $p(x)$ на X і щільності класів $\{p(C_1|x \in X)\}, \{p(C_2|x \in X)\}$. Дані без міток не обов'язково говорять нам багато про умовні розподіли, оскільки ми не можемо ідентифікувати класи без міток. Однак ми можемо покращити нашу оцінку щільності ймовірності $p(x)$, використовуючи немарковані дані.

Найпростіший приклад - два несуміжні класи на дійсній прямій. У цьому випадку ризик Байєса дорівнює нулю, і з урахуванням достатньої кількості немаркованих точок структуру можна повністю відновити лише одним позначеним прикладом. Загалом, немарковані дані надають нам інформацію про розподіл ймовірностей $p(x)$, тоді як марковані точки розповідають про умовні розподіли.

У цій роботі ми розглядаємо варіант цієї задачі, де $p(x)$ розміщує всі свої міри на компактному (низьковимірному) многовиді в $\mathbb{R}^n$.

Отже, немарковані приклади можуть бути використані для оцінки многовида, а марковані приклади вказують класифікатор, визначений на цьому многовиді.

Для мотивації використання конструкції многовида розглянемо простий синтетичний приклад, показаний на рисунку 2.2. Два класи складаються з двох частин кривої, показаної на рисунку 1. Нам дано кілька позначених точок та 500 немаркованих точок, показаних на рисунка 2 та 3 відповідно. Мета - встановити мітку точки, позначеної знаком питання. У контексті цього прикладу можна зробити кілька спостережень.

1) Спостерігаючи за зображенням на рисунку 2, ми бачимо, що ми

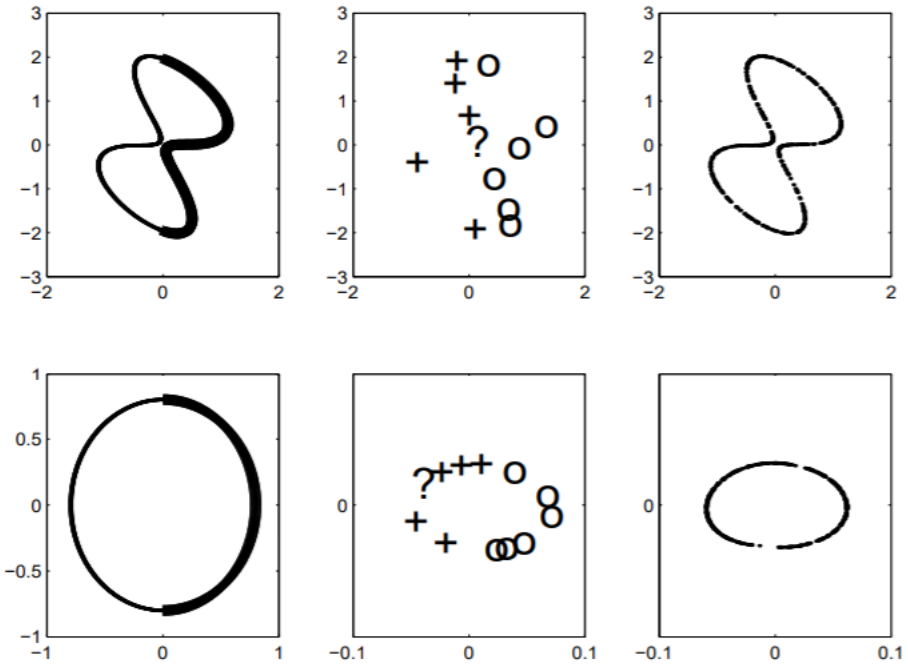

Рисунок 2.2 – 1. Два класи на плоскій кривій. 2. Позначені приклади. "?"є об'єктом для класифікації. 3. 500 випадкових прикладів без маркування. 4. Ідеальне зображення кривої. 5. Позиції маркованих точок та "?" після застосування власних функцій Лапласіана. 6. Позиції всіх прикладів

не можемо впевнено класифікувати знак питання, використовуючи лише позначені приклади. З іншого боку, проблема здається набагато більш здійсненною, враховуючи немарковані дані, показані на рисунку 3.

2) Оскільки існує бвзовий многовид, спочатку здається очевидним, що відстані вздовж кривої є більш значущими, ніж евклідові відстані в площині. Багато точок, які трапляються близько в площині, знаходяться на протилежних сторонах кривої. Тому замість того, щоб будувати класифікатори, визначені на площині ($\mathbb{R}^2$), видається кращим мати класифікатори, визначені на самій кривій.

3) Незважаючи на те, що дані свідчать про базовий многовид, проблема все ще не зовсім тривіальна, оскільки дві різні частини кривої заплутано наближаються одна до одної. Існує безліч можливих потенційних представлень многовида, і той, який надає сама крива, є незадовільним. В ідеалі ми хотіли б мати представлення даних, яке

фіксує той факт, що це замкнута крива. Більш конкретно, ми хотіли б вбудовувати криву, де координати змінюються якомога повільніше, коли хтось проходить криву. Таке ідеальне зображення показано на рисунку 4. Зверніть увагу, що обидва вони представляють одну і ту ж базову структуру многовиду, але з різними функціями координат. Виявляється на рисунку 6, беручи двовимірне представлення даних за допомогою власних карт Лапласіана, ми наближаємось до бажаного многовиду. На рисунку 5 показано розташування маркованих точок у новому просторі представлення. Ми бачимо, що "?"тепер падає прямо посередині знаків "+"і його клас легко можна визначити як "+".

Цей штучний приклад ілюструє, що відновлення многовида та розробка класифікаторів на самому многовиді може дати нам перевагу в задачах класифікації. Щоб відновити многовид, нам потрібні лише немарковані дані. Потім марковані дані використовуються для розробки класифікатора, визначеного на цьому многовиді. Таким чином, враховуючи набір маркованих прикладів $((x_i, y_i); x_i \in \mathbb{R}^d, y_i \in Y)$ та набір немаркрованих прикладів $(x_j \in \mathbb{R}^d)$, далі логічним є пошук класифікатора

$$f : \mathbb{R}^d \to Y$$

Оскільки d дуже велике, це негайно призводить до труднощів через "прокляття розмірності". Натомість ми використовуємо той факт, що всі $x_d \in \mathcal{M}$, де $\mathcal{M}$ є низьковимірним многовидом. Тому ми будуємо класифікатори форми

$$f : \mathcal{M} \to Y$$

В якості моделі для многовиду будемо використовувати зважений граф, вершинами якого є точки даних. Дві точки даних пов'язані ребром тоді і тільки тоді, коли точки сусідні, що, як правило, означає, що або відстань між ними менше, ніж деяке ϵ, або що одна з них знаходиться в безлічі n найближчих сусідів іншої.

Кожному ребру ми можемо віднести відстань між відповідними

точками. “Геодезична відстань” між двома вершинами - це довжина найкоротшого шляху між ними на графі суміжності. Зверніть увагу, що геодезична відстань може сильно відрізнятися від відстані в навколишньому просторі. Можна показати, що якщо точки відбирати з розподілу ймовірностей, що підтримується на всьому многовиді, геодезична відстань на графі буде збігатися до фактичної геодезичної відстані на многовиді, оскільки кількість точок прагне до нескінченності [8].

Після того, як ми встановили наближення до многовиду, нам потрібен метод для використання структури моделі для побудови класифікатора. Одним із можливих простих підходів було б використання «геодезичних найближчих сусідів». Найближчим геодезичним сусідом немаркованої точки u є позначена точка l така, що «геодезична відстань» по краях графа суміжності між точками u та l є найкоротшою. Тоді, як і у звичайних найближчих сусідів, мітка l присвоюється u.

Однак, хоча цей метод є простим і добре мотивованим, цей метод потенційно нестабільний. Навіть порівняно невелика кількість шуму або декілька викидів можуть різко змінити результати.

Наш підхід заснований на операторі Лапласа-Белтрамі на многовиді. Рімановий багатовид, тобто многовид з поняттям місцевої відстані, має природний оператор Δ на диференційованих функціях, який відомий як оператор Лапласа-Бельтрамі, або Лапласіан. Існує багато длосліджень про зв’язок між геометричними властивостями многовида та оператором Лапласа-Бельтрамі. для вступу до цієї теми.

У випадку $\mathbb{R}^n$ оператор Лапласа-Бельтрамі просто $\Delta = -\sum\limits_i \frac{\partial^2}{\partial x_i^2}$.

Δ - позитивно-напіввизначений самоспряжений (щодо скалярного добутку в $\mathcal{L}^2$) оператор на подвійно диференційованих функціях. Що примітно, виявляється, коли $\mathcal{M}$ є компактним многовидом, Δ має дискретний спектр, а власні функції Δ забезпечують ортогональну основу для простору Гільберта $\mathcal{L}^2(\mathcal{M})$.

Зауваження. Тут Δ визначається лише в підпросторі в $\mathcal{L}^2(\mathcal{M})$.

Тому будь-яку функцію $f \in \mathcal{L}^2(\mathcal{M})$ можна записати як

$$f(x) = \sum_{i=0}^{\infty} a_i e_i(x),$$

де e_i - власні функції, $\Delta e_i = \lambda_i e_i$.

Припустивши, що дані лежать на многовиді $\mathcal{M}$, ми розглянемо найпростішу модель, де приналежність до класу представлена квадратично-інтегруючою функцією $m : \mathcal{M} \to \{-1, 1\}$. Еквівалентно, можна сказати, що класи представлені вимірними множинами S_1, S_2 з нульовим перетином. Як варіант, якщо S_1 і S_2 перетинаються, ми можемо покласти $m(x) = 1 - 2Prob(x \in S_1)$. Єдиною умовою, яка нам потрібна, є те, що $m(x)$ є вимірною функцією.

Класифікаційну задачу можна інтерпретувати як проблему інтерполяції функції на багатоманітності. Оскільки функцію можна записати через власні функції Лапласіана, ми регулюємо коефіцієнти Лапласіана, щоб забезпечити оптимальне навчання до маркованих точок, так само, як ми можемо апроксимувати сигнал із рядом Фур'є. Насправді, коли $\mathcal{M}$ - коло, ми отримуємо ряд Фур'є. $m(x) \approx \sum_0^N a_i e_i(x)$.

Виявляється, не лише власні функції Лапласіана є природною основою для розгляду, але вони також задовольняють певній умові оптимальності. У певному сенсі, який ми уточнимо пізніше, вони забезпечують максимально плавне наближення, подібно до способу побудови сплайнів.

Дано k точок $x_1, ..., x_k \in \mathbb{R}^l$, ми припустимо, що перші $s < k$ точки мають мітки c_i, де $c_i \in \{-1, 1\}$, а решта не позначені. Мета - позначити немарковані точки. Ми також вводимо пряме розширення алгоритму, коли існує більше двох класів.

Тут дані $(x_i, y_i)_{i=1}^k$ вважаються випадковими, так що існує невідома міра ймовірності μ на просторі $X \times Y$, з якого беруться дані. Ця міра μ визначає функцію

$$f_\mu : X \to Y \tag{2.13}$$

, що задовольняє $f_\mu(x) = \int y d\mu_x$, де μ_x - умовна міра на $X \times Y$. З цієї конструкції f_μ можна сказати, що це справжня функція введення-виведення, що відображає середовище, яке виробляє дані.

Таким чином, вимірювання похибки f дорівнює

$$\int_X (f - f_\mu)^2 d\mu_X \tag{2.14}$$

де μ_X - міра на X, індукована μ.

Пошук функції f, що мінімізує цю помилку апроксимації на вихідних даних - мета теорії навчання. Для пошуку такого f важливо мати простір $\mathcal{H}$ - простір гіпотез - в якому можна працювати. Отже, розглянемо опуклий простір неперервних функцій $f : X \to Y$, (пам'ятаємо $Y \subset \mathbb{R}$), який як підмножина $C(X)$ є компактним, а де $C(X)$ - банаховий простір неперервних функцій з $\|f\| = \max_X |f(x)|$.

Узагальнений алгоритм розв'язку задачі:

– Побудова графу суміжності з n найближчими сусідами. Вузли i та j, що відповідають точкам x_i та x_j, з'єднані ребром, якщо i є серед n найближчих сусідів j або j є серед n найближчих сусідів i. Відстань може бути стандартною евклідовою відстанню в $\mathbb{R}^l$ або якоюсь іншою відстанню, наприклад, кутом, що є природним для задач класифікації тексту. Для наших експериментів ми приймаємо ваги як одне ціле. Таким чином $w_{ij} = 1$, якщо точки x_i і x_j близькі, а $w_{ij} = 0$ в іншому випадку.

– Власні функції. Обчисліть p власних векторів, що відповідають найменшим власним значенням для задачі власних векторів:

$$L\mathbf{e} = \lambda\mathbf{e}$$

Матриця $L = W - D$ - це граф Лапласіан для графа суміжності. Тут W - матриця суміжності, визначена вище, а D - діагональна матриця такого ж розміру, що і W, із записами суми рядків W, $D_{ii} = \sum_j W_{ji}$.

Лапласіан - це симетрична, позитивна напів-визначена матриця, яку можна вважати оператором над функціями, визначеними у вершинах

графа. Власні функції можна інтерпретувати як узагальнення низькочастотних гармонік Фур'є на многовиді, визначеному точками даних.

$$\mathbf{E} = \begin{pmatrix} e_{11} & e_{12} & e_{13} & \dots & e_{1k} \\ e_{21} & e_{22} & e_{23} & \dots & e_{2k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ e_{p1} & e_{p2} & e_{p3} & \dots & e_{pk} \end{pmatrix}$$

– Створення класифікатора. Для наближення класу ми мінімізуємо функцію помилки:

$$Err(a) = \sum_{i=1}^{s} \left(c_i - \sum_{j=1}^{p} a_j e_{ji} \right)^2$$

де p - кількість власних функцій, які ми хочемо застосувати, і сума береться по всіх маркованих точках, а мінімізація розглядається на просторі коефіцієнтів $a = (a_1, ..., a_p)^T$. Рішення задається формулою

$$\mathbf{a} = \left(\mathbf{E}_{lab}^T \mathbf{E}_{lab} \right)^{-1} \mathbf{E}_{lab}^T \mathbf{c}$$

де $\mathbf{c} = (c_1, ..., c_s)$ та

$$\mathbf{E}_{lab} = \begin{pmatrix} e_{11} & e_{12} & e_{13} & \dots & e_{1s} \\ e_{21} & e_{22} & e_{23} & \dots & e_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ e_{p1} & e_{p2} & e_{p3} & \dots & e_{ps} \end{pmatrix}$$

- матриця значень власних функцій на маркованих точках. Для випадку кількох класів ми створюємо класифікатор "один проти всіх"для кожного

окремого класу.

– Класифікація немічених точок. Якщо $x_i, i > s$ - це немечена точка, яку ми ставимо

$$c_i = \begin{cases} 1, & \text{if } \sum_{j=1}^{p} e_{ij} a_j \geqslant 0 \\ -1, & \text{if } \sum_{j=1}^{p} e_{ij} a_j < 0 \end{cases}$$

Це, звичайно, просто застосування лінійного класифікатора, побудованого на кроці 3. Якщо існує кілька класів, класифікатори «один проти всіх» змагаються, використовуючи $\sum_{j=1}^{p} e_{ij} a_j$ як міру довіри.

Висновки до розділу 2

Ми надали обґрунтування використання структури многовиду для задач класифікації. Звичайно, цю структуру не можна використовувати, якщо ми не маємо розумної моделі для многовида.

Маючи певне наближення до многовиду, ми застосовуєм метод для використання структури моделі для побудови класифікатора.

Отже, ми застосували дифузійні карти для роз'язку задачі напівавтоматичного навчання в контексті власних функцій для побудови ядра. В наступному розділі наведений числовий алгоритми для розв'язання задач.

3 ПОБУДОВА ЧИСЛОВОГО АЛГОРИТМИ ДЛЯ РОЗВ'ЯЗАННЯ ЗАДАЧ НАПІВАВТОМАТИЧНОГО НАВЧАННЯ

На практиці правильний вибір ϵ при приблизній побудові власних значень і власних функцій є делікатною справою, яка суттєво впливає на ефективність методів, заснованих на ядрі, заснованих на цих величинах. Деякі евристичні правила вибору ϵ були запропоновані в [9] та [10]. Ці правила не застосовуються універсально; їх потрібно вибирати відповідно до набору даних, що розглядаються.

3.1 Алгоритм вибору параметрів

На відміну від традиційної літератури, де фіксоване значення ϵ використовується для всіх власних значень та власних функцій, виконаємо підбір ϵ наступним чином:

$$K_n(x,t) = \sum_k \left(n\lambda_k^{\epsilon_{j_k}}\right)^{-1} \Phi_k^{\epsilon_{j_k}}(x)\Phi_k^{\epsilon_{j_k}}(t) \tag{3.1}$$

Тобто вибиремо власні значення та відповідні власні функції з різних ядер виду W^ϵ для побудови власного ядра. На відміну від традиційного методу комбінування різних ядер із фіксованого словника, в цій роботі використовується одне ядро, отримане використовуючи компоненти Mercer різних ядер зі словника.

Правило вибору параметра ϵ_{j_k} ґрунтується на загальновідомому критерії квазіоптимальності [11], який є одним з найпростіших і найдавніших, але все-таки досить ефективним методом вибору параметра регуляризації.

Згідно з цією стратегією, вибирається відповідне значення ϵ (тобто

параметр регуляризації) з послідовності допустимих значень $\{\epsilon_j\}$, які зазвичай утворюють геометричну послідовність, тобто $\epsilon_j = \epsilon_0 q^j, j = 1, 2, ..., M; q < 1$. Запропоновано використовувати критерій квазіоптимальності в контексті наближення власних значень оператора Лапласа-Бельтрамі. Тоді для кожного конкретного k обчислюємо послідовність наближених власних значень $\lambda_k^{\epsilon_j}, j = 1, 2, ..., M$, і вибераємо $\epsilon_{j_k} \in \{\epsilon_j\}$ таким чином, щоб різниці $|\lambda_k^{\epsilon_j} - \lambda_k^{\epsilon_{j-1}}|$ досягали свого мінімального значення при $j = j_k$.

Оскільки розмір сітки ϵ_j важко оцінити заздалегідь і, в той же час, це сильно впливає на ефективність методу, запропоновано наступну стратегію вибору розміру сітки M.

Зауваження. Підсумовування у формулі 3.1 повинно бути зроблено для індексів k, для яких відповідне власне значення $\lambda_k = \lambda_k^{\epsilon_{j_k}}$ є ненульовим.

Також відомо, що перше власне значення $\lambda_1^{\epsilon_j} = 0$. Щоб запобігти тому, щоб інші $\lambda_k^{\epsilon_j}$ ставали занадто близькими до нуля зі зменшенням ϵ_j, пропонується зупинити продовження послідовності ϵ_j, як тільки значення $\lambda_2^{\epsilon_M}$ стає досить малим. Отже, максимальний розмір сітки M - це найменше ціле число, для якого $\lambda_2^{\epsilon_M} < \lambda_2^{(thr)}$, де $\lambda_2^{(thr)}$ - деякий розрахунковий поріг. Враховуючи вищезазначене, формула для розрахунку ядра отримує наступний вигляд 3.1:

$$K_n(x,t) = 1 + \sum_{k=2}^{n} \left(n\lambda_k^{\epsilon_{j_k}}\right)^{-1} \Phi_k^{\epsilon_{j_k}}(x)\Phi_k^{\epsilon_{j_k}}(t) \tag{3.2}$$

Наведений нижче алгоритм 1 описує поєднання наближення 2.10 із

критерієм квазіоптимальності.

Алгоритм 1: Генерація відворюючого ядра з даних.

Наведені дані $\{x_i\}_{i=1}^n \subset X$. Введіть сітку для параметра

$\epsilon_j = q^j, j = 1, ..., M$

for *(j = 1:M)* **do**

  Рахуйте L^{ϵ_j} як в (2), та власну систему $(\phi_k^{\epsilon_j}, \lambda_k^{\epsilon_j}), k = 1, ..., n$

  if $\lambda_2^{\epsilon_j} < \lambda_2^{(thr)}$ **then**

    break

for *(k = 1:n)* **do**

  Знайти $\epsilon_k = arg\min_{\epsilon_j} |\lambda_k^{\epsilon_j} - \lambda_k^{\epsilon_{j-1}}|$.

  $\lambda_k := \lambda_k^{\epsilon_k}, \phi_k := \phi_k^{\epsilon_k}$

Рахуйте

$$\Phi_k^{\epsilon}(x) = \frac{\sum_{j=1}^n W^{\epsilon}(x, x_j)\phi_k^{\epsilon}(x_j)}{\sum_{j=1}^n W^{\epsilon}(x, x_j) - n\lambda_k^{\epsilon}}$$

Формування функції ядра

$$K_n(x,t) = 1 + \sum_{k=2}^{n} \left(n\lambda_k^{\epsilon_{j_k}}\right)^{-1} \Phi_k^{\epsilon_{j_k}}(x)\Phi_k^{\epsilon_{j_k}}(t) \tag{3.3}$$

Алгоритм 2 використовує побудоване ядро 3.3 для Ridge regression. Регресія виконується у поєднанні з принципом невідповідності для вибору

параметра регуляризації α.

Алгоритм 2: ridge регресія з побудованим ядром 3.3

Наведені дані $\{x_i\}_{i=1}^n \subset X\{x_i, y_i\}_{i=1}^m$ та розмічені дані $y = \{y\}_{i=1}^m$.

Сформуйте ядро, використовуючи алгоритм 1

Введіть сітку для параметра $\alpha : \alpha_k = p^k, k = 1, 2, ..., N$

Обчислити матрицю Грама $\hat{K}_m$, що складається з підматриці $K_n(x_i, x_j)_{i,j=1}^m(11)$ у позначених точках.

for *(k = 1:N)* **do**

Рахуйте C_{α_k} як

$$C_{\alpha_k} = \left(\alpha_k I + \hat{K}_m\right)^{-1} y,$$

Знайдіть α_{min} таким, щоб $\|\hat{K}_m C_{\alpha_k} - y\|$ було мінімізовано.

Функція прийняття рішень має вигляд

$$f_n^*(x) = \sum_{i=1}^{m} (C_{\alpha_{min}})_i K_n(x, x_i).$$

3.2 Перевірка результів

Для практичної перевірки створеного алгоритму, що використовувався у дослідженні, були підготовлені дані для класифікації біологічної активності молекул під мішень, використовуючи марковані дані з відкритих хімічних баз, проведений аналіз простору ознак, необхідних для опису хімічних сполук.

На вхід алгоритму для навчання класифікатора подавались 4715 молекл. З них маркованих даних – 214 молекул, немаркованих – 4501. Для кожної молекули розраховували 70089 дескрипторів, що їх описують. Мова йде про різноманітні $2D$ та $3D$ ознаки та фізико-хімічні властивості.

Після центрування та нормалізації даних наступним кроком було пониження розмірності простору ознак. Для цього використовувалися наступні техніки:

– видалення ознак, в котрих дуже мала дисперсія (менше 0.01);

– видалення однієї з попарно-корелюючих ознак в сенсі кореляції Пірсона (якщо кореляція перевищує 0.95 – залишаємо одну з ознак);

– видалення ознак, котрі які є неважливими для маішені. (Застосовувася алгоритм градієнтний біустинг на деревах рішень(XGBoost) на тренувальній вибірці.)

Маючи оброблений датасет для моделювання - ми перейшли до підбору гіперпараметрів розробленої моделі для підвищення основних метрик оцінки результатів класифікації, зокрема – точність, влучність та повнота. Для отримання бажаного була застосована технка перехресної перевірки[12].

У 3.2 наведено приклад програмної реалізації класифікатору, що вирішує задачу напівавтоматичного навчання описаним методом. Також були імплементовані існуючі методи розв'зяання задачі напівавтоматичного навчання такі як Label Propagation та Label Spreading. Результати класифікації також наведені у додатках.

У А.2 наведені результати класифікації на іграшковому наборі даних "два місяці а також на наборі даних біологічних сполук, що мають активніть під певну мішень. Розмірність простору ознак для даного набору даних не дозволяє візуалізвати їх на площині чи у просторі. При спробі зменшити розмірність простору ознак до 3 за допомогою методу головних компонент - пояснювальна дисперсія для кожної компоненти залишається дуже малою. Саме тому в додатках наведені лише матриці помилок та точність класифікації.

Висновки до розділу 3

Для підору гіперпараметру ϵ був використаний евристичний підхід. Проведена комп'ютерна реалізація алгоритму, що використовувався у дослідженні, розглянуті та опрацьовані біологічні дані, зокрема на яких тестувалася якість класифікації створеного алгоритму та існуючих

рішень.

Практично перевірено результати класифікації біологічної активності хімічних сполук під певну мішень, використовуючи розроблений метод напівавтоматичного навчання з використанням розрідженого графу Лапласіану.

Спроможність алгоритму проілюстрована за допомогою добре відомого набору даних "Два місяці а також за допомогою набору даних хімічних сполук. Для першого набору даних ми отримали нульову похибку тесту, використовуючи лише мінімальну кількість навчальних зразків.

ВИСНОВКИ

У ході даної роботи був проведений огляд розроблених методів напівавтоматичного навчання, який показав, що метрики оцінки якості класифікації, зокрема точність, влучність та повнота, а також час роботи алгоритму можуть бути покращені використовуючи залежне від даних регуляризоване ядро, побудованих з використанням ідей з дифузійної геометрії.

Тому було запропоновано використовувати роздріджений граф Лапласіан замість повного для розв'язання задачі напівавтоматичного навчання. Для побудови функції прийняття рішень були використані ядерні функції, побудовані з власних функцій дифузійних карт.

Комп'ютерна реалізація розробленого алгоритму дала змогу проаналізувати якість класифікації. Спроможність до розв'язування задач напівавтоматичного навчання перевірялась на іграшковому наборі даних "Два місяці а також на підготовлених даних біологічної активності молекул до певної мішені.

Завдяки гарно підібраним гіперпараметрам моделі розроблена методика дозволяє підвищити точність класифікації в середньому на 7.33% в порівнянні з існуючими методиками.

У майбутньому планується дослідити, який вплив має розрахунок вагових коефіцієнтів графу Лапласіану на точність передбачення моделлю немаркованих точок а також відшукати залежність точності класифікації від кількості маркованих точок, необхідних моделі для навчання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Belkin Mikhail, Niyogi Partha. Convergence of Laplacian eigenmaps //Advances in Neural Information Processing Systems. — 2007. — Vol. 19. —P. 129.

2. CHUNG Fan RK. Spectral Graph Theory, Regional Conference Series inMath. // CBMS, Amer. Math. Soc. — 1997.

3. Error estimates for spectral convergence of the graph Laplacian onrandom geometric graphs toward the Laplace–Beltrami operator /Nicol aas Garc ia Tiillos, Moritz Gerlach, Matthias Hein, Dejan Slepcev //Foundations of Computational Mathematics. — 2020. — Vol. 20, no. 4. —P. 827–887.

4. Data Based Construction of Kernels for Semi-Supervised LearningWith Less Labels / Hrushikesh Mhaskar, Sergei V. Pereverzyev,Vasyl Yu. Semenov, Evgeniya V. Semenova // Frontiers in AppliedMathematics and Statistics. — 2019. — Vol. 5. — P. 21. — Access mode:https://www.frontiersin.org/article/10.3389/fams.2019.00021.

5. Coifman Ronald R, Lafon Stephane. Diffusion maps // Applied andcomputational harmonic analysis. — 2006. — Vol. 21, no. 1. — P. 5–30.

6. Mohar Bojan. Some applications of Laplace eigenvalues of graphs // Graphsymmetry. — Springer, 1997. — P. 225–275.

7. Chavel Isaac. Eigenvalues in Riemannian geometry. — Academic press,1984.

8. Tenenbaum Joshua B, De Silva Vin, Langford John C. A global geometricframework for nonlinear dimensionality reduction // science. — 2000. — Vol.290, no. 5500. — P. 2319–2323.

9. Levy Bruno. Laplace-Beltrami Eigenfunctions Towards an Algorithm ThatUnderstands Geometry // Shape Modeling and Applications, 2006. SMI 512006. IEEE International Conference on. — 2006. — 07. — Vol. 2006. — P. 13– 13.

10. Rustamov Raif. Laplace-Beltrami eigenfunctions for deformation

invariantshape representation // Symposium on Geometry Processing. — 2007. —01. — P. 225–233.

11. Tikhonov Andrei Nikolaevich, Glasko Vladlen Borisovich. Use of theregularization method in non-linear problems // USSR ComputationalMathematics and Mathematical Physics. — 1965. — Vol. 5, no. 3. — P. 93–107.

12. Browne Michael W. Cross-validation methods // Journal of mathematicalpsychology. — 2000. — Vol. 44, no. 1. — P. 108–132.

ДОДАТОК А ТЕКСТИ ПРОГРАМ

Наведені тексти інструментальних програм для проведення експериментальних досліджень.

А.1 Програма 1

Лістинг 1: Main Model

```
import inspect
from collections import defaultdict
from functools import partial
from multiprocessing import Pool, cpu_count

import numpy as np
from scipy.spatial.distance import pdist, squareform
from tqdm import tqdm


# from sklearn.utils import check_X_y
# from sklearn.utils.multiclass import check_classification_targets


class Estimator(object):

    def __init__(self, eps_0=1, p_=0.5, N_=45, n_jobs=None):

        self.eps_0 = eps_0
        self.tol_ = 0
        self.n_, self.m_ = 0, 0
        self.n_jobs = n_jobs

        self.eigen_values_ = np.empty(0)
        self.eigen_vectors_ = np.empty(0)

        self.train_samples_ = 0
        self.coeff_ = np.empty(0)

        self.p_ = p_
        self.N_ = N_

        self.X_, self.y_ = np.empty(0), np.empty(0)
```

```
        self.X_train_, self.y_train_ = np.empty(0), np.empty(0)

    def __repr__(self):
        return f'{self.__class__.__name__}(' \
               f'eps_0={self.eps_0},␣' \
               f'tol={self.tol_},␣' \
               f'p={self.p_},␣' \
               f'N={self.N_})'

    @classmethod
    def _get_param_names(cls):
        """Get parameter names for the estimator"""
        # fetch the constructor or the original constructor before
        # deprecation wrapping if any
        init = getattr(cls.__init__, 'deprecated_original', cls.__init__)
        if init is object.__init__:
            # No explicit constructor to introspect
            return []

        # introspect the constructor arguments to find the model parameters
        # to represent
        init_signature = inspect.signature(init)
        # Consider the constructor parameters excluding 'self'
        parameters = [p for p in init_signature.parameters.values()
                      if p.name != 'self' and p.kind != p.VAR_KEYWORD]
        for p in parameters:
            if p.kind == p.VAR_POSITIONAL:
                raise RuntimeError("scikit-learn␣estimators␣should␣always␣"
                                   "specify␣their␣parameters␣in␣the␣signature"
                                   "␣of␣their␣__init__␣(no␣varargs)."
                                   "␣%s␣with␣constructor␣%s␣doesn't␣"
                                   "␣follow␣this␣convention."
                                   % (cls, init_signature))
        # Extract and sort argument names excluding 'self'
        return sorted([p.name for p in parameters])

    def get_params(self, deep=True):
        """
        Get parameters for this estimator.

        Parameters

        ----------

        deep : bool, default=True
            If True, will return the parameters for this estimator and
            contained subobjects that are estimators.
```

```
        Returns
        -------
        params : dict
            Parameter names mapped to their values.
        """
        out = dict()
        for key in self._get_param_names():
            value = getattr(self, key)
            if deep and hasattr(value, 'get_params'):
                deep_items = value.get_params().items()
                out.update((key + '__' + k, val) for k, val in deep_items)
            out[key] = value
        return out

    def set_params(self, **params):
        """
        Set the parameters of this estimator.
        The method works on simple estimators as well as on nested objects.

        Parameters
        ----------
        **params : dict
            Estimator parameters.

        Returns
        -------
        self : estimator instance
            Estimator instance.
        """
        if not params:
            # Simple optimization to gain speed (inspect is slow)
            return self
        valid_params = self.get_params(deep=True)

        nested_params = defaultdict(dict)  # grouped by prefix
        for key, value in params.items():
            key, delim, sub_key = key.partition('__')
            if key not in valid_params:
                raise ValueError('Invalid parameter %s for estimator %s. '
                                 'Check the list of available parameters '
                                 'with `estimator.get_params().keys()`.' %
                                 (key, self))

            if delim:
```

```
            nested_params[key][sub_key] = value
        else:
            setattr(self, key, value)
            valid_params[key] = value

    for key, sub_params in nested_params.items():
        valid_params[key].set_params(**sub_params)

    return self

# ----------------------------------------------------------------------

def weight(self, xi, xj):
    norm = np.linalg.norm(xi - xj)
    if norm <= self.tol_:
        return (
        4 * np.pi * self.tol_)**(-(self.m_ + 2) / 2) * \
        np.exp(-norm ** 2 / (4 * self.tol_)
        )
    else:
        return 0

def graph_Laplacian(self):
    L = np.zeros((self.n_, self.n_))
    for i in range(self.n_):
        for j in range(self.n_):
            if i != j:
                L[i][j] = - self.weight(self.X_[i, :], self.X_[j, :])
            elif i == j:
                for k in range(self.n_):
                    if i != k:
                        L[i][j] += self.weight(self.X_[i, :], self.X_[k, :])
    return L / self.n_

def epsilon_finder(self, q=0.9, lambda_thr=1e-6):

    j = 1
    eig_values_vectors = []

    while True:

        self.tol_ = self.eps_0 * q ** j
        L = self.graph_Laplacian()
        self.eigen_values_, self.eigen_vectors_ = np.linalg.eig(L)
```

```
            eig_values_vectors = np.append(eig_values_vectors, self.eigen_values_)
            if self.eigen_values_[1] < lambda_thr:
                break
            j += 1

        eig_values_vectors = np.reshape(eig_values_vectors, newshape=(-1, self.n_))

        if eig_values_vectors.shape[0] <= 1:
            return q ** (j - 1)
        else:
            d = {}
            for i in range(eig_values_vectors.shape[0] - 1):
                d[i] = np.linalg.norm(eig_values_vectors[i + 1] - eig_values_vectors[i])
            return q ** (min(d, key=d.get) + 1)

# ------------------------------------------------------------------

def eigen_func(self, x, k):
    weights = [self.weight(xi=x, xj=xj) for xj in self.X_]
    return np.dot(
        weights,
        self.eigen_vectors_[:, k]) / (np.sum(weights) -
        self.n_ * self.eigen_values_[k]
    )

def kernel_function(self, x, t):
    K = 0
    for k in range(self.n_):
        K += self.eigen_func(x, k) *
        self.eigen_func(t, k) / (self.n_ * self.eigen_values_[k])
    return K

def gram_matrix(self):

    K = np.zeros((self.train_samples_, self.train_samples_))
    for i in range(self.train_samples_):
        for j in range(self.train_samples_):
            K[i][j] = self.kernel_function(self.X_train_[i, :],
                                           self.X_train_[j, :])
    return K

# ------------------------------------------------------------------

def coeff_c_finder(self, K):
```

```
        alpha = [self.p_ ** k for k in range(self.N_)]

        Identity = np.identity(self.train_samples_)
        coeffs = np.zeros((self.N_, self.train_samples_))

        for k in range(self.N_):
            coeffs[k] = np.dot(np.linalg.inv(alpha[k] * Identity + K), self.y_train_)
        d = {}
        for k in range(self.N_):
            d[alpha[k]] = np.linalg.norm(np.dot(K, coeffs[k]) - self.y_train_)
        a = min(d, key=d.get)

        return np.dot(np.linalg.inv(a * Identity + K), self.y_train_)

    def decision_function(self, x):

        res = 0
        for i in range(self.train_samples_):
            res += self.coeff_[i] * self.kernel_function(x, self.X_train_[i, :])
        return res

    # ------------------------------------------------------------------

    def fit(self, X, y):

        self.X_ = X
        self.y_ = y
        self.n_, self.m_ = self.X_.shape

        self.tol_ = self.epsilon_finder()

        L = self.graph_Laplacian()
        self.eigen_values_, self.eigen_vectors_ = np.linalg.eig(L)

        self.X_train_, self.y_train_ = X[y != -1], y[y != -1]
        self.train_samples_ = self.X_train_.shape[0]

        K = self.gram_matrix()

        self.coeff_ = self.coeff_c_finder(K)
        return self

    def predict(self, X):
        return np.where(self.predict_proba(X) > .5, 1, 0)
```

```
    def predict_proba(self, X):
        if self.n_jobs:
            cores = self.n_jobs
        else:
            cores = cpu_count()

        data_split = np.array_split(X, cores)
        pool = Pool(cores)
        y_pred = np.concatenate(
            pool.map(partial(np.apply_along_axis, self.decision_function, 1),
                     tqdm(data_split))
        )
        pool.close()
        pool.join()
        return y_pred
```

A.2 Програма 2

Лістинг 2: Existing Model

```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.semi_supervised import (
    LabelPropagation,
    LabelSpreading,
    SelfTrainingClassifier
)
from sklearn.svm import SVC

from main import plot_decision_boundary, get_data


if __name__ == '__main__':
    X_train, X_test, y_train, y_test = get_data()

    X = np.concatenate([X_train, X_test], axis=0)
    y = np.concatenate([y_train, -1 * np.ones_like(y_test)], axis=0)

    models = (
        LabelPropagation(max_iter=10000),
```

```
    LabelSpreading(),
    SelfTrainingClassifier(base_estimator=SVC(probability=True, gamma="auto"))
)
color_maps = ('Blues', 'Greens', 'Reds')

for model, cmap in zip(models, color_maps):
    model.fit(X, y)
    y_pred = model.predict(X[y == -1])

    sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt="d", cmap=cmap)
    print('-'*50, f'\nModel name: {model.__str__()}\n'
          f'Accuracy_score: {accuracy_score(y_test, y_pred)}')
    plt.show()

    plot_decision_boundary(X, y, y_test, y_pred, model)
```

ДОДАТОК Б ВЕЛИКІ РИСУНКИ

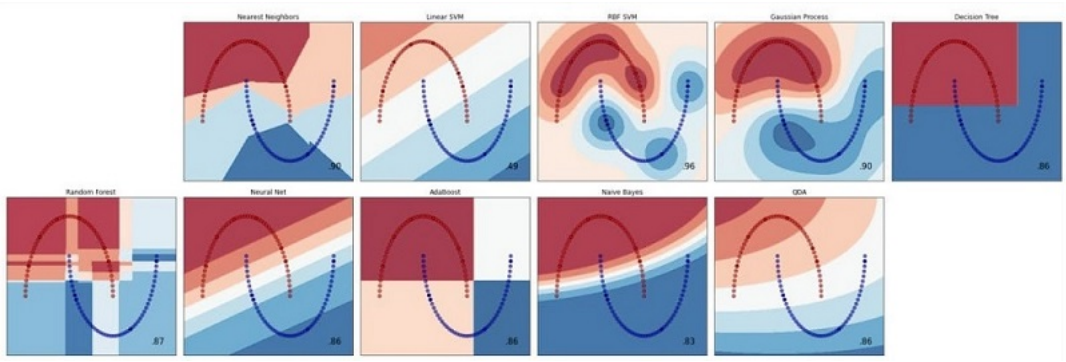

Рисунок Б.1 – Класифікація різними моделями іграшкового датасету.

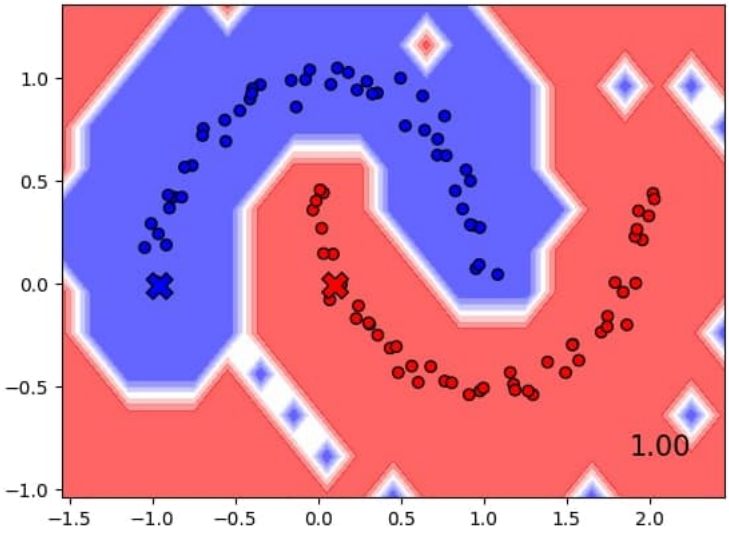

Рисунок Б.2 – Класифікація іграшкового датасету власною моделлю.

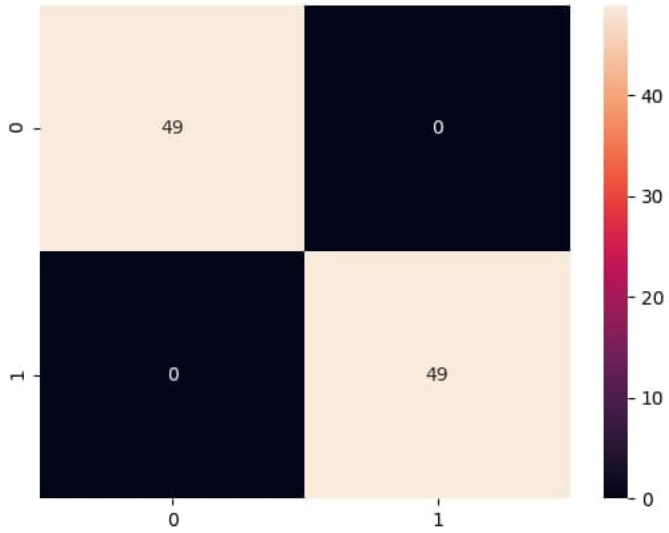

Рисунок Б.3 – Матриця невідповідностей за результатами класифікації власним методом.

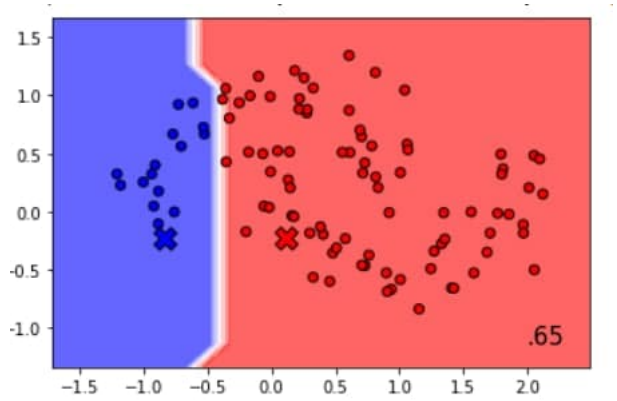

Рисунок Б.4 – Класифікація іграшкового датасету методом LabelPropagation.

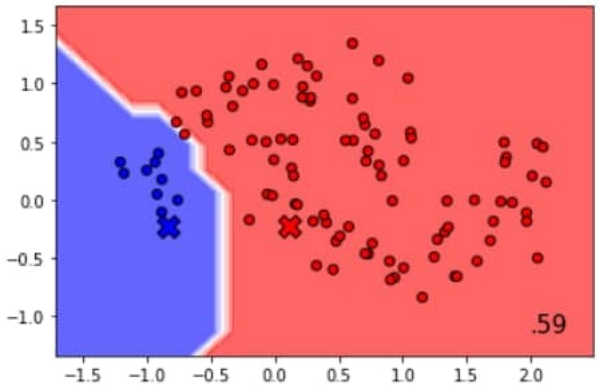

Рисунок Б.5 – Класифікація іграшкового датасету методом LabelSpreading.

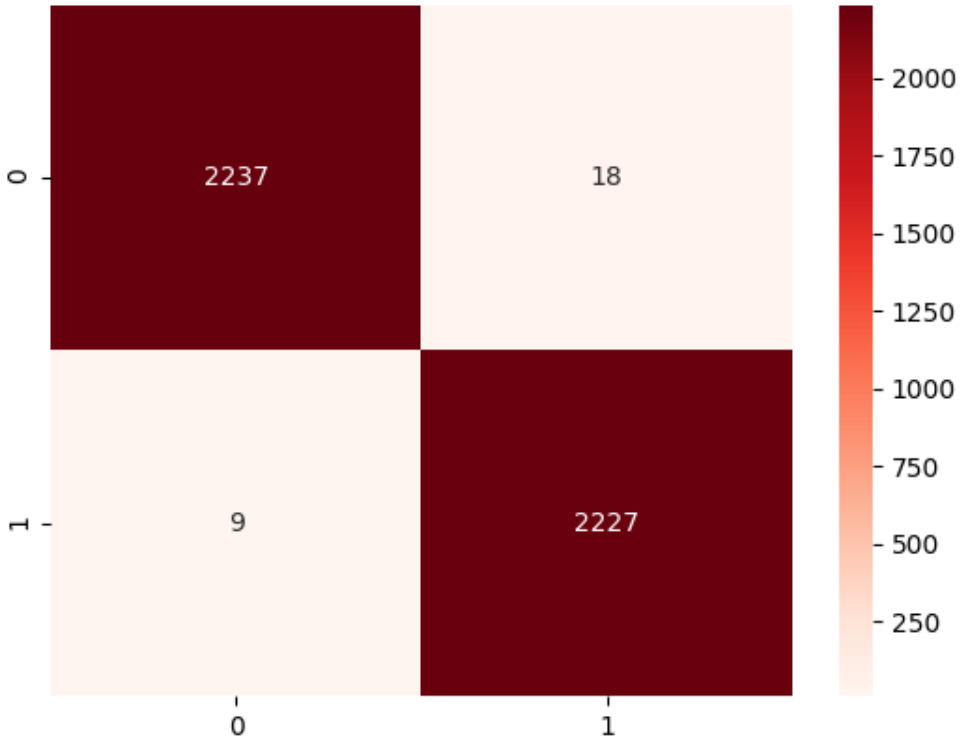

Рисунок Б.6 – Матриця невідповідностей за результатами класифікації підготовленого датасету власним методом. Точність класифікації: 99.4%

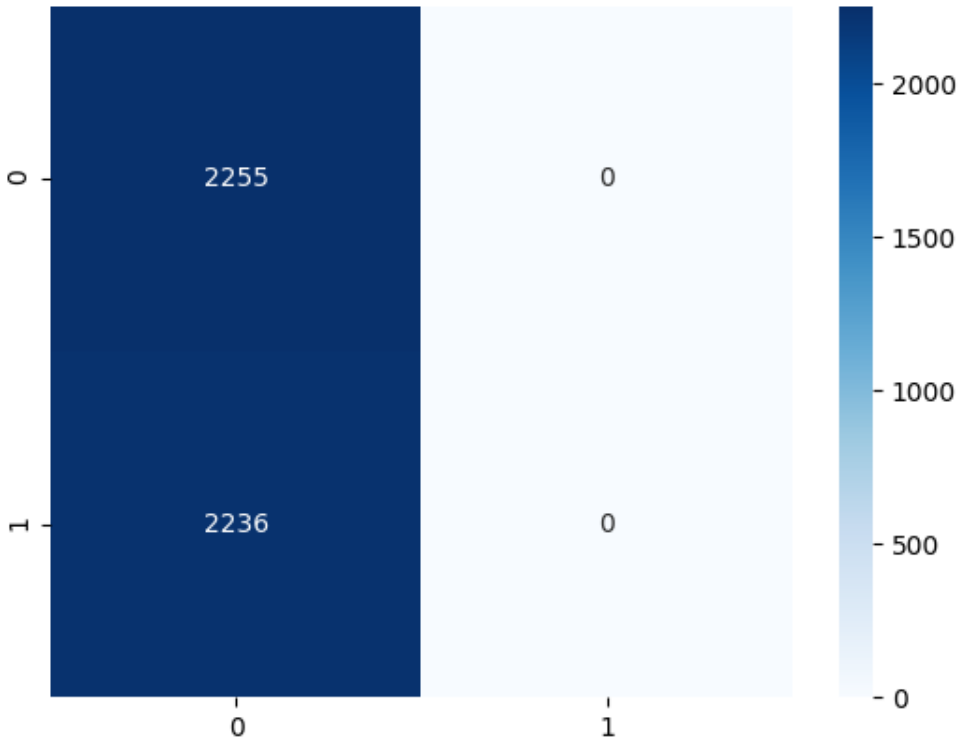

Рисунок Б.7 – Матриця невідповідностей за результатами класифікації підготовленого датасету методом LabelPropagation.Точність класифікації: 50.2%

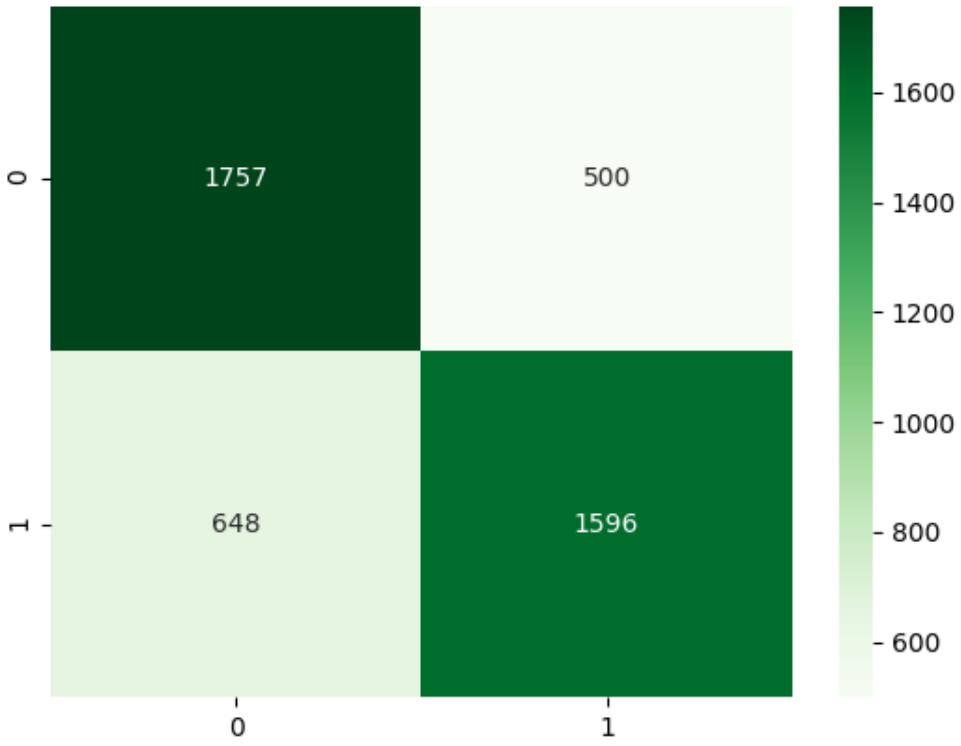

Рисунок Б.8 – Матриця невідповідностей за результатами класифікації підготовленого датасету методом LabelSpreading. Точність класифікації: 74.5%