ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до магістерської роботи
на ступінь вищої освіти магістр

на тему: «**РОЗРОБКА СИСТЕМИ КОНТРОЛЮ ОСАНКИ ЛЮДИНИ ПРИ РОБОТІ ЗА КОМП'ЮТЕРОМ НА ОСНОВІ МЕТОДІВ МАШИННОГО НАВЧАННЯ**»

Виконав: студент 6 курсу, групи ПДМ–61
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності/спеціалізації)

Бріт Я.О.
(прізвище та ініціали)

Керівник Жебка В.В.
(прізвище та ініціали)

Рецензент _____
(прізвище та ініціали)

Київ – 2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти -«Магістр»
Спеціальність підготовки – 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____Негоденко О.В.
"____" _____2022 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТА

БРІТУ ЯРОСЛАВУ ОЛЕГОВИЧУ
(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка методики контролю осанки людини при роботі за комп'ютером на основі методів машинного навчання»

Керівник роботи: Жебка В.В., д.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу від «13» жовтня 2020 року №230.
2. Строк подання студентом роботи «24» грудня 2020 року
3. Вхідні дані до роботи
Науково-технічна література з теорії методів машинного навчання, систем нейронних мереж;
Науково-технічна література з питань, пов'язаних з програмним забезпеченням щодо розпізнавання постаті людини з зображень;

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).

4.1 Методи машинного навчання та нейронні мережі для розпізнавання осанки людини з зображення.
4.2 Вимоги та опис системи.
4.3 Опис проектування системи.
4.4 Опис використаних технологій.

5. Перелік демонстраційного матеріалу (назва основних слайдів)
1. Актуальність проблеми
2. Існуюче програмне забезпечення та методи розпізнавання
3. Принцип роботи інформаційної системи, результати отримані у ході дослідження
4. Практичне застосування створеної системи, оцінка результатів
5. Схема створеного програмного забезпечення, скриншоти програми

6. Дата видачі завдання «02» листопада 2020

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	02.11-07.11	Виконано
2	Формування ідеї дослідження	08.11-10.11	Виконано
3	Створення та навчання моделі для вилучення полів	11.11-18.11	Виконано
5	Концепція прикладного програмного забезпечення	29.11-04.12	Виконано
6	Вступ, висновки, реферат	05.12-12.12	Виконано
7	Попередній захист роботи	22.12	
8	Здача роботи	24.12	

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина магістерської роботи 64 с., 20 рис., 13 джерел.

РОЗПІЗНАВАННЯ ПОСТАНІ, МЕТОДИ МАШИННОГО НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ, TENSORFLOW, GOOGLE CHROME, ДОДАТОК-РОЗШИРЕННЯ

Об'єкт дослідження – стеження за осанкою людини під час роботи за комп'ютером.

Предмет дослідження – додаток-розширення для браузера Google Chrome.

Мета роботи – зменшення негативного впливу сидячого образу роботи на здоров'я людини за рахунок стеження за осанкою людини під час роботи за комп'ютером за допомогою додатку-розширення для веб-браузера Google Chrome та з використанням машинного навчання.

Методи дослідження – методи машинного навчання, відкриті бібліотеки та сервіси, мови програмування, браузер Google Chrome.

У роботі проведено аналіз загальних положень та теорії методів машинного навчання, досліджено основні принципи та методи машинного програмування. На основі дослідження предметної області, було запропоновано власне дослідження у основі якого дослідження результатів окремих методів машинного навчання та їх поєднання у єдиній системі. Проведено огляд теоретичних основ та базових принципів взаємодії з наскрізною платформою побудови та навчання систем машинного навчання та нейронних мереж TensorFlow.

Для реалізації та демонстрації ідеї дипломного проекту створено додаток-розширення для веб-браузера Google Chrome з використання мови веб-розмітки HTML, мови стилів сторінок CSS та мови JavaScript на основі бібліотек TensorFlow, які імпортують попередньо створену та натреновану систему машинного навчання.

Створений додаток є простим у застосуванні, має зручний та зрозумілий інтерфейс, виконує своє основне завдання – стеження за осанкою користувача

комп’ютера під час користування браузером Google Chrome.Отже, розроблено та описано веб-додаток, завданням якого є розпізнавання та групування україномовних документів.

У якості вихідних даних є два набори зображень людини з гарною та поганою осанкою.

Даний додаток може бути використано на всіх комп’ютерах зі встановленим браузером Google Chrome.

Галузь використання – приватне використання окремими користувачами.

ЗМІСТ

ВСТУП

У роботі проведено аналіз загальних положень та теорії методів машинного навчання, досліджено основні принципи та методи машинного програмування. На основі дослідження предметної області, було запропоновано власне дослідження у основі якого дослідження результатів окремих методів машинного навчання та їх поєднання у єдиній системі. Проведено огляд теоретичних основ та базових принципів взаємодії з наскрізною платформою побудови та навчання систем машинного навчання та нейронних мереж TensorFlow.

Для реалізації та демонстрації ідеї дипломного проекту створено додаток-розширення для веб-браузера Google Chrome з використання мови веб-розмітки HTML, мови стилів сторінок CSS та мови JavaScript на основі бібліотек TensorFlow, які імпортують попередньо створену та натреновану систему машинного навчання.

Створений додаток є простим у застосуванні, має зручний та зрозумілий інтерфейс, виконує своє основне завдання – стеження за осанкою користувача комп'ютера під час користування браузером Google Chrome.

Мета дослідження – зменшення негативного впливу сидячого образу роботи на здоров'я людини за рахунок стеження за осанкою людини під час роботи за комп'ютером за допомогою додатку-розширення для веб-браузера Google Chrome та з використанням машинного навчання.

Об'єкт сфери дослідження – стеження за осанкою людини під час роботи за комп'ютером.

Предмет дослідження – додаток-розширення для браузера Google Chrome.

Методи дослідження – методи машинного навчання, відкриті бібліотеки та сервіси, мови програмування, браузер Google Chrome.

1. ТЕОРЕТИЧНА ЧАСТИНА

1.1. Огляд машинного навчання

Проблема пошуку шаблонів у даних є фундаментальною і має тривалу та успішну історію. Наприклад, великі астрономічні спостереження Тихо Браге в 16 столітті дозволили Йоганнесу Кеплеру відкрити емпіричні закони планетарного руху, що, у свою чергу, стало трампліном для розвитку класичної механіки. Подібним чином відкриття закономірностей в атомних спектрах зіграло ключову роль у розвитку та верифікації квантової фізики на початку XX ст. Сфера розпізнавання образів стосується автоматичного виявлення закономірностей у даних за допомогою комп'ютерних алгоритмів та з використанням цих закономірностей для здійснення таких дій, як класифікація даних за різними категоріями.

Розглянемо приклад розпізнавання рукописних цифр, показаний на рисунку 1.1.

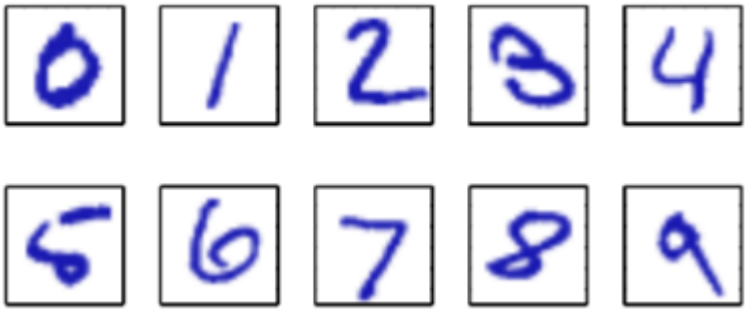


Рисунок 1.1. Приклад набору рукописних чисел для розпізнавання

Кожна цифра відповідає зображенню 28 × 28 пікселів і тому може бути представлена вектором x, що містить 784 дійсних чисел. Мета полягає в тому, щоб побудувати машину, яка візьме такий вектор x як вхідний сигнал та видасть визначення цифр 0...9 як вихід.

Це нетривіальна проблема через велику різноманітність почерків. Для розв'язку такої задачі можуть бути використані правила або евристика для розрізнення цифр на основі форми штрихів, але на практиці такий підхід призводить до розширення правил та виникнення винятків із правил, і незмінно дає погані результати.

Набагато кращі результати можна отримати, застосувавши підхід машинного навчання, у якому використовується великий набір N цифр $\{x_1, ..., x_N\}$, що називається навчальним набором, для налаштування параметрів адаптивної моделі. Категорії цифр у навчальному наборі відомі заздалегідь, як правило, шляхом огляду їх окремо та маркування вручну. Ми можемо виділити категорію цифри, використовуючи цільовий вектор t, який представляє ідентичність відповідної цифри. Варто зауважити, що є один такий цільовий вектор t для кожного цифрового зображення x.

Результат виконання алгоритму машинного навчання можна виразити як функцію $y(x)$, яка приймає нове цифрове зображення x як вхід і генерує вихідний вектор y, закодований так само, як і цільові вектори. Точна форма функції $y(x)$ визначається на етапі навчання на основі даних навчання. Після того як модель навчена, вона може визначити ідентичність нових цифрових зображень, які, як кажуть, складають тестовий набір. Здатність правильно класифікувати нові приклади, що відрізняються від тих, що використовуються для навчання, відома як узагальнення. У практичних застосуваннях мінливість векторів вхідних даних буде такою, що навчальні дані можуть складати лише крихітну частину всіх можливих вхідних векторів, і тому узагальнення є центральною метою розпізнавання образів.

Для більшості практичних застосувань вхідні змінні зазвичай попередньо обробляються, щоб перетворити їх у деякий новий простір змінних, де проблему розпізнавання образів буде легше вирішити. Наприклад, у проблемі розпізнавання цифр зображення цифр зазвичай перекладаються та

масштабуються так, щоб кожна з цифр міститься в квадраті фіксованого розміру. Це значно зменшує мінливість всередині кожного класу цифр, тому що розташування та масштаб усіх цифр тепер є однаковими, що значно полегшує подальший алгоритм розпізнавання образів у розпізнаванні різні класи. Цей етап попередньої обробки іноді буває також називається вилученням функцій. Зверніть увагу, що нові дані тесту необхідно попередньо обробити за допомогою ті ж кроки, що і дані навчання.

Попередня обробка також може бути виконана для прискорення обчислень. Наприклад, якщо метою є розпізнавання облич у режимі реального часу у відеопотоці з високою роздільною здатністю, комп'ютер повинен обробляти величезну кількість пікселів на секунду і подавати їх безпосередньо до складного алгоритму розпізнавання образів, що може бути неможливим для обчислень. Натомість мета полягає в тому, щоб знайти функції, які швидко обчислюються, і тим не менше також зберігають корисну інформацію, що дозволяє розрізняти обличчя від не-облич. Ці функції потім використовуються як вхідні дані для алгоритмів розпізнавання шаблонів.

Додатки, в яких навчальні дані містять приклади вхідних векторів разом з відповідними цільовими векторами відомі як завдання *навчання під наглядом*. Такі випадки, як приклад розпізнавання цифр, в якому мета полягає у призначенні кожному вхідному вектору однієї з кінцевої кількості дискретних категорій, називаються *класифікаційними завданнями*. Якщо бажаний результат складається з однієї або декількох безперервних змінних, то завдання називається *регресією*. Прикладом проблеми регресії може бути передбачення виходу в процесі хімічного виробництва, в якому вхідними даними є концентрації реагентів, температури та тиск.

В інших проблемах розпізнавання образів навчальні дані складаються з набору векторів x вхідних даних без відповідних цільових значень. Такі завдання називаються завданнями *без нагляду навчання*. Мета у таких завдань

може полягати у виявленні груп подібних прикладів у межах даних, де це називається *кластеризацією*, або для визначення розподілу даних у межах простору вхідних даних, відомого як *оцінка щільності*, або для проектування даних з високомірного простір до двох -трьох вимірів з метою *візуалізації*.

1.2. Машинне навчання у програмуванні

Термін «машинне навчання» у загальному означає автоматизоване визначення змістовних шаблонів у тому чи іншому наборі даних.

За останні десятиліття машинне навчання стало звичним інструментом у більшості завдань, які вимагають роботи з великими наборами даних. Вони зустрічаються повсякчас – пошукові двигуни вивчають як надати найкращий пошуковий результат; анти-спам забезпечення вивчає як фільтрувати електронну пошту; кредитні картки захищені програмним забезпеченням, що навчається розпізнавати злочинні наміри; цифрові камери вчяться розпізнавати обличчя; розумні асистенти на мобільних пристроях вчяться розпізнавати голосові команди. До цього машинне навчання широко використовується у наукових застосунках в галузях біотехнологій, медицині, астрономії тощо.

Загальною особливістю цих додатків є те, що на відміну від звичайних комп’ютерів, інженер програмного забезпечення не може надати повний та достатньо деталізований опис того, як опрацьовувати такі специфічні та складні шаблони даних та інформації. Слідуючи прикладам живих істот, які досягають певних знань та навичок у процесі навчання, засоби машинного навчання наділяються можливістю «навчатися» та «адаптуватися».

Розглянемо більш детально один з прикладів, наведених вище, а саме роботу програмного забезпечення, яке фільтрує небажані та потенційно шкідливі листи на електронній пошті. Очевидним та простим рішенням було б запам’ятовування машиною листів, які користувач попередньо відмітив як

спам. Коли новий лист приходить на пошту, машина порівнює новий лист з визначеним набором спам-листів. Якщо наявний збіг, то лист буде поміщено у категорію спаму, якщо ж ні – лист вільно потрапляє до папки вхідних листів.

Однак, для виконання подібного завдання можна використати машинне навчання. При цьому машина зможе перейти від одиночних прикладів спаму до узагальнення та ширшого впорядкування. Щоб досягти впорядкування у задачі фільтрування спам-листів, машина може не просто фільтрувати лист за набором користувача, а переглядати їх та виділити певний набір ознак та ключових слів, що зустрічались у спам-листах раніше. Потім, коли буде отримано нового листа, машина перевірятиме, чи містить такий лист потенційно небезпечні слова, та відмічатиме лист відповідною позначкою. Така система потенційно зможе виділяти листи навіть з прихованим спам-змістом.

Тож чому нам необхідно машинне навчання та чому не є релевантним пряме програмування машин на виконання конкретної задачі?

Два аспекти заданого питання наводять справедливість ідеї про користь використання програм, що вчяться та покращуються на основі власного «досвіду» – це аспект складності можливих задач та необхідність адаптування.

Складність можливих задач можна розділити на дві категорії:

- Задачі, які виконуються людиною. Є безліч задач, які щоденно та рутино виконуються людиною. Але чим більше часу проходить при виконанні таких задач, тим менш об’єктивною стає людина при оцінці того, як такі задачі мають бути виконані у загальному та машиною окремо. Наприклад, це може бути керування автомобілем, розпізнавання мовлення, аналіз зображень. У виконанні таких та подібних завдань, сучасні системи машинного навчання, які навчаються та використовують власний набутий досвід, показують більш, ніж задовільні показники при умові надання таким системам достатньої кількості даних для тренування.

- Задачі, які знаходяться за межами людських здібностей. Інша широка галузь завдань, які вигідно користуються перевагами машинного навчання, пов’язана з аналізом великих та складних наборів даних. Такі набори можуть зустрічатися як набори астрономічних даних, у прогнозуванні погоди, у аналізі генетичних даних, при роботі пошукових машин, при трансформації медичних архівів у практичні знання. З накопиченням все більших обсягів цифрової інформації, стає зрозумілим, що такі обсяги зберігають цінну інформацію, при цьому такі обсяги занадто великі та складні для аналізу людиною. Навчання машин визначати змістовні шаблони у складних наборах даних є дуже перспективним напрямом, особливо з урахуванням обчислювальних можливостей комп’ютерів, що постійно збільшуються.

Необхідність адаптуватися. Цей аспект полягає у тому, що звичні програми залишаються майже незмінними від моменту їх встановлення. Однак багато задач можуть змінюватися з часом або залежно від потреб конкретного користувача. Інструменти машинного навчання дозволяють створювати програми, чия поведінка та робота можуть адаптуватися відповідно до даних на вході, так само як адаптуватися до змін у середовищі, з яким вони співпрацюють. Типовими прикладами можуть стати програми розпізнавання рукописного тексту, де одна й та сама програма адаптується до почерку того чи іншого користувача; програми розпізнавання спам-листів, які адаптуються до змін у характері побудови спам-листів; програми розпізнавання голосу та інші [5].

1.3. Основні парадигми машинного навчання

Машинне навчання, безперечно, досить широка область для досліджень. Як наслідок, область машинного навчання поділяється на декілька підрозділів, які займаються різними видами завдань навчання. Для початку, опишемо чотири основні параметри, за якими поділяються парадигми машинного навчання.

Перший параметр – навчання з або без наглядача. Оскільки навчання передбачає взаємодію між учнем та навколишнім середовищем, можна розділити навчальні завдання відповідно до характеру цієї взаємодії. Перше, на що слід звернути увагу – це різниця між навчанням з наглядачем та без. Як приклад розглянемо задачу визначення спам-листів на електронній пошті та задачу виявлення аномалій.

Для виявлення спаму ми розглядаємо початкові умови, при яких учень отримує тренувальний набір листів, які заздалегідь відмічено як спам або не спам. В основі такого навчання полягає те, що учень повинен визначити для себе правило, відповідно до якого наступні лист будуть відмічені як спам чи не спам.

На відміну від цього, у задачі виявлення аномалій, учень отримує лише великий набір листів без міток, а його задача полягає в тому, щоб навчитися виявляти «незвичайні» листи.

Більш абстрактно, розглядаючи навчання як процес «використання досвіду для здобуття експертизи», контрольоване навчання описує сценарій, у якому «досвід», тобто навчальний приклад, містить значну інформацію (скажімо, мітки спаму/не спаму), якої немає у невидимих «тестових прикладах», до яких слід застосувати набутий досвід. У цьому випадку набуті знання спрямовані на те, щоб передбачити, якої інформації бракує для тестових даних.

У таких випадках ми можемо думати про навколишнє середовище як про вчителя, який «контролює» учня, надаючи додаткову інформацію (позначки).

Однак у навчанні без нагляду немає відмінностей між навчальними та тестовими даними. Учень обробляє вхідні дані з метою придумати якийсь узагальнений або стислий варіант цих даних. Типовим прикладом такого завдання є групування набору даних у підмножини подібних об'єктів.

Другий параметр – активні та пасивні учні. Навчальні парадигми можуть розрізнятися за роллю, які виконує учень. Розрізняють активних учнів та пасивних учнів. Активний учень взаємодію з середовищем під час тренування, наприклад за допомогою запитань або експериментів, в той час як пасивний учень лише спостерігає дані, які були йому надані, без будь-якого впливу чи додаткової обробки цих даних. На прикладі з фільтрацією спам-листів, пасивний учень просто очікує, коли користувач відмітить той чи інший лист як спам або не спам. Активний учень міг би власноруч обрати певний лист або створити лист самостійно та запропонувати користувачу відмітити його як спам/не спам, щоб покращити своє розуміння того, що насправді є спамом.

Третій параметр – корисність вчителя. Коли хтось думає про людське навчання, про дитину вдома чи у школі, у процесі часто бере участь корисний вчитель, який намагається надати учню інформацію, яка є найбільш корисною для досягнення цілей навчання. На відміну від цього, коли вчений досліджує природу, його навколишнє середовище, яке виконує роль вчителя, можна вважати пасивним - яблука падають, зірки світять, а дощ падає без огляду на потреби учня. У машинному навчанні моделюють такі сценарії навчання, приймаючи, що навчальні дані (або досвід учня) породжуються якимось випадковим процесом. Це основний матеріал у галузі «статистичного навчання». Нарешті, навчання також виникає, коли вхідні дані учня генеруються «змагальним» вчителем. Це може мати місце у прикладі фільтрації спаму (якщо спамер докладає зусиль ввести в оману розробника фільтрації

спаму) або навчанні виявляти шахрайство. Модель «змагального» вчителя також використовується як найгірший сценарій, коли немає більш м'якого налаштування. Якщо учень зможе вчитися проти вчителя-суперника, такий учень гарантовано досягнете успіху у взаємодії з будь-яким учителем.

Останній параметр – це відмінність між ситуаціями, в яких учень повинен реагувати в режимі онлайн протягом усього процесу навчання, і ситуаціями, в яких учень повинен залучати набуті знання лише після того, як мав можливість обробити великі обсяги даних. Наприклад, біржовий брокер повинен приймати щоденні рішення, виходячи з досвіду, накопиченого до цього часу. З часом він може стати експертом, але міг би припуститися критичних помилок у процесі. Навпаки, у багатьох налаштуваннях інтелектуального аналізу даних, учень - майнер даних - має великі обсяги навчальних даних для навчання та аналізу, перш ніж робити висновки [5].

1.3.1. Вибір та оцінка моделі

Загалом, частка неправильно класифікованих зразків до загальної кількості вибірок називається частотою помилок, тобто якщо a з m вибірок неправильно класифіковані, то частота помилок дорівнює E = a/m. Відповідно, 1 - a/m називають точністю. Загалом, різниця між результатами, передбаченими учнем, і висновком називають помилкою. Помилка, обчислена на навчальному наборі, називається помилкою навчання або емпіричною помилкою, а помилка, обчислена за новими вибірками, називається помилкою узагальнення. Очевидно, ми хочемо мати учня з невеликою помилкою узагальнення.

Однак, оскільки деталі нових зразків невідомі на етапі навчання, ми можемо лише спробувати мінімізувати емпіричну похибку на практиці. Досить часто ми отримуємо учнів, які добре працюють на навчальному наборі з невеликою або навіть нульовою емпіричною похибкою, тобто 100% точністю.

Однак, чи є вони учням, що необхідні для роботи у машинному навчанні? На жаль, такі учні є поганими у більшості випадків.

Хороші учні, яких справді необхідні, – це ті, хто вчиться добре на нових зразках. Отже, хороші учні повинні вчитися загальним правилам з навчальних прикладів так, що вивчені правила застосовуються до всіх потенційних зразків. Однак, коли учень вивчає приклади навчання «надто добре», ймовірно, що деякі особливості навчальних прикладів беруться за загальні властивості, які матимуть усі потенційні вибірки, що призводить до зниження продуктивності узагальнення. У машинному навчанні це явище відоме як надмірна комплектація, і відомо протилежне як недооснащення, тобто учень не зумів вивчити загальне властивості навчальних прикладів. Рисунок 1.2 ілюструє різницю між надмірним і недооснащеним.

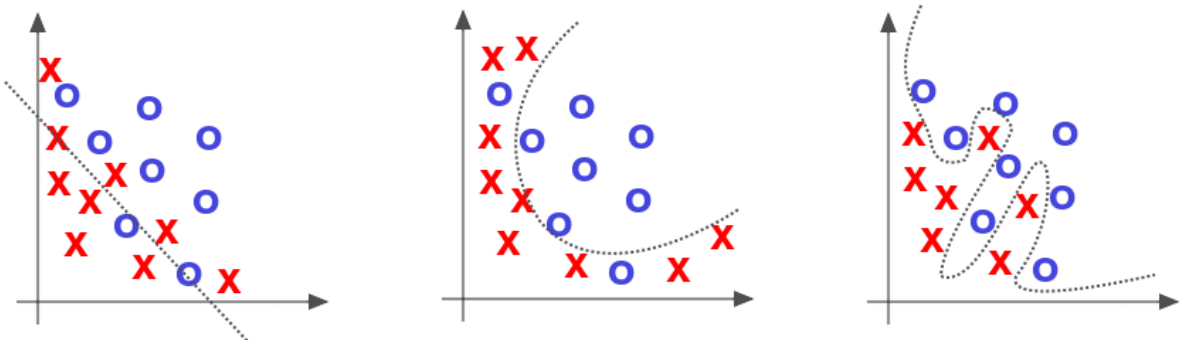


Рисунок 1.2. Зображення можливих результатів навчання – недооснащення, приблизно точного результату та надмірної підготовки

На рисунку 1.2 зображено приклад недооснащення (графік ліворуч) – результат недостатньо точний; очікуваний результат, який приблизно відповідає дійсності (графік по центру); та надмірна підготовка – результат занадто якісний (графік праворуч).

Серед багатьох можливих причин, надмірно сильне навчання є загальною причиною надмірної підготовки, оскільки такі учні можуть дізнатися незагальні особливості навчальних прикладів.

Навпаки, недооснащення зазвичай пояснюється слабкою здатністю до навчання. На практиці недооснащення порівняно легко подолати. Наприклад, можливо зробити більше розгалужень у навчанні на дереві рішень або додати більше додаткових епох навчання у навчанні нейронних мереж.

Однак, надмірна підготовка - це фундаментальна складність машинного навчання та майже кожний алгоритм навчання впровадив деякі механізми боротьби з надмірною підготовкою.

Тим не менш, необхідно усвідомлювати, що надмірного оснащення не уникнути, і все, що можна зробити, це зменшити ризик цього. Цей аргумент можна обґрунтувати таким чином. Проблеми машинного навчання часто є NP - важкими або навіть складнішими, але практичні алгоритми навчання мають завершувати навчання в поліноміальному часі. Отже, якщо надмірності можна уникнути, то необхідно мінімізувати її емпіричну помилка, що призведе до оптимального рішення, і тому ми маємо конструктивний доказ P = NP. Іншими словами, надмірне оснащення неминуче, поки покладено P ≠ NP.

На практиці часто існує кілька кандидатур алгоритмів навчання, і навіть один і той же алгоритм навчання може створювати різні моделі за різних наборів параметрів. Потім, який алгоритм навчання необхідно вибрати та які набори параметрів необхідно встановити? Ця проблема називається вибором моделі. Ідеальне рішення - оцінити всі кандидати моделей і вибрати ту, яка має найменшу помилку узагальнення. Однак, як згадувалося раніше, ми не можемо отримати помилку узагальнення безпосередньо, поки емпірична помилка страждає від надмірного оснащення.

Оцінити помилку узагальнення можна за допомогою тестування. Для цього ми використовуємо набір тестів для оцінки здатності учня класифікувати

нові зразки та використовуємо помилку тестування як наближення до помилки узагальнення. Загалом, вважається, що тестові зразки є незалежними та ідентичними вибраними з розподілу правдивої вибірки. Слід зауважити, що тестовий набір та навчальний набір повинні бути максимально виключаючими один одного таким чином, що тестові зразки слід мають уникати появи у навчальному наборі або будь-яким чином бути використаними у навчальному процесі.

Чому тестові зразки не повинні з’являтися у навчанні встановити? Щоб зрозуміти це, розглянемо наступний сценарій. Припустимо, ми використовуємо один і той же набір із десяти питань як для вправи, так і для іспиту, тоді чи іспит справді відображає результати навчання студентів? Відповідь "ні", тому що деякі учні можуть отримати хороші оцінки, навіть якщо вони знають, як вирішити ці десять питань. Аналогічно, узагальнююча здатність, якої ми бажаємо надати моделі, така сама, як і здатність студентів навчатися і здобувати знання. Відповідно, приклади навчання відповідають вправам, а зразки тестування відповідають екзамену. Отже, оцінка може бути надто оптимістичною, якщо тестові зразки вже побачені у навчальному процесі.

Однак, маючи єдиний набір даних зразків D = {(x1, y1),(x2, y2),..., (xm, ym)}, як можна проводити навчання та тестування? Відповідь полягає у створенні навчального набору S та тестування множини T з набору даних D.

Кілька загальновживаних методів буде оглянуто далі.

Метод утримання розбиває набір даних D на дві непересічні підмножини: один як навчальний набір S, а інший як тестувальна множина T, де $D = S \cup T$ і $S \cap T = \emptyset$. Модель навчається на навчальному наборі S, а потім обчислюється помилка тестування на тестувальному наборі T як оцінка помилки узагальнення. Взявши за приклад проблеми бінарної класифікації, нехай D буде набором даних з 1000 вибірок, і його розділено на навчальний набір S з 700 зразками та тестовий набір T з 300 зразками. Після навчання на S, припустимо,

що модель неправильно класифікувала 90 зразків на T, тоді ми маємо коефіцієнт помилок (90/300) × 100% = 30%, і відповідно, точність 1 - 30% = 70%.

Перехресна перевірка розбиває набір даних D на k непересічних підмножин з подібними розмірами, тобто D = D1 ∪ D2 ∪ ⋯ ∪ Dk, Di ∩ Dj = ∅ (i ≠ j). Як правило, кожна підмножина Di намагається зберегти початковий розподіл даних за допомогою стратифікованої вибірки. На кожній ітерації перехресної перевірки ми використовуємо об'єднання k - 1 підмножин як навчальний набір, щоб встановити для навчання моделі, а потім використати залишкову підмножину як тестовий набір для оцінки моделі. Повторюємо цей процес k разів і використовуємо кожну підмножину як тестовий набір точно один раз. Нарешті, маємо в середньому по k випробувань для отримання результату оцінки. Оскільки стабільність та вірність перехресної перевірки значною мірою залежать від значення k, вона також відома як k-кратна перехресна перевірка. Найбільший загальновживане значення k становить 10 та відповідний метод називається 10-кратною перехресною перевіркою. Інші загальні значення k включають 5 і 20. Рисунок 1.3 ілюструє ідею 10-кратної перехресної перевірки.

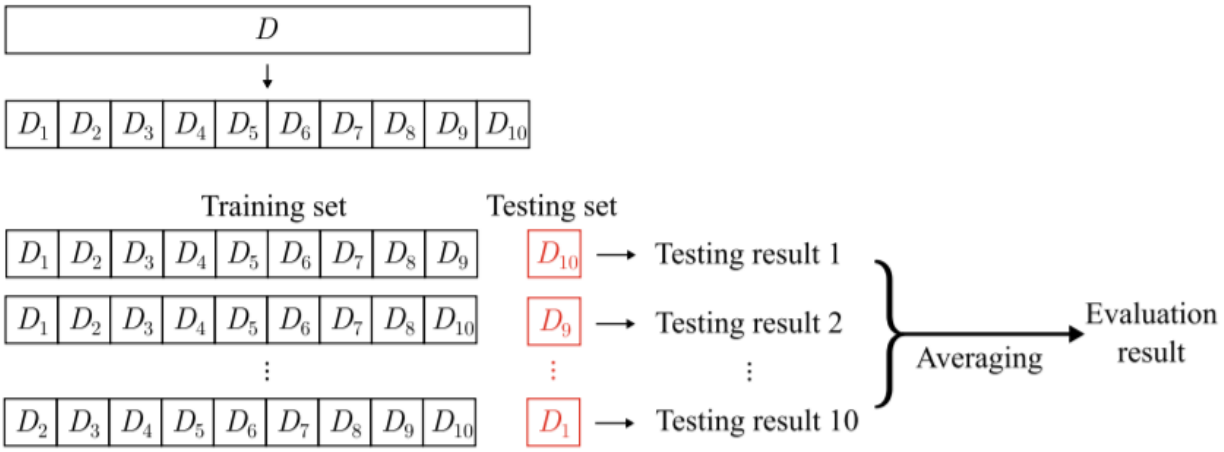


Рисунок 1.3. Зображення моделі 10-кратної перехресної перевірки

Ми хочемо оцінити модель, навчену D. Однак, незалежно від того, чи використовуємо ми затримку або перехресну перевірку, навчальний набір завжди менший за D. Отже, упередженість оцінки неминуча через різницю в розмірах між набором тренувань та D. Чи можливо зменшити вплив невеликого навчального набору, але при цьому бути ефективним для обчислень?

Одним із рішень є бутстрепінг, який використовує техніку вибіркового завантаження. З огляду на набір даних D, що містить m вибірок, бутстрепінг вибирає зразок D` з даних шляхом випадкового відбору одного зразка з D, копіює його в D`, а потім поміщає його назад у D, щоб у нього ще був шанс бути обраним наступного разу. Повторення цього процесу m разів призводить до бутстрепінгового набору даних D`, що містить m вибірок. Через заміну деякі зразки з D можуть не відображатися в D`, інші можуть з'являтися більше одного разу. Зробимо оцінку: ймовірність того, що зразок не відберуть у m раундах, становить $(1 - \frac{1}{m})^m$, а отже прийняття ліміту дає

$$\lim_{m\to\infty}\left(1-\frac{1}{m}\right)^m = \frac{1}{e} \approx 0.368, \qquad (1.1)$$

що означає, що приблизно 36,8% вихідних зразків не відображаються у наборі даних D`.

Потім можна використовувати D` як навчальний набір і D\D` в якості набору тестування, що оціночна модель і фактична модель, яку ми хочемо оцінити на D, використовують m прикладів навчання. Крім того, у нас ще є окремий набір тестування, що містить приблизно 1/3 оригінальних зразків, які не використовуються для навчання. Результат оцінки, отриманий за допомогою такого підходу називається оцінкою "поза сумки".

Бутстрапінг є особливо корисним, коли набір даних є невеликим, або коли немає ефективного способу розподілу навчальних і тестувальних наборів. Крім того, бутстрапінг може створювати кілька наборів даних, які можуть бути корисними для таких методів, як ансамблеве навчання. Проте, оскільки

початковий розподіл даних змінився під час бутстрапінгу, оцінка також є упередженою. Тому, коли у нас є велика кількість даних, часто використовуються затримка та перехресна перевірка.

Більшість алгоритмів навчання мають параметри для встановлення, а різні набори параметрів часто призводять до моделей зі значними відмінностями виконання. Отже, оцінка та вибір моделі - це не лише вибір алгоритмів навчання, а й вибір конфігурації параметрів. Процес пошуку правильних параметрів називається налаштуванням параметрів.

Можна подумати, що немає суттєвої різниці між налаштуванням параметрів та вибором алгоритму: кожне налаштування параметрів веде до однієї моделі, і ми вибираємо ту, яка надає найкращі результати як остаточна модель. Ця ідея в основному слушна; проте є одна проблема: оскільки параметри часто переоцінюються, неможливо спробувати всі набори параметрів. Тому на практиці зазвичай встановлюється діапазон і розмір кроку для кожного параметру, наприклад, діапазон [0, 0.2] та розмір кроку 0,05, що призведе лише до п’яти варіантів параметрів-кандидатів. Такий компроміс між обчислювальними витратами та якістю оцінки робить можливим навчання, хоча вибраний параметр є зазвичай не оптимальним. Насправді навіть після такого компромісу налаштування параметрів все ще може бути досить складним. Можемо зробити просту оцінку. Припустимо, що алгоритм має три параметри, і кожен враховує лише п’ять значень–кандидатів, отже нам потрібно оцінити 5^3 = 125 моделей для кожної пари навчальних і тестувальних наборів. Потужні алгоритми навчання часто мають цілком достатньо багато параметрів, які потрібно налаштувати, що призводить до великого навантаження з налаштування параметрів. Якість налаштування параметрів також часто життєво важлива у реальних застосунках [6].

1.4. Огляд наукових робіт та статей подібної тематики та напрямку

Перед виконанням даної магістерської роботи, було проведено пошук та дослідження наукових робіт та статей подібної тематики та області дослідження. Далі коротко оглянемо деякі з цих робіт.

«Detection of sitting posture using hierarchical image composition and deep learning» – стаття литовсько-польских науковців, яка досліджує визначення сидячої постаті людини використовуючи ієрархічну композицію зображень та глибоке навчання. У роботі пропонується глибока рекурентна ієрархічна мережа побудована на MobileNetV2, яка надає більшу гнучкість шляхом зменшення або ліквідації похибок, які пов’язані з низькою роздільною здатністю чи неповним обсягом обхвату людської постаті на зображенні. У роботі досягнена точність 91.47% при частоті кадрів 10fps [1].

«Classification of Children’s Sitting Postures Using Machine Learning Algorithms» – стаття шанхайських науковців, ідея якої полягає у використанні п’яти різних методів машинного навчання для визначення сидячої постаті людини та порівнянні їх результатів з подальшим визначення найточнішого методу. У роботі розглянуті такі методи машинного навчання - Hidden Markov Models, Naïve Bayes classifier, decision tree, multinomial logistic regression, and support vector machine. Також були використані різні класифікаційні методи, ефективність яких визначалась крос-валідаційним шляхом. У роботі досягнена точність 95.3% [2].

«Sitting Posture Monitoring System Based on a Low-Cost Load Cell Using Machine Learning» – стаття корейських науковців, ідея якої полягає у використанні різних методів класифікації зображень та методів машинного навчання для визначення сидячої постаті людини та досягнення якомога точніших результатів. Додатково у роботі створено систему сенсорів для покращення точності результатів. Система SPMS (Sitting posture monitoring

systems) являє собою набір спеціальних сенсорів, які відслідковують сидячу постать людини, доповнюючи та уточнюючи дані зображень. Сенсори кріпляться на спинку або сидіння стільця. У роботі досягнена середня точність у 97.20% та максимальна у 97.94%[3].

2. НАУКОВО-ДОСЛІДНА ЧАСТИНА

2.1. Основні методи, алгоритми та ідея дослідження

2.1.1. Загальна теорія основних методів машинного навчання

Далі розглянемо основні методи машинного навчання та їх математичні основи, а саме такі :

- регресія;
- класифікація;
- кластеризація.

Регресія.

Мета регресії – передбачити значення однієї чи декількох безперервних цільових змінних t з урахуванням значення D-вимірного вектора x вхідних змінних. Поліном є конкретним прикладом широкого класу функцій, які називаються моделями лінійної регресії, які мають спільну властивість бути лінійними функціями регульованих параметрів. Найпростіша форма моделі лінійної регресії також є лінійними функціями вхідних змінних. Однак ми можемо отримати набагато більш корисний клас функцій, приймаючи лінійні комбінації фіксованої множини нелінійних функцій вхідних змінних, відомих як базисні функції. Такі моделі є лінійними функціями параметрів, що дає їм прості аналітичні властивості, і все ж може бути нелінійним щодо вхідних змінних.

Враховуючи набір навчальних даних, що містить N спостережень $\{x_n\}$, де $n = 1...N$, разом з відповідними цільовими значеннями $\{t_n\}$, метою є передбачити значення t для нового значення x. У найпростішому підході це можна зробити безпосередньо побудувавши відповідну функцію y(x), значення якої для нових входів x складають передбачення для відповідних значень t. Загальніше, з точки зору перспективи, ми прагнемо моделювати прогнозний розподіл p(t|x),

оскільки це виражає невизначеність щодо значення t для кожного значення x. З цього умовного розподілу ми можемо зробити прогнози t для будь-якого нового значення x таким чином, щоб мінімізувати очікуване значення відповідно обраної функції втрат.

Хоча лінійні моделі мають суттєві обмеження як практичні методи розпізнавання образів, особливо для задач, що стосуються вхідних просторів високої розмірності, вони мають прийнятні аналітичні властивості і складають основу для більш складних моделей.

Найпростіша лінійна модель регресії є такою, що включає лінійну комбінацію вхідних змінних, тобто

$$y(x, w) = w_0 + w_1 x_1 + \cdots + w_D x_D, \tag{2.1}$$

де $x = (x_1, \ldots, x_D)^T$. Така форма запису також відома як лінійна регресія. Ключова властивість цієї моделі полягає в тому, що це є лінійною функцією параметрів $w_0, \ldots, w_D$. Водночас це також лінійна функція вхідних змінних x_i, що наводить на значні обмеження даної моделі. Однак клас таких моделей можна розширити, враховуючи лінійні комбінації фіксованих нелінійних функцій вхідних змінних у вигляді

$$y(x, w) = w_0 + \sum_{j=1}^{M-1} w_j \varphi_j(x), \tag{2.2}$$

де $\varphi_j(x)$ називають базисною функцією. Надаючи індексу j максимального значення M – 1, загальне кількість параметрів у такій моделі буде M.

Параметр w_0 дозволяє будь-яке фіксоване зміщення в даних та іноді називається параметром зміщення. Також іноді зручно визначати додаткову штучну базисну функцію $\varphi_0(x) = 1$ таким чином, що

$$y(x, w) = w_0 + \sum_{j=1}^{M-1} w_j \varphi_j(x) = w^T \varphi(x), \tag{2.3}$$

де $w = (w_0, \dots, w_{M-1})^T$ та $\varphi = (\varphi_0, \dots, \varphi_{M-1})^T$. У багатьох практичних застосуваннях розпізнавання образів застосовується певна форма фіксованої попередньої обробки, або вилучення функції до вихідних змінних даних. Якщо вихідні змінні містять вектор x, то ознаки можна виразити через базисні функції $\{\varphi_j(x)\}$.

Використовуючи нелінійні базисні функції, ми дозволяємо функції y(x, w) бути нелінійною функцією вхідного вектора x. Функції виду (2.3) називаються лінійними моделями, тому що ця функція є лінійною за w. Саме ця лінійність параметрів значно спростить аналіз цього класу моделей. Однак це також призводить до деяких суттєвих обмежень.

Приклад поліноміальної регресії, є особливим приклад цієї моделі, в якій є одна вхідна змінна x, а базисні функції мають вигляд степенів x так, що $\varphi_j(x) = x_j$. Одне обмеження поліноміальних базисних функцій полягає у тому, що вони є глобальними функціями вхідної змінної, тому зміни в одному регіоні вхідного простору впливають на всі інші регіони. Це можна вирішити поділом вхідного простір вгору по областях і вписуванням різних поліном у кожну область, що є підходом до сплайнових функцій.

Існує багато інших можливих варіантів базових функцій, наприклад

$$\varphi_j(\mathrm{x}) = exp\left\{-\frac{(x - \mu_j)^2}{2s^2}\right\}, \qquad (2.4)$$

де μ_j визначає розташування базисних функцій у вхідному просторі, а параметр s - їх просторовий масштаб. Зазвичай їх називають "гаусовими" базисними функціями, хоча слід зазначити, що вони не вимагають імовірнісної інтерпретації, і, зокрема, коефіцієнт нормалізації не має значення оскільки ці базисні функції будуть множитися на адаптивні параметри w_j.

Іншим варіантом є сигмоїдні базисні функції вигляду

$$\varphi_j(\mathrm{x}) = \sigma\left(\frac{x - \mu_j}{s}\right), \tag{2.5}$$

де σ(a) - логістична сигмоїдна функція, визначена як

$$\sigma(a) = \frac{1}{1 + \exp(-a)}. \tag{2.6}$$

Крім того, ми можемо використовувати функцію "tanh", оскільки це пов'язано з сигмоїдом через tanh(a) = 2σ (a) - 1, і тому загальна лінійна комбінація сигмоподібні функції еквівалентні загальній лінійній комбінації функцій "tanh".

Ще одним можливим вибором базисної функції є базис Фур'є, що призводить до розширення синусоїдальних функцій. Кожна базова функція представляє певну частоту і має нескінченну просторову протяжність. Натомість базові функції, які локалізовані до кінцевих областей вхідного простору обов'язково включають спектр різних просторових частот. У багатьох додатках для обробки сигналів цікаво розглянути базові функції, локалізовані як у просторі, так і на частоті, що призводить до класу функції, відомі як вейвлети. Вони також визначаються як взаємно ортогональні, щоб спростити їх застосування. Вейвлети найбільш застосовні, коли вхідні значення актуальні на звичайній решітці, такі як послідовні точки часу у часовій послідовності, або пікселі на зображенні.

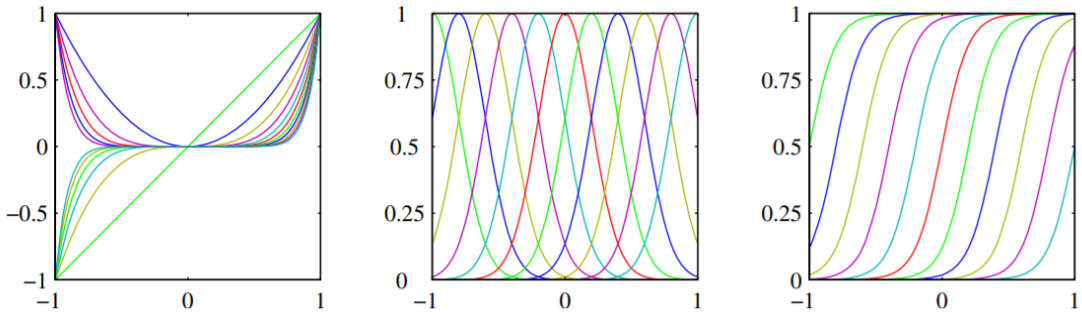


Рисунок 2.1. Приклади графіків базисних функцій

На рис. 2.1 зображено приклади базисних функцій, що показують поліном зліва, гауссів (формула 2.4) у центрі, а сигмоїдальна форма – праворуч [7].

Класифікація

Мета класифікації – взяти вхідний вектор x і віднести його до одного з K дискретних класів C_k, де k = 1, ..., K. У найпоширенішому сценарії класи вважаються роз'єднаними, так що кожен вхід є співвіднесеним до одного і лише одного класу. Таким чином, вхідний простір поділяється на регіони рішень, кордони яких називаються межами прийняття рішень або поверхнями прийняття рішень. Далі буде розглянуто лінійні моделі класифікації, під якими мається на увазі, що поверхні прийняття рішень є лінійними функціями вхідного вектора x, отже, визначені за (D-1)-вимірними гіперплощинами всередині D-вимірного вхідного простору. Множини даних, класи яких можуть бути розділені точно лінійними поверхнями прийняття рішень, називають лінійно роздільними.

Для задач регресії цільовою змінною t був просто вектор дійсних чисел, значення яких необхідно передбачити. У разі класифікації існують різні способи використання цільових значень для представлення міток класів. Для ймовірнісних моделей найзручнішим, у разі задач двох класів, є двійкове представлення, для якого існує єдина цільова змінна $t \in \{0, 1\}$ така, що t = 1 представляє клас C_1 і t = 0 представляє клас C_2. Ми можемо інтерпретувати значення t як ймовірність того, що клас є C_1, при цьому значення ймовірності приймають лише крайні значення 0 і 1. Для K > 2 класів зручно використовувати схему кодування 1-з-K, у якій t дорівнює вектору довжини K такому, що якщо клас C_j, то всі елементи t_k з t дорівнюють нулю, крім елемента t_j, який приймає значення 1. Наприклад, якщо у нас K = 5 класів, тоді зразок класу 2 отримає цільовий вектор

$$t = (0,1,0,0,0)^T. \quad (2.7)$$

Знову ж таки, ми можемо інтерпретувати значення t_k як ймовірність того, що клас є C_k. За невірогідної моделі, альтернативні варіанти представлення цільової змінної іноді виявляється зручним.

Найпростіший передбачає побудову дискримінантної функції, яка безпосередньо призначає кожен вектор x до певного класу. Однак більш потужний підхід моделює умовний розподіл ймовірностей $p(C_k \mid x)$ на стадії висновку, а потім використовує цей розподіл для прийняття оптимальних рішень. Розділяючи умови та прийняття рішення, ми отримуємо численні переваги. Існує два різних підходи до визначення умовних ймовірностей $p(C_k \mid x)$. Один спосіб полягає в тому, щоб моделювати їх безпосередньо, наприклад, представляючи їх як параметричні моделі, а потім оптимізуючи параметри за допомогою навчального набору. Іншим шляхом можна прийняти генеративний підхід, в якому ми моделюємо умовно-класову щільність задану $p(x \mid C_k)$, разом з попередніми ймовірностями $p(C_k)$ для класів, а потім ми обчислюємо необхідні наступні ймовірності, використовуючи теорему Байєса

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)}. \quad (2.8)$$

У моделях лінійної регресії, розглянутих раніше, передбачення моделі y(x, w) задана лінійною функцією параметрів w. У найпростішому випадку, модель є лінійною за вхідними змінними і тому має вигляд $y(x) = w^T x + w_0$, так що y - дійсне число. Однак для проблем класифікації ми бажано передбачити дискретні мітки класів або, загальніше, подальші ймовірності, що лежать у діапазон (0, 1). Для цього розглядається узагальнення цієї моделі, в якому перетворюється лінійну функцію w за допомогою нелінійної функції $f(\cdot)$ так, що

$$y(x) = f(w^T x + w_0). \quad (2.9)$$

У літературі стосовно машинного навчання, $f(\cdot)$ відома як функція активації, де її обернену в статистичній літературі називають функцією зв'язку. Поверхні рішень відповідають тому, що $y(x) = \text{const}$, так що $w^T_x + w_0 = \text{const}$, а отже, поверхні вирішення є лінійними функціями від x, навіть якщо функція $f(\cdot)$ нелінійна. Тому клас моделей, описаних у (формула вище), називають узагальненими лінійними моделями.

Проте, на відміну від використовуваних моделей для регресії, вони більше не є лінійними за параметрами через наявність нелінійної функції $f(\cdot)$. Це призведе до більш складних аналітичних та обчислювальних властивостей, ніж для моделей лінійної регресії. Тим не менше, ці моделі є все ще відносно прості у порівнянні з більш загальними нелінійними моделями.

Алгоритми, розглянуті надалі, будуть однаково застосовні, якщо спочатку здійснити фіксоване нелінійне перетворення вхідних змінних за допомогою вектору базису функції $\varphi(x)$, як для регресійних моделей.

Дискримінант - це функція, яка приймає вхідний вектор x і призначає його одному з K класів, позначається C_k. Далі розгляд обмежить увагу лінійними дискримінантами, а саме тими, для яких поверхнями прийняття рішення є гіперплощини. Щоб спростити обговорення, спочатку розглянеться випадок двох класів, а потім дослідимо розширення до $K > 2$ класів.

Найпростіше уявлення про лінійну дискримінантну функцію отримують, взявши лінійну функцію вхідного вектору так, що

$$y(x) = w^T x + \omega_0, \qquad (2.10)$$

де w називається ваговим вектором, а $\omega 0$ - зміщенням. Негатив базису іноді називають порогом. Вхідний вектор x призначається класу C_1, якщо $y(x) \geq 0$, а класу C_2 - інакше. Відповідна межа прийняття рішення визначається співвідношенням $y(x)=0$, що відповідає (D-1)-вимірній гіперплощині всередині D-вимірного простору входів. Розглянемо дві точки x_A і x_B, обидві з яких лежать на поверхні прийняття рішення. Оскільки $y(x_A) = y(x_B) = 0$, маємо $w^T(x_A - x_B) = 0$, а отже, вектор w є ортогональним

кожному вектору, що лежить на поверхні прийняття рішення, і тому w визначає орієнтацію поверхні прийняття рішення. Аналогічно, якщо x - точка на поверхні прийняття рішення, тоді y(x) = 0, і тому нормальна відстань від початку координат до поверхні прийняття рівна

$$\frac{w^T x}{||w||} = \frac{\omega_0}{||w||}. \tag{2.11}$$

Тому ми бачимо, що параметр зміщення ω_0 визначає позицію поверхні прийняття рішень. Ці властивості проілюстровані для випадку D = 2 на рисунку 2.2.

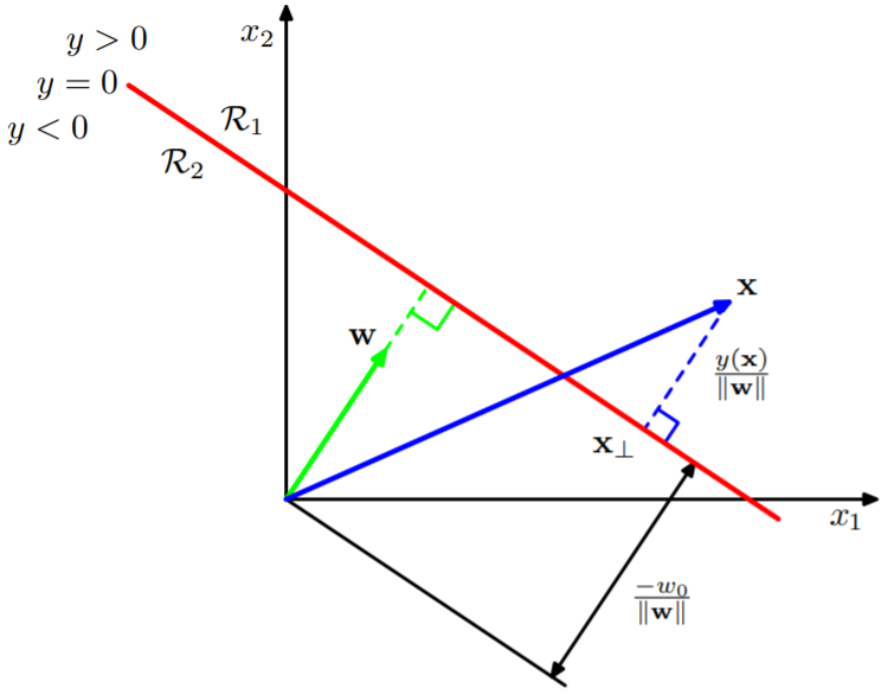


Рисунок 2.2. Ілюстрація геометрії лінійної дискримінантної функції у двох вимірах.

Поверхня прийняття рішення, показана червоним кольором, перпендикулярна до w, і її зміщення від початок контролюється параметром

зміщення ω_0. Також дається знакова ортогональна відстань загальної точки x від поверхні прийняття рішення по y(x)/||w||.

Крім того, зауважимо, що значення y(x) дає знакову міру перпендикулярної відстані r точки x від поверхні прийняття рішення. Щоб побачити це, розглянемо довільну точку x і нехай $x_\perp$ - її ортогональна проекція на поверхню рішення, так що

$$x = x_\perp + r\frac{w}{||w||}. \tag{2.12}$$

Перемноживши обидві частини рівняння на w^T , додавши ω_0 та використавши, що $y(x) = w^T x + \omega_0$ та $y(x_\perp) = w^T x_\perp + \omega_0$, ми отримаємо

$$r = \frac{y(x)}{||x||}. \tag{2.13}$$

Результат цього зображено на рисунку 2.2.

Тепер розглянемо розширення лінійних дискримінантів до K > 2 класів. Виникає спокуса побудувати дискримінант K-класу шляхом об'єднання ряду двокласних дискримінантних функцій. Однак це призводить до деяких серйозних труднощів, як буде продемонстровано. Розглянемо використання класифікаторів K − 1, кожен з яких вирішує двокласну задачу відокремлення точок у певному класі C_k від точок не в цьому класі. Це відомо як класифікатор "один проти решти". Лівий приклад на рисунку 2.3 показує приклад, що включає три класи, де цей підхід веде до областей вхідного простору які неоднозначно класифікуються.

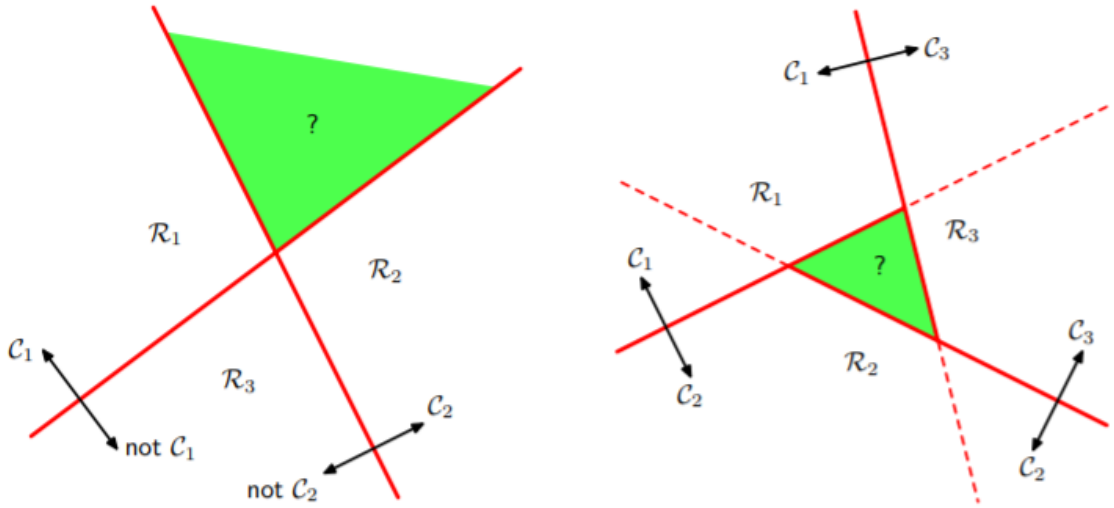


Рисунок 2.3. Зображення варіантів класифікатору «один проти решти»

Альтернативою є введення двійкових дискримінантних функцій K(K - 1)/2, по одній для кожної можливої пари класів. Це відоме як класифікатор "один проти одного". Тоді кожна точка класифікується за більшістю влучань серед дискримінаційних функцій. Однак це також стикається з проблемою неоднозначних регіонів, як показано на малюнку на правій діаграмі рисунка (номер рисунку вище).

Ми можемо уникнути цих труднощів, розглянувши єдиний дискримінант K-класу, що містить K лінійних функцій виду

$$y_k(x) = w_k^T x + \omega_{k0}, \qquad (2.14)$$

а потім присвоївши точку x класу C_k, якщо $y_k(x) > y_j(x)$ для всіх j = k. Межа рішень між класом C_k та класом C_j задається формулою $y_k(x) = y_j(x)$ та отже, відповідає (D-1)-вимірній гіперплощині, визначеній

$$(w_k - w_j)^T x + (\omega_{k0} - \omega_{j0}) = 0. \qquad (2.15)$$

Області прийняття рішень такого дискримінанта завжди поодиноко пов'язані між собою та опуклі. Щоб побачити це, розглянемо дві точки x_A і x_B,

обидві з яких знаходяться всередині області рішень R_k, як показано на рисунку 2.4.

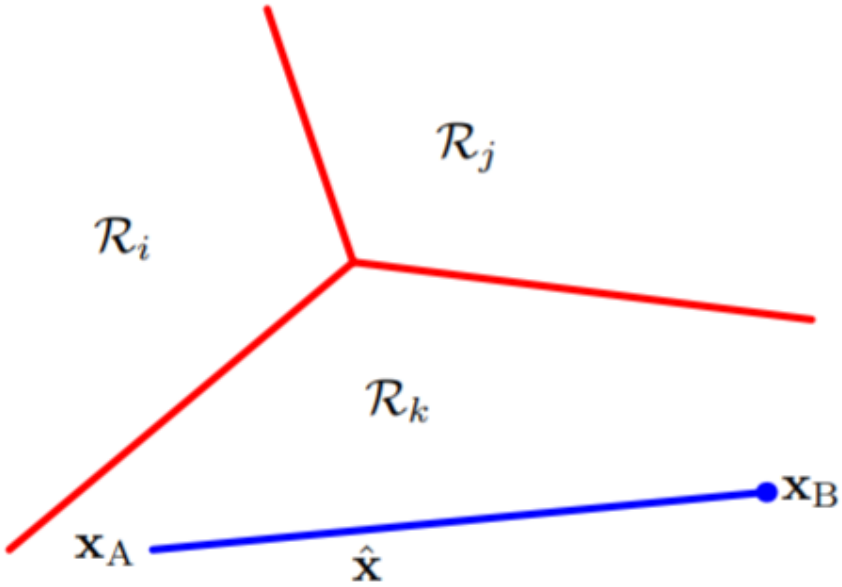


Рисунок 2.4. Зображення області рішень R_k та двох точок на ній

Ілюстрація областей прийняття рішення для багатокласового лінійного дискримінанта, де кордони рішення показані червоним кольором. Якщо дві точки x_A і x_B обидві лежать всередині однієї області прийняття рішень R_k, то будь-яка точка x_b, що лежить на прямій з'єднання цих двох точок також повинно лежати в R_k, а отже, область прийняття рішення має бути поодиноко сполучені та опуклі.

Будь-яку точку x, що лежить на прямій, що з'єднує x_A та x_B, можна виразити у вигляді

$$x = \lambda x_A + (1 - \lambda)x_B, \qquad (2.16)$$

де $0 \leq \lambda \leq 1$.

Оскільки і x_A, і x_B лежать всередині R_k, випливає, що $y_k(x_A) > y_j(x_A)$ і $y_k(x_B) > y_j(x_B)$, для всіх $j = k$, а отже, $y_k(x) > y_j(x)$, і тому x також лежить всередині R_k. Таким чином, R_k поодиноко пов'язаний і опуклий.

Зауважимо, що для двох класів можна використати формалізм на основі двох дискримінантних функцій $y_1(x)$ та $y_2(x)$, або ж використовувати простіше, але еквівалентне формулювання на основі єдиного дискримінанта функції $y(x)$.

Лапласівська апроксимація. Для подальшого розгляду та використання лінійної регресії, варто зазначити, що на даному етапі не можна точно інтегруватися над параметром вектора w, оскільки задній розподіл не є гаусовим. Тому необхідно ввести деяку форму наближення – а саме лапласівську апроксимацію.

Наближення Лапласа, яке має на меті знайти гаусове наближення до щільності ймовірності, визначається набором безперервних змінних. Розглянемо спочатку випадок однієї безперервної змінної z і припустимо, що розподіл p(z) визначається формулою

$$p(z) = \frac{1}{Z} f(z), \tag{2.17}$$

де $Z = \int f(z)\, dz$ - коефіцієнт нормалізації. Будемо вважати, що значення Z невідоме. У методі Лапласа мета полягає в тому, щоб знайти гаусівське наближення q(z), яке зосереджене на режимі розподілу p(z). Перший крок - це знайти режим p(z), іншими словами, таку точку z_0, що $p(z_0) = 0$, або еквівалентно

$$\left.\frac{df(z)}{dz}\right|_{z=z_0} = 0. \tag{2.18}$$

Гаусівський розподіл має властивість, що його логарифм є квадратичною функцією змінних. Тому ми розглянемо розклад Тенлора ln f(z) з центром на режим z_0 так що

$$\ln f(z) \approx \ln f(z_0) - \frac{1}{2} A(z - z_0)^2, \tag{2.19}$$

$$\text{де } A = -\frac{d^2}{dz^2} lnf(z)\Big|_{z=z_0} \tag{2.20}$$

Зауважимо, що доданок першого порядку у розкладі Тейлора не з'являється, оскільки z_0 є локальний максимум розподілу. Беручи експоненту, отримуємо

$$f(z) \approx f(z_0) exp\left\{-\frac{A}{2}(z - z_0)^2\right\}. \tag{2.21}$$

Потім можна отримати нормований розподіл q(z) шляхом використання стандартного результату нормалізації Гаусівського, так що

$$q(z) \approx \left(\frac{A}{2\pi}\right)^{1/2} exp\left\{-\frac{A}{2}(z - z_0)^2\right\}. \tag{2.22}$$

Наближення Лапласа показано на малюнку (номер малюнка нижче). Зауважимо, що Гаусівське наближення буде чітко визначено, лише якщо його точність A > 0, інакше кажучи стаціонарна точка z_0 повинна бути локальним максимумом, так що друга похідна від f(z) в точці z_0 від'ємна.

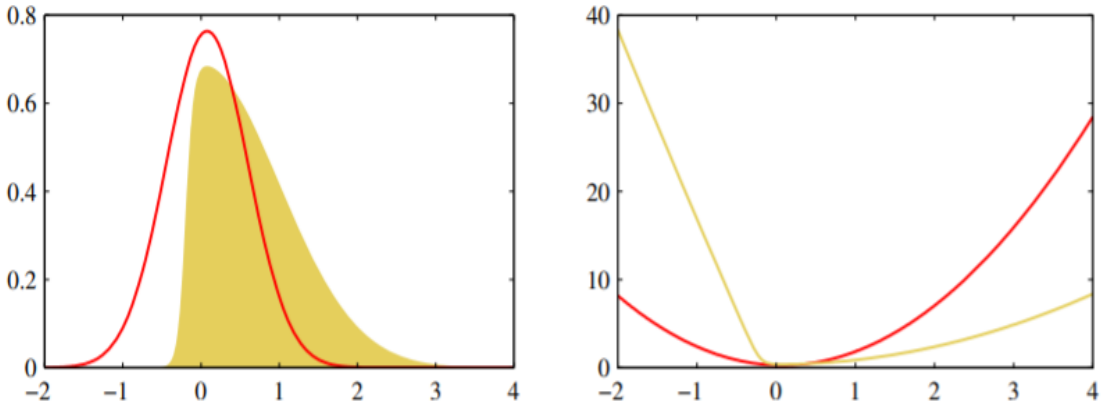


Рисунок 2.5. Наближення Лапласа застосоване до розподілу $p(z)$

Рисунок 2.5 зображує наближення Лапласа застосоване до розподілу $p(z) \propto \exp(-z^2/2)\sigma(20z + 4)$, де σ(z) - логістична сигмоїдна функція, визначена $\sigma(z) = (1 + e^{-z})^{-1}$. Лівий графік показує нормований розподіл p(z) жовтим

разом із наближенням Лапласа з центром на моді z_0 p(z) червоним. Правий графік показує від'ємні логарифми відповідних кривих.

Метод Лапласа можна розширити на апроксимацію розподілу p(z) = f (z)/Z визначеним над M-вимірним простором z. У нерухомій точці z_0 градієнт ∇f(z) зникне. Розширюючись навколо цієї нерухомої точки, маємо

$$\ln f(z) \approx \ln f(z_0) - \frac{1}{2}(z - z_0)^T A(z - z_0), \tag{2.23}$$

де M×M-матриця Гессе A визначається формулою

$$A = -\nabla\nabla \ln f(z)|_{z=z_0} \tag{2.24}$$

і ∇ є оператором градієнту. Взявши експоненцію обох сторін, ми отримаємо

$$f(z) \approx f(z_0) \exp\left\{-\frac{1}{2}(z - z_0)^T A(z - z_0)\right\}. \tag{2.25}$$

Розподіл q(z) пропорційний f(z), і відповідний коефіцієнт нормалізації можна знайти шляхом перевірки, використовуючи стандартний результат для нормованого багатовимірного гауссівського розподілу, отримуючи

$$q(z) = \frac{|A|^{1/2}}{(2\pi)^{M/2}} exp\left\{-\frac{1}{2}(z - z_0)^T A(z - z_0)\right\} = \boldsymbol{\mathcal{N}}(z|z_0, A^{-1}) \tag{2.26}$$

де |A| позначає визначник A. Цей гаусівський розподіл буде правильно визначеним за умови, що його матриця точності, задана A, є позитивно визначеною, що означає що точка нерухомості z_0 повинна бути локальним максимумом, а не мінімумом або сідловою точкою.

Щоб застосувати наближення Лапласа, нам спочатку потрібно знайти режим z_0, а потім обрахувати матрицю Гессе в цьому режимі. На практиці режим зазвичай знаходиться за допомогою певної форми чисельного алгоритму оптимізації.

Багато розподілів, які зустрічаються на практиці, будуть мультимодальними, тому існуватимуть різні наближення Лапласа, згідно з якими розглядається режим. Зауважимо, що для застосування методу Лапласа

не потрібно знати нормалізацію Z істинного розподілу. Як результат центральної граничної теореми, очікується, що подальший розподіл моделі стає дедалі краще будучи наближеним гаусівським наближенням, відповідно до того, як кількість спостережуваних точок даних збільшується, і тому очікується, що наближення Лапласа найбільш корисне у ситуаціях, коли кількість точок даних відносно велика.

Одним з основних недоліків наближення Лапласа є те, що оскільки він базується на Гаусівському розподілі, він безпосередньо застосовується лише до дійсних змінних. В інших випадках можливо, можна застосувати наближення Лапласа для перетворення змінної. Наприклад, якщо $0 \leq \tau < \infty$, то можна розглядати наближення Лапласа з $\ln \tau$. Найсерйознішим обмеженням наближення Лапласа є те, що воно базується виключно на аспектах справжнього розподілу за певним значенням змінної, і тому не може захопити важливі глобальні властивості [7].

Кластеризація

Навчання без нагляду спрямоване на виявлення основних властивостей та шаблонів з немічених навчальних зразків та укладання основи для подальшого аналізу даних. Серед різноманітних методів навчання без нагляду найбільш досліджена та прикладна є кластеризація.

Кластеризація має на меті розділити набір даних на непересічні підмножини, де кожна підмножина називається кластером. Завдяки розподілу кожен кластер потенційно відповідає певній концепції (категорії), наприклад, «світло-зелений кавун», «темно-зелений кавун», «кавун із кісточками», «кавун без кісточок» або навіть «кавун місцевого виробництва» та «імпортний кавун». Варто зазначити, що алгоритми кластеризації не знають про такі концепції до кластеризації і відповідають лише за створення кластерів. Однак концепція кожного кластера інтерпретується користувачем.

Формально, для даного набору даних $D = \{x_1, x_2, ..., x_m\}$, що містить зразки без міток, кожен зразок $x_i = (x_{i1}; x_{i2}; ...; x_{in})$ є n-вимірним вектором. Потім алгоритм кластеризації розділяє набір даних D на k непересічних кластерів $\{C_l \mid l = 1, 2, ..., k\}$, де $C_{l'} \cap_{l' \neq l} C_l = \emptyset$ та $D = \cup_{l=1}^{k} C_l$. Відповідно, ми позначимо $\lambda_j \in \{1, 2, ..., k\}$ як мітку кластера зразка x_j (тобто $x_j \in C_{\lambda j}$). Тоді результат кластеризації можна представити у вигляді вектора мітки кластера $\lambda = (\lambda_1; \lambda_2; ...; \lambda_m)$ з m елементами.

Кластеризація може використовуватися сама по собі для ідентифікації невід'ємної структури даних, а також може служити інструментом попередньої обробки для інших навчальних завдань, таких як класифікація. Наприклад, компанія може хотіти класифікувати нових користувачів у різні «категорії», але це може бути непросто. У такому випадку ми можемо використовувати кластеризацію, щоб згрупувати всіх користувачів у кластери, де кожен кластер представляє категорію користувача. Потім на кластерах можна побудувати модель класифікації для класифікації категорії нових користувачів.

Залежно від використовуваної стратегії навчання алгоритми кластеризації можна розділити на кілька категорій. Перш за все, розглянемо дві фундаментальні проблеми, пов'язані з кластеризацією - міру продуктивності та розрахунок відстані.

Показники ефективності для кластеризації також називаються *індексами достовірності*. Оскільки результат класифікації оцінюється за допомогою показників ефективності в навчанні під наглядом, результат кластеризації також необхідно оцінювати за допомогою деяких індексів обґрунтованості. Крім того, як тільки індекс валідності буде обрано, ми зможемо вбудувати його в мету оптимізації алгоритмів кластеризації, щоб сформовані кластери були більш узгодженими з бажаними результатами.

Отже, що можна назвати хорошою кластеризацією? Інтуїтивно, ми хочемо, щоб «щось своєрідне зійшлося»; тобто вибірки в одному кластері повинні бути максимально схожими, тоді як вибірки з різних кластерів повинні бути максимально різними. Іншими словами, ми шукаємо кластери з високою *внутрішньокластерною* подібністю та низькою *міжкластерною* подібністю.

Грубо кажучи, існує два типи індексів достовірності кластеризації. Перший тип - це *зовнішній індекс*, який порівнює результат кластеризації з *еталонною моделлю*. Другий тип - це *внутрішній індекс*, який оцінює результат кластеризації без використання будь-якої еталонної моделі.

Враховуючи набір даних $D = \{x_1, x_2, ..., x_m\}$, припустимо, що алгоритм кластеризації створює кластери $C = \{C_1, C_2, ..., C_k\}$ і опорна модель дає кластери $C^* = \{C_1^*, C_2^*, \dots, C_s^*\}$. Відповідно, нехай λ і λ^* позначають вектори міток кластеризації. C та C^* відповідно. Тоді для кожної пари зразків ми можемо визначте наступні чотири умови:

$$a = |SS|, \qquad SS = \{(x_i, x_j) | \lambda_i = \lambda_j, \lambda_i^* = \lambda_j^*, i < j\}, \tag{2.27}$$

$$b = |SD|, \qquad SD = \{(x_i, x_j) | \lambda_i = \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j\}, \tag{2.28}$$

$$c = |DS|, \qquad DS = \{(x_i, x_j) | \lambda_i \neq \lambda_j, \lambda_i^* = \lambda_j^*, i < j\}, \tag{2.29}$$

$$d = |DD|, \qquad DD = \{(x_i, x_j) | \lambda_i \neq \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j\}, \tag{2.30}$$

де набір *SS* включає пари вибірок таких, що обидві вибірки належать до одного кластера в *C*, а також належать до того самого кластер у *C**; набір *SD* містить пари вибірок таких, що обидві вибірки належать до того самого кластера в *C*, але не в *C**; множини *DS* і *DD* можна інтерпретувати аналогічно. Оскільки кожна пара зразків (x_i, x_j) $(i < j)$ може з'являтися лише в одному наборі, маємо $a + b + c + d = m(m-1)/2$.

На основі формул (2.27) - (2.30) деякі поширені зовнішні індекси можна визначити наступним чином

- Коефіцієнт Джакарда

$$\mathrm{JC} = \frac{a}{a+b+c} \tag{2.31}$$

- Індекс Фоулкса та Маллоуса

$$\mathrm{FMI} = \sqrt{\frac{a}{a+b} * \frac{a}{a+c}} \tag{2.32}$$

- Індекс Ранда

$$\mathrm{RI} = \frac{2(a+d)}{m(m-1)} \tag{2.33}$$

Наведені вище індекси зовнішньої достовірності приймають значення в інтервалі [0, 1], і більше значення індексу свідчить про кращу якість кластеризації.

Внутрішні індекси валідності оцінюють якість кластеризації без використання еталонної моделі. Враховуючи сформовані кластери $C = \{C_1, C_2, \ldots, C_k\}$, ми можемо визначити наступні чотири умови:

$$\mathrm{avg}(C) = \frac{2}{|C|(|C|-1)} \sum_{1 \le i < j \le |C|} \mathrm{dist}(x_i, x_j) \tag{2.34}$$

$$\mathrm{diam}(C) = \max_{1 \le i < j \le |C|} \mathrm{dist}(x_i, x_j) \tag{2.35}$$

$$d_{min}(C_i, C_j) = \min_{x_i \in C_i, x_j \in C_j} \mathrm{dist}(x_i, x_j) \tag{2.36}$$

$$d_{cen}(C_i, C_k) = \mathrm{dist}(\mu_i, \mu_j) \tag{2.37}$$

де $\mu = \frac{1}{|C|} \sum_{1 \le i \le |C|} x_i$ позначає центроїд кластера C, і dist(·, ·) вимірює відстань між двома зразками. Тут, avg(C) - середня відстань між зразками в кластері C; diam(C) - найбільша відстань між зразками в кластері C; $\mathrm{d_{min}}$ (C_i, C_j) - це відстань між двома найближчими зразками в кластерах C_i і C_j; а $\mathrm{d_{cen}}$(C_i, C_j) - це відстань між центроїдами кластерів C_i і C_j.

На основі формул (2.34) - (2.37) деякі загальновживані внутрішні індекси достовірності можна визначити таким чином:

- Індекс Девіса-Болдвіна

$$\mathrm{DBI} = \frac{1}{k}\sum_{i=1}^{k}\max_{j\neq i}\left(\frac{\mathrm{avg}(C_i) + \mathrm{avg}(C_j)}{d_{cen}(C_i, C_j)}\right) \tag{2.38}$$

- Індекс Данна

$$\mathrm{DI} = \min_{1\leq i\leq k}\left\{\min_{j\neq i}\left(\frac{d_{min}(C_i, C_j)}{\max\limits_{1\leq i\leq k}\mathrm{diam}(C_i)}\right)\right\} \tag{2.39}$$

Менше значення DBI вказує на кращу якість кластеризації, тоді як більше значення DI вказує на кращу якість кластеризації [7].

2.1.2. Головна ідея дослідження

Згідно з дослідженням, близько 75% всіх працівників у розвинених країнах у свої професійній діяльності працюють у сидячому положенні [8], що призводить до довгочасних порушень у сидячій постаті людини, що в свою чергу призводить до кістково-м’язових ускладнень у спині, шиї, плечах, руках та ногах людини [9, 10]. Хронічні болі у спині вже стають повсякденною проблемою багатьох людей, стаючи майже професійним захворюванням [11].

З огляду на це, головною ідеєю даної магістерської роботи було впровадження оптимальної системи машинного навчання для визначення осанки людини. Як основний апарат для аналізу зображення людської постаті було обрано гібридний підхід з використанням згорткової нейронної мережі та архітектури довгої короткострокової пам’яті. Прикладною реалізацією ідеї є реалізація у вигляді розширення для веб-браузеру Google Chrome.

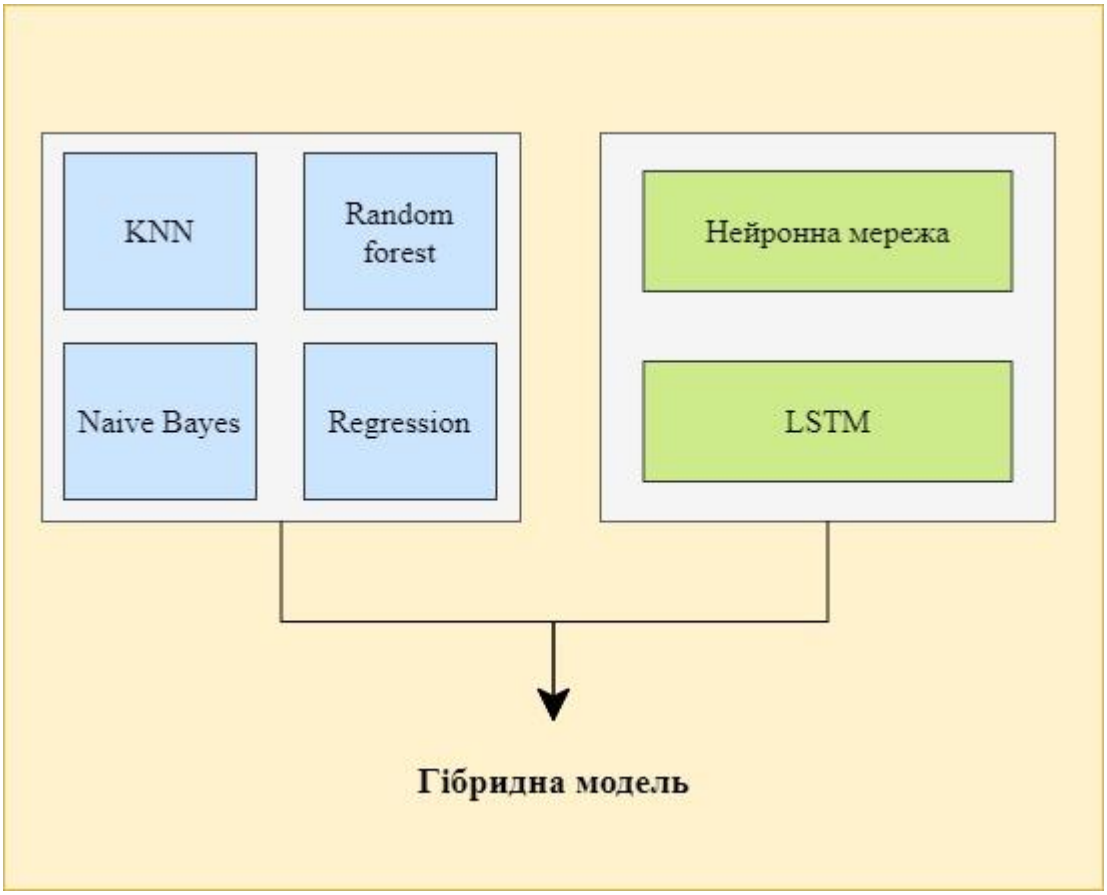


Рисунок 2.6. Загальна схема досліджуваної гібридної системи

На рисунку 2.6 зображено схему досліджуваної гібридної системи, яка складається з нейронної мережі та архітектури LSTM, а також додаткових методів машинного навчання у поєднанні з основним апаратом для покращення результатів, а саме random forest, KNN, Naive Bayes, decision tree.

2.1.3. Огляд основного апарату дослідження

Основний апарат, який буде складати базу проведення дослідження, це гібрид згорткової нейронної мережі та архітектури короткочасної довгострокової пам’яті. Розглянемо обидва компоненти детальніше.

Згорткова нейронна мережа – клас глибинних нейронних мереж, які переважно використовуються для аналізу візуальних зображень. Загальний

принцип роботи згорткових нейронних мереж у розпізнаванні зображень – це зображення протиставлення матриці глибини насичення кольору. Для чорно-білих зображень це буде матриця відтінків сірого, де 0 – це білий колір, а 1 – чорний. Кольорові зображення розкладаються на матрицю базових кольорів RGB – червоний, зелений та синій – де кожний елемент кожної матриці буде відповідати насиченню певного кольору відповідного пікселю.

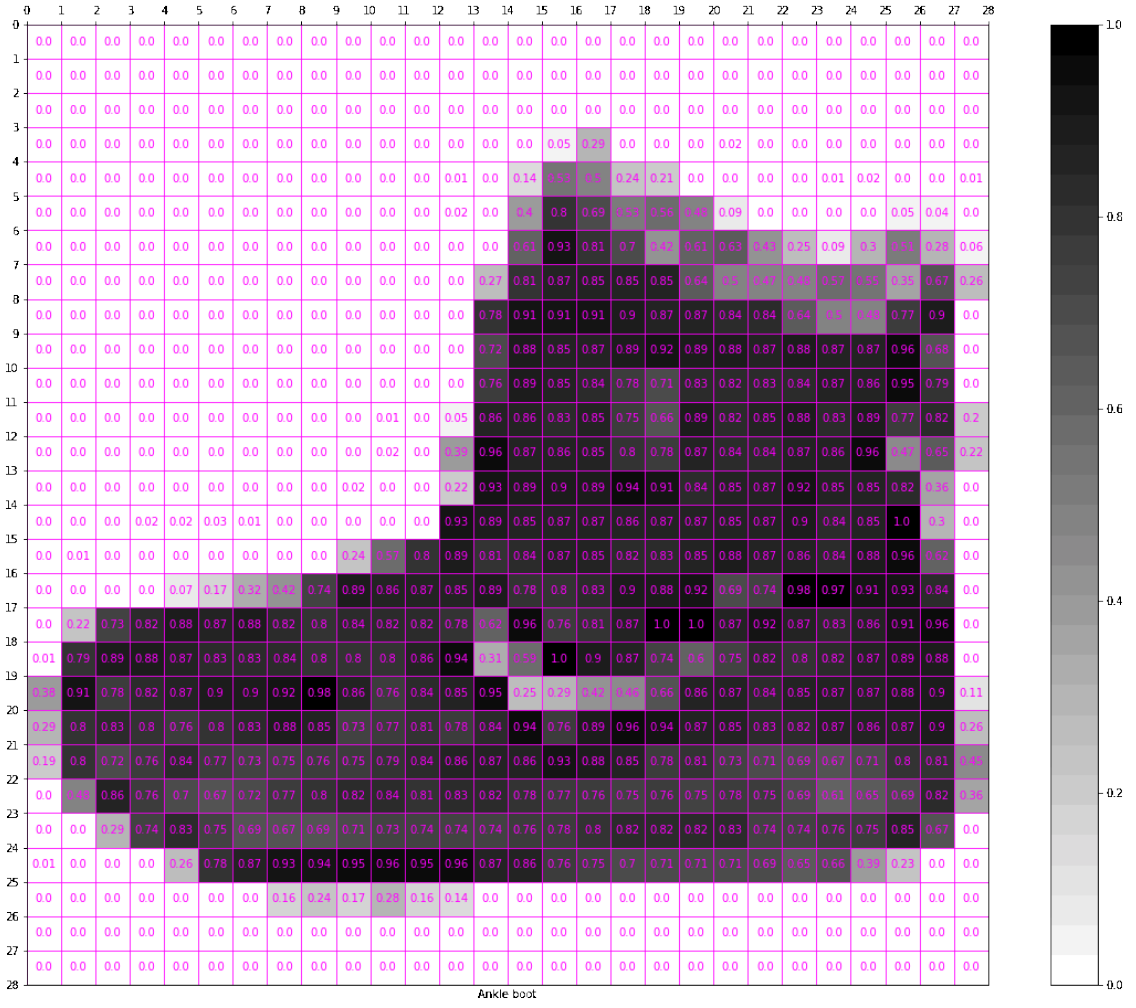


Рисунок 2.7. Зображення матриці відтінків сірого для простого зображення черевика

На рисунку 2.7 зображена матриця для просто зображення черевика розміром 28*28 пікселів.

Після представлення зображення у вигляді матриці (або матриць) відтінків, наступним важливим етапом є накладання згорткових фільтрів. Згорткові фільтри є набором ваг, які застосовуються до матриці відтінків. Ваги, які використовуються у згорткових фільтрах, визначаються та оновлюються у зворотному розповсюдженні під час навчання нейронної мережі. Процес накладання фільтрів на зображення генерує «відфільтровані» версії зображення, які називають мапами особливостей. Фільтри можуть як загострювати, так і згладжувати певні особливості зображень (кольори, відтінки, нерівномірності тощо). Але саме використання різних фільтрів з різними наборами параметрів забезпечує точність передбачень нейронної мережі.

Для прикладу розглянемо фільтр, який являє собою матрицю 3*3 зі значеннями $\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\0 & 0 & 0\end{bmatrix}$ та просте чорно-біле зображення букви «М», яке вже представлене у вигляді простої матриці 5*5, де кожний елемент набуває лише значень 0 або 1 (відсутність або наявність кольору), тобто $\begin{bmatrix}1 & 0 & 0 & 0 & 1\\1 & 1 & 0 & 1 & 1\\1 & 0 & 1 & 0 & 1\\1 & 0 & 0 & 0 & 1\\1 & 0 & 0 & 0 & 1\end{bmatrix}$.

У результаті накладання фільтру отримуємо мапу особливостей, яка буде мати такі значення $\begin{bmatrix}3 & 2 & 3\\4 & 3 & 4\\3 & 1 & 3\end{bmatrix}$. Варто зазначити, що наведений приклад є досить узагальненим, тому що на практиці значення, що відповідають пікселям, рідко приймають лише значень 0 або 1, а фільтри можуть мати одночасно негативні та позитивні значення.

Довга короткочасна пам'ять – це архітектура нейронних мереж, яка зазвичай використовується у рекурентних нейронних мережах. Мережа ДКЧП є особливим типом рекурентних нейронних мереж, яка має здатність до

запам’ятовування довготривалих зв’язків. На відміну від звичайних рекурентних нейронних мереж, які мають один (і той самий) шар у повторюваному блоці, ДКЧП мають чотири шари, які співпрацюють один з одним. Блоки (або ж вузли) ДКЧП використовують замість або у додачу до інших вузлів мережі.

Базовим набором шарів у ДКЧП є такі: 1-ий та 2-ий шар – сигмоїдна функція активації, 3-ій шар – гіперболічна тангенціальна функція активації, 4-ий шар - сигмоїдна функція активації. Основна ідея ДКЧП полягає у відслідковуванні стану шару у часі та контролю шарів, відповідно до їх результатів. Для цього у ДКЧП також наявні так звані «брами», які пропускають або не пропускають результати роботи шару. Різні комбінації типів та кількості шарів, а також брам формують різні варіації ДКЧП з унікальними властивостями для виконання певних задач.

2.1.4. Додаткові методи машинного навчання для проведення дослідження

У додачу до основних елементів апарату дослідження, також розглянемо використані додаткові методи машинного навчання, а саме random forest, KNN, Naive Bayes.

«Випадковий ліс» (англ. random forest, іноді random decision forest) – метод ансамблемового навчання для класифікації, регресії та інших завдань, що оперують деревами рішень під час тренування. Під час завдань класифікації, виходом «випадкового лісу» є клас, який був обраний для більшості дерев. Під час виконання задач регресії, виходом є середнє значення передбачення для кожного дерева. Зазвичай, «випадкові ліси» використовуються для корекції схильності дерев рішень до надмірної комплектації. Також вони як правило дають кращі та точніші результати, ніж звичайні дерева рішень.

Метод k-найближчих сусідів (англ. k-nearest neighbours, KNN) – метод машинного навчання з вчителем, який використовується у задачах класифікації та регресії. Цей метод припускає, що подібні дані розташовуються у значній близькості. Метод приймає за вхідні дані певний числовий набір та розраховує відстані між елементами, сприймаючи їх у вигляді графу. Потім на основі отриманих відстаней, метод формує припущення про належність певної величини чи значення до тієї чи іншої групи.

Наївний баєсівський класифікатор (англ. naive Bayes) – один з простих ймовірнісних класифікаторів, який базується на основі теореми Байєса. Більш детальний опис у розділі 1.3.3, ст. 27, формула 1.9.

Дерево рішень (англ. decision tree) – передбачувальна модель, що відображає знання про певний об’єкт у множину рішень. Дерева рішень використовуються (як моделі прогнозування) для переходу від спостережень за предметом (представленим у гілках) до висновків про цільове значення елемента (представлене у листках). Моделі дерев, де цільова змінна може приймати дискретний набір значень, називаються деревами класифікації; у цих деревних структурах листя представляють мітки класів, а гілки – поєднання ознак, які ведуть до цих міток класів. Дерева рішень, де цільова змінна може приймати безперервні значення (зазвичай дійсні числа), називаються деревами регресії. Дерева рішень є одними з найпопулярніших алгоритмів машинного навчання з огляду на їх зрозумілість та простоту [7].

2.1.5. Базові статистичні та математичні параметри для налаштування основного апарату дослідження

Далі розглянемо основні математичні та статистичні величини, якими звично оперують для аналізу зображень.

Коефіцієнт асиметрії – числова характеристика розподілу ймовірностей дійної випадкової величини. Асиметрією теоретичного розподілу ймовірностей

випадкової величини називають відношення центрального моменту третього порядку μ_3 до куба середнього квадратичного відхилення σ^3 :

$$\gamma_1 = \mu_3/\sigma^3. \quad (2.40)$$

Квантиль – числова характеристика випадкових величин. Квантилі відсікають в межах ряду певну частину його членів. Тобто, квантиль розподілу значень – це таке число x_p, що значення p-ї частини сукупності менше або рівне x_p. Наприклад, квантиль 0.50 змінної – таке значення x_p, що 50% значень змінної потрапляють нижче даного значення.

Стандартне відхилення – один із найчастіше вживаних показників розсіювання випадкової величини відносної її математичного сподівання. Позначається символом σ та визначається як квадратний корінь із дисперсії.

Коефіцієнт ексцесу – числова характеристика розподілу ймовірностей випадкової величини, яка характеризує стрімкість підвищення кривої розподілу у порівнянні з нормальною кривою.

Також із базових понять у розрахунках застосовується середнє значення та квадратний корінь.

Для подальшого спрощення подачі результатів проведеного дослідження, введемо відповідні позначення для розглянутих понять, а саме: коефіцієнт асиметрії – «skew», квантиль – «percentile», стандартне відхилення – «SD», коефіцієнт ексцесу – «kurtosis», середнє значення – «mean», квадратний корінь – «SR».

2.1.6. Налаштування та проведення дослідження

Для проведення дослідження методів та алгоритмів для визначення постаті були використані вище описані методи та параметри у різних комбінаціях, проаналізовано їх результати та точність, проведено аналіз результатів.

Загальна схема алгоритму системи, над якою проводилося дослідження, зображена на рисунку 2.8.

Рисунок 2.8. Загальна схема досліджуваної системи

Використані налаштування додаткових методів машинного навчання подано нижче у таблиці 2.1.

Таблиця 2.1

Використані у дослідженні методи машинного навчання, їх параметри

Метод	**Використаний параметр**
KNN	Евклідова відстань, 3 сусіди
Random forest	Максимальна глибина = 3
Naïve Bayes	Без використання базової «ваги»
LSTM	10-шарова

Для оцінки методів, комбінацій та налаштувань були використані такі метрики:

$$\text{Влучність} = \frac{\text{ІП}}{\text{ІП} + \text{ХП}} \tag{2.41}$$

$$\text{Повнота} = \frac{\text{ІП}}{\text{ІП} + \text{ХН}} \tag{2.42}$$

$$F - \text{міра} = 2 * \frac{\text{Влучність} * \text{Повнота}}{\text{Влучність} + \text{Повнота}} \tag{2.43}$$

$$\text{Точність} = \frac{\text{ІП} + \text{ІН}}{\text{ІП} + \text{ІН} + \text{ХП} + \text{ХН}}, \tag{2.44}$$

де ІП, ІН, ХП, ХН – стани результатів, а саме ІП – істинно позитивний (рівнозначно точному влучанню), ІН – істинно негативний (рівнозначно точному відхиленню результату), ХП – хибно позитивний (рівнозначно хибній тривозі, переоцінці), ХН – хибно негативний (рівнозначно промаху результату, недооцінці).

2.1.7. Результати роботи досліджуваної системи

У наступних таблицях наведені результати різних наборів базових математичних параметрів у налаштуванні та використанні окремих компонентів досліджуваної системи метриками, які розглянуто вище у цьому пункті.

Таблиця 2.2

Оцінка результатів використання Random Forest для визначення осанки

Параметр	*Точність*	*Влучність*	*F-міра*	*Повнота*
Mean	87,86	0,85	0,92	0,86
Mean, SD	86,12	0,83	0,89	0,88
Mean, SD, SR	90,21	0,90	0,89	0,90
Mean, SD, SR, percentile	90,11	0,92	0,88	0,89
Mean, SD, SR, percentile, kurtosis	90,32	0,90	0,90	0,89
Mean, SD, SR, percentile, kurtosis, skew	90,59	0,90	0,92	0,91

Таблиця 2.3

Оцінка результатів використання KNN для визначення осанки

Параметр	*Точність*	*Влучність*	*F-міра*	*Повнота*
Mean	81,94	0,78	0,86	0,81
Mean, SD	82,49	0,78	0,88	0,81
Mean, SD, SR	82,03	0,78	0,86	0,81
Mean, SD, SR, percentile	82,69	0,779	0,86	0,82
Mean, SD, SR, percentile, kurtosis	82,59	0,80	0,77	0,81
Mean, SD, SR, percentile, kurtosis, skew	82,7	0,81	0,80	0,81

Таблиця 2.4

Оцінка результатів використання Naïve Bayes для визначення осанки

Параметр	*Точність*	*Влучність*	*F-міра*	*Повнота*
Mean	91,00	0,90	0,90	0,91
Mean, SD	92,92	0,91	0,93	0,93
Mean, SD, SR	92,35	0,89	0,93	0,91
Mean, SD, SR, percentile	88,65	0,87	0,86	0,88
Mean, SD, SR, percentile, kurtosis	90,67	0,90	0,91	0,90
Mean, SD, SR, percentile, kurtosis, skew	90,82	0,90	0,91	0,91

Таблиця 2.5

Оцінка результатів використання LSTM для визначення осанки

Параметр	*Точність*	*Влучність*	*F-міра*	*Повнота*
Mean	88,49	0,88	0,86	0,87
Mean, SD	89,07	0,87	0,87	0,87
Mean, SD, SR	90,4	0,91	0,90	0,91
Mean, SD, SR, percentile	90,4	0,91	0,90	0,90
Mean, SD, SR, percentile, kurtosis	91,19	0,91	0,90	0,91
Mean, SD, SR, percentile, kurtosis, skew	91,77	0,91	0,90	0,91

Аналізуючи отримані результати, можна дійти висновку, що окремо кожен з обраних методів машинного навчання дає результати, які досить грубо можуть бути заокруглені до ≈ 90%.

Однак, при створенні ідеї даної магістерської роботи, очікувана точність складала щонайменше 93%. Тому продовжимо оцінювання для нейронної мережі у поєднанні з архітектурою LSTM, що було запропоновано у ідеї дослідження.

Таблиця 2.6

Оцінка результатів використання гібриду CNN+LSTM для визначення осанки

Параметр	*Точність*	*Влучність*	*F-міра*	*Повнота*
Mean	91,06	0,91	0,91	0,91
Mean, SD	91,31	0,92	0,92	0,92
Mean, SD, SR	91,48	0,92	0,92	0,92
Mean, SD, SR, percentile	91,82	0,93	0,93	0,92
Mean, SD, SR, percentile, kurtosis	92,36	0,94	0,93	0,94
Mean, SD, SR, percentile, kurtosis, skew	92,46	0,95	0,94	0,95

Отже, як видно з результатів, дослідження виявилось досить вдалим, оскільки при вірному налаштуванні, достатньому навчанні та використанні всіх параметрів з гібридною системою, можливо досягнути точності ±92%.

3. ПРАКТИЧНА ЧАСТИНА

3.1. Огляд прикладних інструментів для проведення дослідження

3.1.1. Огляд TensorFlow

TensorFlow широко використовується як бібліотека впровадження машинного навчання. Її створив Google як частину проекту Google Brain, а пізніше вона стала доступна як продукт з відкритим кодом, оскільки існувало декілька бібліотек машинного та глибокого навчання, які привертали увагу користувачів. З доступом до відкритого коду все більше і більше людей у сфері штучного інтелекту (AI) та машинного навчання змогли перейняти TensorFlow і почати створювати функції та продукти з його використанням. Це не тільки допомагало користувачам із впровадженням стандартного машинного навчання та алгоритмів глибокого навчання, але також дозволило їм реалізувати індивідуальні та диференційовані версії алгоритмів для бізнес-додатків та різні дослідження. Насправді вона незабаром стала однією із найпопулярніших бібліотек у спільнотах машинного навчання та штучного інтелекту – настільки люди створювали величезну кількість програм за допомогою TensorFlow під капотом. В основному це пояснюється тим, що сама Google використовує TensorFlow у більшості своїх продуктів, будь то Карти Google, Gmail, або інші програми.

Хоча TensorFlow має свої сильні сторони в певних областях, вона також мала деякі обмеження, через які розробникам було важко починати роботу з нею, порівняно з іншими бібліотеками, такими як PyTorch, Theano та OpenCV. Коли команда TensorFlow серйозно віндеслась до відгуків спільноти про TensorFlow, це призвело до більшісті змін, необхідних для того, щоб зробити TensorFlow ще більш ефективною і простою у роботі – як результат з’явилась TensorFlow 2.0. TensorFlow 2.0 проголошує, що позбулася деяких попередніх

перешкод, щоб дозволити розробникам більше плавно та легко використовувати TensorFlow [13].

3.1.2. TensorFlow – поняття «тензору»

Базовим поняттям при роботі з TensorFlow є, власне, тензор. Тензори є основним матеріалом TensorFlow, всі обчислення відбуваються за допомогою тензорів. Отже, що ж таке тензор? Відповідно до визначення, наданого командою Google TensorFlow, «Тензор - це узагальнення векторів і матриць на потенційно вищі розміри. Внутрішньо TensorFlow представляє тензори як n-мірні масиви базових типів даних».

Але варто трохи заглибитися у тензори, щоб надати більше, ніж загальний огляд їх сутності. Було б доцільно порівняти їх з векторами або матрицями, щоб виділити ключову динамічну властивість, яка робить тензори такими потужними.

Почнемо з простого вектора. Під вектором зазвичай розуміють те, що має величину і напрямок. Простіше кажучи, це масив, який містить упорядкований список значень. Без напрямку вектора тензор стає скалярним значенням, яке має тільки величину. Вектор може бути використаний для представлення n-кількості речей. Він може відображати область та різні атрибути.

Припустимо, що у нас є вектор Â, як показано на рисунку 3.1. Це наразі представлення без урахування системи координат, але всім відома декартова система координат (вісь x, y, z).

Рисунок 3.1. Просте зображення вектору Â

Вектори також можуть зображатися за допомогою базисних векторів. Базисні вектори пов'язані з системою координат і їх можна використовувати для представлення будь-якого вектору. Ці базисні вектори мають довжину 1 і, отже, також відомі як одиничні вектори. Визначено напрямок цих базисних векторів за їх відповідними координатами. Наприклад, для тривимірного представлення, ми маємо три базисних вектора (xˆ, yˆ, zˆ), тому xˆ матиме значення напрямку координати осі x, а базис yˆ мав би напрямок осі y. Так само це має місце і для zˆ.

Щоб знайти першу складову вектора Â вздовж осі x, його проектують на вісь x, як показано на рисунку 3.2. Тепер, у точці, де проекція перетинає вісь x, отримано x-компоненту вектора.

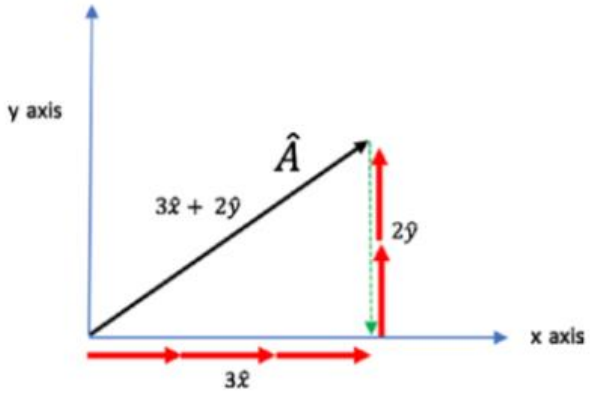


Рисунок 3.2. Проекція вектору на осі координат

Також легко розпізнати цю компоненту x як суму кількох базисних векторів уздовж осі x. У цьому випадку три базисних вектори дадуть компоненту x вектора Â. Так само ми можемо знайти у компонента вектора Â, проецюючи його на вісь y та додаючи базисні вектори (2y) вздовж осі y для її представлення. Звідси отримаємо значення вектора Â як $\hat{A} = 3\hat{x} + 2\hat{y}$.

Якщо в підсумку помножити вектор на інший вектор, отримаємо у результаті скалярну величину, тоді як якщо помножити вектор на скаляр, це просто збільшує або зменшує вектор в тій же пропорції, з точки зору його величини, не змінюючи напрямку. Однак, якщо помножити вектор на тензором, це призведе до того, що новий вектор буде мати нову величину, а також новий напрямок.

Зрештою, тензор - це також математична сутність, за допомогою якої представляють різні властивості, схожі на скаляр, вектор або матрицю. Це правда що тензор - це узагальнення скаляра чи вектора. Одним словом, тензори є багатовимірні масиви, які мають деякі динамічні властивості. Вектор - це одномірний тензор, тоді як двовимірні тензори є матрицями (рисунок 3.3).

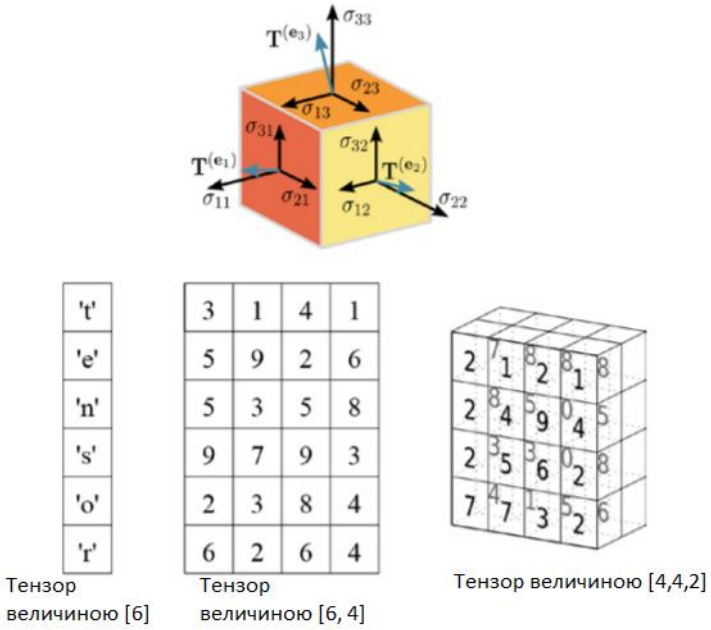


Рисунок 3.3. Тензори різних розмірностей

Тензори можуть бути двох типів: постійні або змінні.

Ранжування тензорів іноді може бути заплутаним, але з точки зору тензорів, ранг просто вказує кількість необхідних напрямків для опису властивості об’єкта, маючи на увазі розміри масиву, що міститься в самому тензорі. Розбивши це на різні об’єкти, скаляр не має ніякого напрямку і, отже, автоматично стає тензором рангу 0, тоді як вектор, який можна описати, використовуючи лише один напрямок, стає тензором першого рангу. Наступний об'єкт, який є матрицею, потребує двох напрямків для його опису і стає тензором другого рангу.

Форма тензорів – це умовне позначення кількості вимірів у тензорі. Тензор нульового рангу буде позначений як [], першого рангу - [a], другого - [a, b], де a, b – умовні позначення цифри, яка буде відповідати розміру виміру, наприклад для матриці 3x3 - [3, 3] [12].

3.1.3. TensorFlow – поняття «потоку»

Тепер йде друга частина TensorFlow: flow, або ж «поток». Це в своїй основі є фреймворком обчислення графів, який використовує для своїх обчислень тензори. Типовий граф складається з двох сутностей: вузлів та ребер, як показано на малюнку 1. Вузли також називають вершинами.

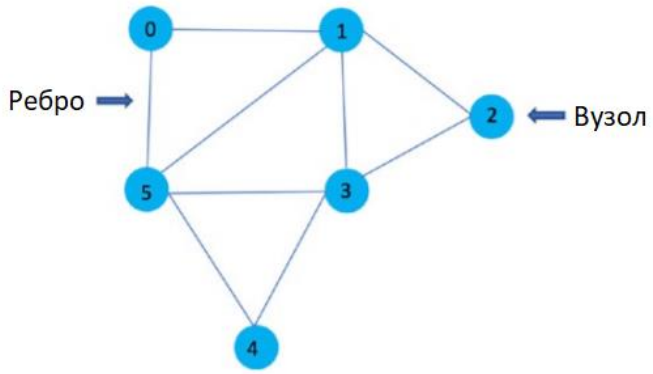


Рисунок 3.4. Типове зображення графу

Ребра по суті є зв'язками між вузлами, через які протікають дані, і вузли, де здійснюються фактичні обчислення. Загалом, графік може бути циклічним або ациклічним, але в TensorFlow, він завжди ациклічний. Він не може починатися і закінчуватися в одному і тому ж вузлі.

Давайте розглянемо простий обчислювальний граф, який показано на рисунку 3.5, та деякі його властивості.

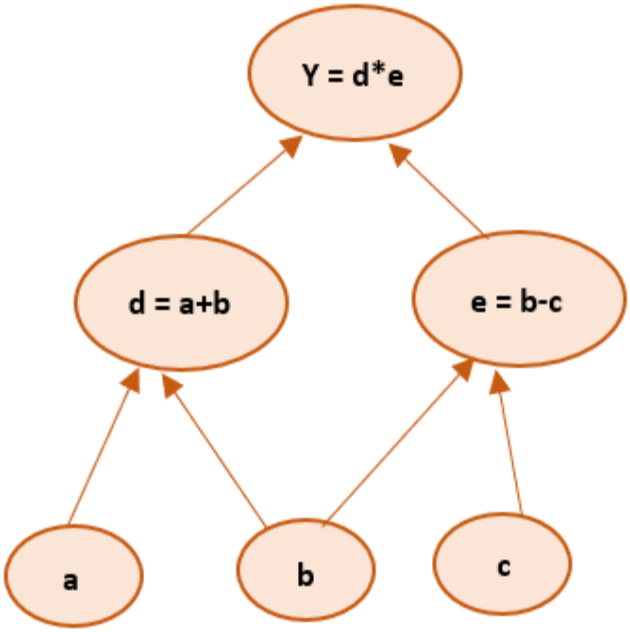


Рисунок 3.5. Обчислювальний граф

Вузли на графіку вказують на якесь обчислення, наприклад додавання, множення, ділення тощо, окрім листових вузлів, які містять фактичні тензори з постійними або змінними значеннями, щодо яких треба виконати певну дію. Ці тензори протікають через краї або з'єднання між вузлами, і обчислення на наступному вузлі призводить до утворення нового тензора. Отже, у зразковому графу новий тензор d створюється за допомогою обчислення у вузлі з використанням інших тензорів a і b. Варто зауважити, що на цьому графу

зазначено, що обчислення відбуваються лише на наступному етапі після листових вузлів, тоді як листові вузли можуть бути лише простими тензорами, які стають вхідними даними для обчислення наступного вузла, що протікає через ребра. Також можна представити обчислення на кожному вузлі через ієрархічну структуру. Вузли на одному рівні можуть виконуватися паралельно, немає взаємозалежність між ними. У цьому випадку d і e можна обчислити паралельно одночасно. Цей атрибут графа допомагає розраховувати обчислювальні графи розподіленим способом, що дозволяє використовувати TensorFlow для масштабних застосувань [12].

3.2. Практична реалізація дослідження

3.2.1. Налаштування TensorFlow

Для початкового тренування та подальшого практичного тестування створеної у ході дослідження системи було використано набір із 400 зображень, які були поділені на 2 групи по 200 зображень відповідно «гарної постаті» та «поганої постаті».

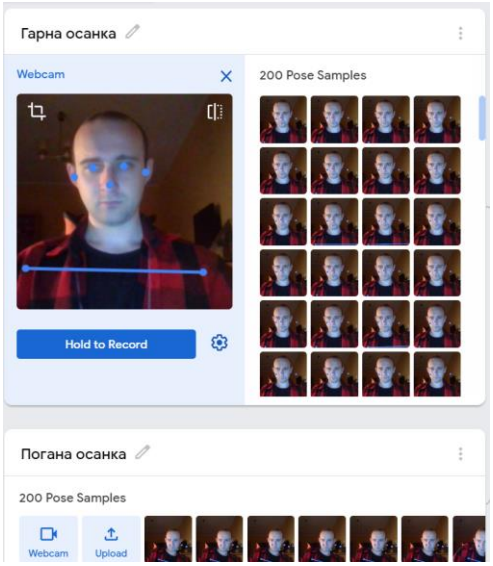


Рисунок 3.6. Додавання зображень до тестувальних наборів

На рисунку 3.6 відображено процес додавання тестувального набору зображень, під час якого TensorFlow автоматично розпізнає зображення людини та будує відповідний «скелет» постаті.

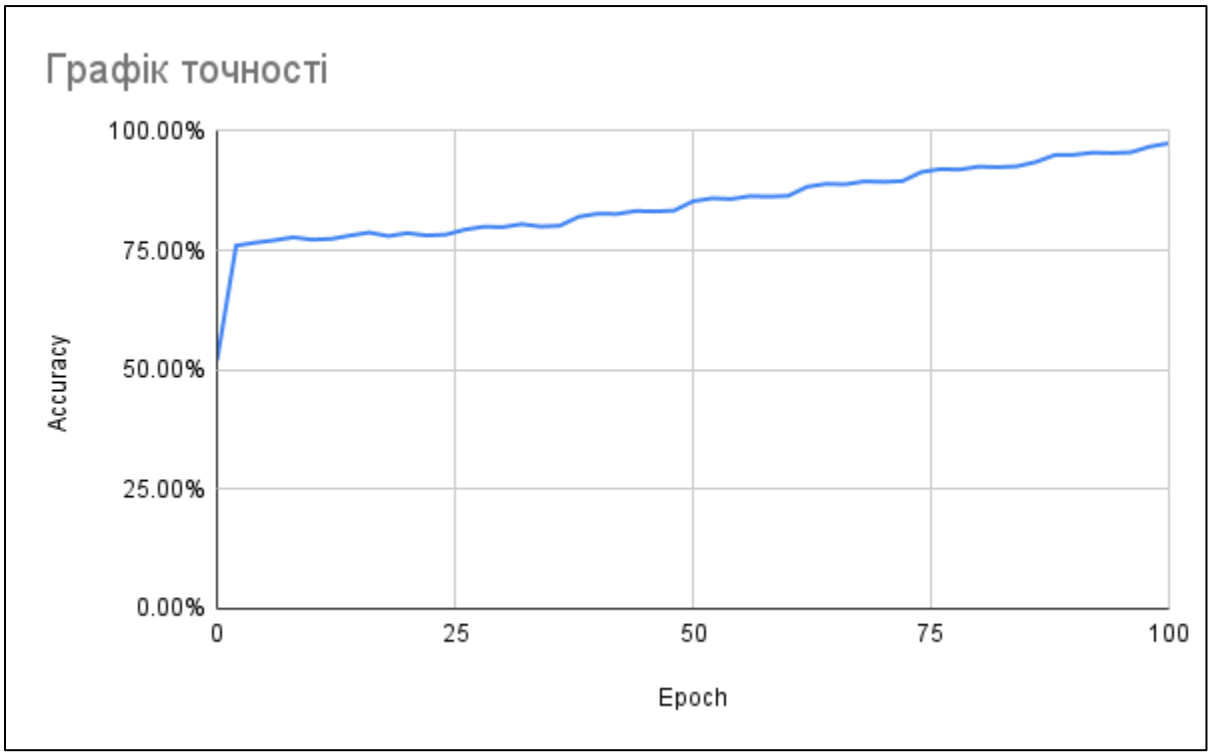


Рисунок 3.7. Графік отриманої точності при тестуванні системи

Варто зазначити, що на практичному застосуванні була досягнена трохи більша точність – 97%, ніж вказана у пункті 2.1.6 даної роботи. Це пов’язане з тим, що на практиці більш реальні, ближчі до того, як справді буде сидіти людина, дані дають точніші результати, ніж тренувальні набори.

Наступним кроком було відображення «скелету» осанки у режимі «он-лайн» з живого зображення з камери.

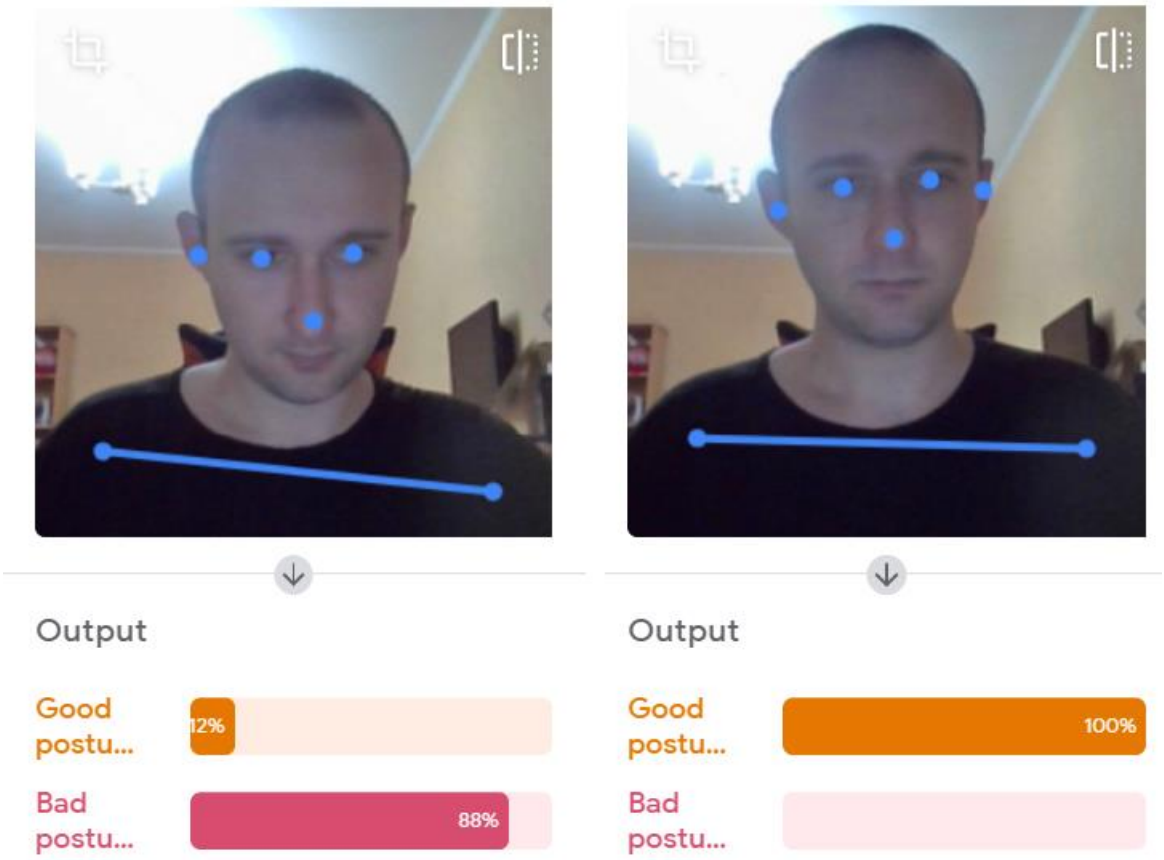


Рисунок 3.8. Живий аналіз зображення з веб-камери з відповідними позначеннями постаті у даний момент

На рисунку 3.8 зображено як система у живому часі аналізує зображення та видає відсоткову ймовірність приналежності постаті людини з відео-потоку веб-камери до групи «поганої» чи «хорошої» осанки.

3.2.2. Розширення для Google Chrome

Для практичного застосування результатів дослідження, було запропоновано створити простий додаток для Google Chrome, який при роботі у браузері буде використовувати веб-камеру користувача, обмінюватися даними з створеною у досліджені системою та повідомляти користувача про стан його осанки.

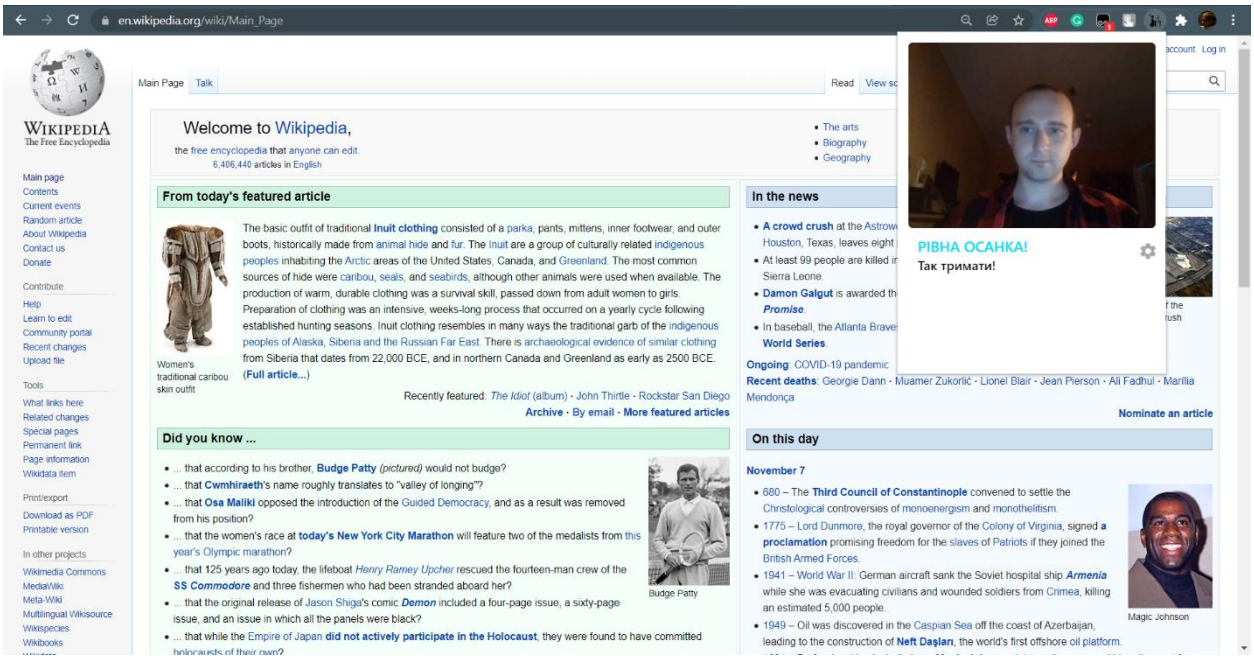


Рисунок 3.9. Загальний вигляд створеного розширення з повідомленням про «гарну» осанку

На рисунку 3.9 зображено вікно розширення, яке відкрите у браузері Google Chrome на головній сторінці Вікіпедії. У вікні розширення міститься живе зображення з веб-камери та текст про стан осанки людини.

Іншою можливістю створеного розширення є можливість працювати «у фоновому режимі», завдяки чому користувач може отримувати повідомлення про погану осанку при закритому віконці розширення. Це є корисним, оскільки

користувач не має постійно відкривати-закривати розширення, щоб відслідкувати стан своєї осанки.

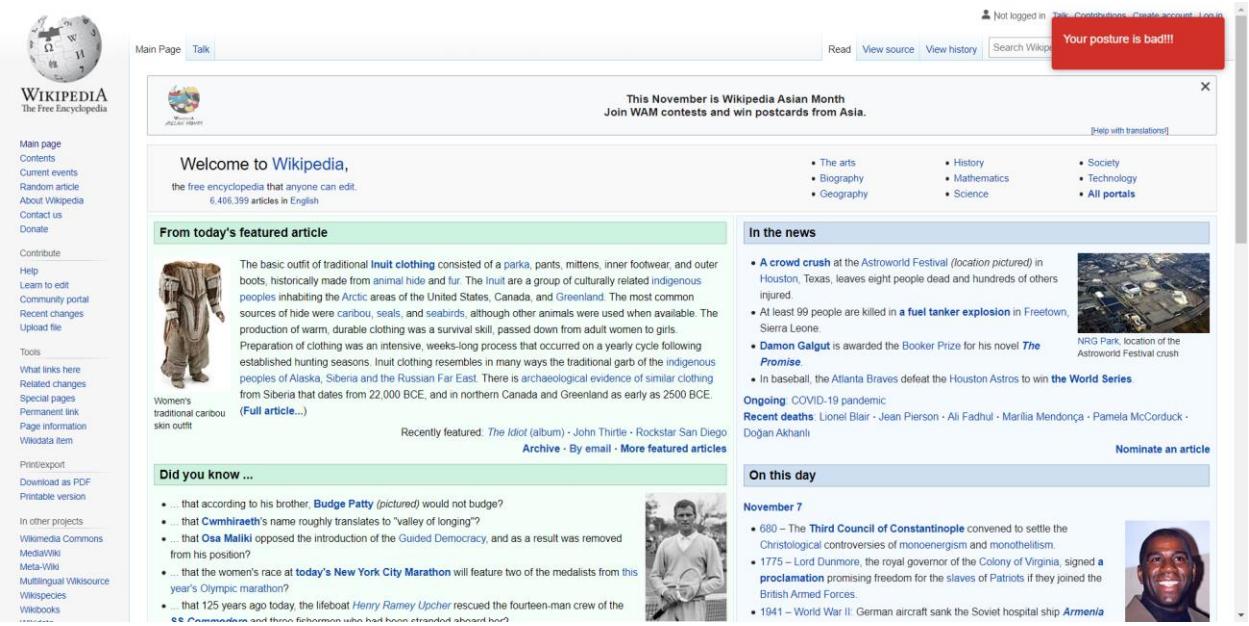


Рисунок 3.10. Вікно браузеру з відкритою сторінкою на якій міститься повідомлення про погану осанку

На рисунку 3.10 зображено червоний прямокутник, який є повідомленням про «погану» осанку користувача. Повідомлення відображене при закритому віконці розширення, тому користувач може займатися своїми задачами у браузері та отримувати повідомлення про осанку в будь-який час. Також у розширенні був доданий звуковий супровід повідомлення, щоб додатково повідомити користувача.

ВИСНОВКИ

У ході виконання дипломної роботи було проведено дослідження методів машинного навчання та нейронних мереж, які використовуються для розпізнавання та аналізу людської постаті. Результатом дослідження є створення гібридної системи методів машинного навчання та згорткової нейронної мережі, які дозволяються аналізувати осанку людини із зображення у режимі реального часу.

Як практичну реалізацію запропонованої системи, було створено додаток-розширення для Google Chrome, який аналізує осанку користувача у режимі реального часу та сповіщає його у разі незадовільної осанки, а також надає зображення перегляду зображення з постійним аналізом у проміжку декількох секунд.

СПИСОК ЛІТЕРАТУРИ

1. Kulikajevas A. Detection of sitting posture using hierarchical image composition and deep learning / A. Kulikajevas, R. Maskeliunas, R. Damaševičius., 2021. – (PeerJ Computer Science). – (7:e442).

2. Kim Y. Classification of Children's Sitting Postures Using Machine Learning Algorithms / Y. Kim, Y. Son, W. Kim., 2018. – 8 с. – (MDPI). – (1280).

3. Roh J. Sitting Posture Monitoring System Based on a Low-Cost Load Cell Using Machine Learning / J. Roh, H. Park, K. Lee., 2018. – 18 с. – (MDPI). – (208).

4. Zhou Z. Machine Learning / Zhi-Hua Zhou. – Singapore, 2016. – 460 с. – (Tsinghua University Press).

5. Shalev-Shwartz S. Understanding Machine Learning: From Theory to Algorithms / S. Shalev-Shwartz, S. Ben-David. – New York, 2014. – 449 с. – (Cambridge University Press).

6. Christopher M. B. Pattern Recognition and Machine Learning / Bishop Christopher M. – New York, 2006. – 749 с. – (Springer Science+Business Media).

7. Buontempo F. Genetic Algorithms and Machine Learning for Programmers / Frances Buontempo., 2019. – 234 с. – (The Pragmatic Programmers).

8. Reinecke S. A continuous passive lumbar motion device to relieve back pain in prolonged sitting / S. Reinecke, R. Hazard, K. Coleman // Advances in Industrial Ergonomics and Safety IV / S. Reinecke, R. Hazard, K. Coleman. – London, 2002. – (Taylor & Francis). – С. 971–976.

9. Winkel K. Evaluation of foot swelling and lower-limb temperatures in relation to leg activity during long-term seated office work / K. Winkel, J. Jorgensen. // Ergonomics. – 1986. – №29. – С. 313–328.

10. Naqvi S. Study of forward sloping seats for VDT workstations / S. A. Naqvi. // Journal of Human Ergology. – 1994. – №23. – С. 41–49.

11. Hoy D. The epidemiology of low back pain / D. Hoy, P. Brooks, R. Blyth. // Best Practice & Research: Clinical Rheumatology. – 2010. – №24. – С. 769–781.

12. Mattmann C. Machine Learning with TensorFlow / Chris Mattmann. – Shelter Island, NY: Manning Publications Co., 2020. – 454 с.

13. Machine learning education [Електронний ресурс] – Режим доступу до ресурсу: https://www.tensorflow.org/resources/learn-ml.

ДОДАТОК А. ПРЕЗЕНТАЦІЯ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Магістерська робота

«РОЗРОБКА СИСТЕМИ КОНТРОЛЮ ОСАНКИ ЛЮДИНИ ПРИ РОБОТІ ЗА КОМП'ЮТЕРОМ НА ОСНОВІ МЕТОДІВ МАШИННОГО НАВЧАННЯ»

Виконав: студент групи ПДМ-61 Бріт Я.О.

Керівник: д.т.н., доц., доцент кафедри ІПЗ Жебка В.В.

Київ - 2021

МЕТА, ОБ'ЄКТА ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: зменшення негативного впливу сидячого образу роботи на здоров'я людини за рахунок стеження за осанкою людини під час роботи за комп'ютером за допомогою додатку-розширення для веб-браузера Google Chrome та з використанням машинного навчання.

Об'єкт дослідження: стеження за осанкою людини під час роботи за комп'ютером.

Предмет дослідження: методи машинного навчання, відкриті бібліотеки та сервіси, мови програмування, браузер Google Chrome.

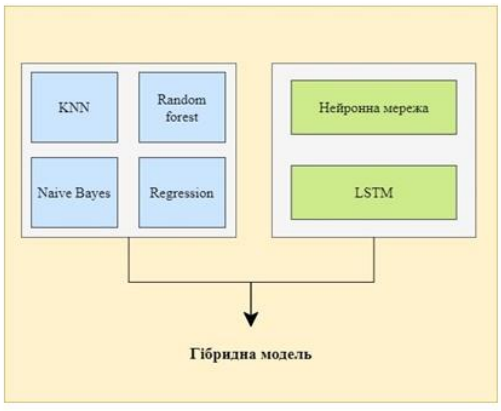
АНАЛІЗ ІСНУЮЧИХ РОБІТ

- «Detection of sitting posture using hierarchical image composition and deep learning» – стаття литовсько-польських науковців. У роботі пропонується глибока рекурентна ієрархічна мережа побудована на MobileNetV2, яка надає більшу гнучкість шляхом зменшення або ліквідації похибок, які пов'язані з низькою роздільною здатністю чи неповним обсягом обхвату людської постаті на зображенні. У роботі досягнена точність 91.47% при частоті кадрів 10fps.
- «Classification of Children's Sitting Postures Using Machine Learning Algorithms» – стаття шанхайських науковців, ідея якої полягає у використанні п'яти різних методів машинного навчання для визначення сидячої постаті людини та порівнянні їх результатів з подальшим визначення найточнішого методу. У роботі розглянуті такі методи машинного навчання - Hidden Markov Models, Naïve Bayes classifier, decision tree, multinomial logistic regression, and support vector machine. У роботі досягнена точність 95.3%.
- «Sitting Posture Monitoring System Based on a Low-Cost Load Cell Using Machine Learning» – стаття корейських науковців, ідея якої полягає у використанні різних методів класифікації зображень. Додатково у роботі створено систему сенсорів для покращення точності результатів. Система SPMS (Sitting posture monitoring systems) являє собою набір спеціальних сенсорів, які відслідковують сидячу постать людини, доповнюючи та уточнюючи дані зображень. Сенсори кріпляться на спинку або сидіння стільця. У роботі досягнена середня точність у 97.20% та максимальна у 97.94%.

3

ГОЛОВНА ІДЕЯ РОБОТИ

- Головною ідеєю даної магістерської роботи було впровадження оптимальної системи машинного навчання для визначення осанки людини. Як основний апарат для аналізу зображення людської постаті було обрано гібридний підхід з використанням згорткової нейронної мережі та архітектури довгої короткострокової пам'яті з додатковим використанням методів машинного навчання.



4

МАТЕМАТИЧНИЙ АПАРАТ ДОСЛІДЖЕННЯ

- *Згорткова нейронна мережа* – клас глибинних нейронних мереж, які переважно використовуються для аналізу візуальних зображень. Загальний принцип роботи згорткових нейронних мереж у розпізнаванні зображень – це зображення протиставлення матриці глибини насичення кольору.
- *«Випадковий ліс»* (англ. random forest, іноді random decision forest) – метод ансамблемового навчання для класифікації, регресії та інших завдань, що оперують деревами рішень під час тренування.
- *Метод k-найближчих сусідів* (англ. k-nearest neighbours, KNN) – метод машинного навчання з вчителем, який використовується у задачах класифікації та регресії.
- *Наївний баєсівський класифікатор* (англ. naive Bayes) – один з простих ймовірнісних класифікаторів, який базується на основі теореми Байєса.
- *Дерево рішень* (англ. decision tree) – передбачувальна модель, що відображає знання про певний об'єкт у множину рішень. Дерева рішень використовуються (як моделі прогнозування) для переходу від спостережень за предметом (представленим у гілках) до висновків про цільове значення елемента (представлене у листках).

5

ЗАГАЛЬНА СХЕМА АЛГОРИТМУ ПРОВЕДЕНОГО ДОСЛІДЖЕННЯ

6

ПРАКТИЧНИЙ РЕЗУЛЬТАТ

- Для практичного застосування результатів дослідження, було запропоновано створити простий додаток для Google Chrome, який при роботі у браузері буде використовувати веб-камеру користувача, обмінюватися даними з створеною у досліджені системою та повідомляти користувача про стан його осанки.

- Іншою можливістю створеного розширення є можливість працювати «у фоновому режимі», завдяки чому користувач може отримувати повідомлення про погану осанку при закритому віконці розширення. Це є корисним, оскільки користувач не має постійно відкривати-закривати розширення, щоб відслідкувати стан своєї осанки.

7

ПРАКТИЧНИЙ РЕЗУЛЬТАТ

Скриншот створеного додатку з живим зображенням з вебкамери та «онлайн» аналізом осанки.

8

ДОСЛІДЖЕННЯ ТА ПРАКТИЧНА РЕАЛІЗАЦІЯ – РЕЗУЛЬТАТИ

- У результаті дослідження створено систему, яка є досить простою для практичного розгортання (необхідно підключити модуль до майбутньої програми) та має здатність працювати напряму із зображенням відеопотоку.
- У таблиці наведені показники створеної системи у фінальному її стані після тренування на наборі даних у розмірі ~400 зображень

Параметр	Точність	Влучність	F-міра	Повнота
Mean	91,06	0,91	0,91	0,91
Mean, SD	91,31	0,92	0,92	0,92
Mean, SD, SR	91,48	0,92	0,92	0,92
Mean, SD, SR, percentile	91,82	0,93	0,93	0,92
Mean, SD, SR, percentile, kurtosis	92,36	0,94	0,93	0,94
Mean, SD, SR, percentile, kurtosis, skew	92,46	0,95	0,94	0,95

- На досить малому наборі даних досягнуто точності у ~ 93%, що є досить конкурентним показником інших робіт. Потенційно, можливе збільшення показника точності при кращих умовах тренування системи.

9

ВИСНОВКИ

У ході виконання дипломної роботи було проведено дослідження методів машинного навчання та нейронних мереж, які використовуються для розпізнавання та аналізу людської постаті. Результатом дослідження є створення гібридної системи методів машинного навчання та згорткової нейронної мережі, які дозволяються аналізувати осанку людини із зображення у режимі реального часу.

Як практичну реалізацію запропонованої системи, було створено додаток-розширення для Google Chrome, який аналізує осанку користувача у режимі реального часу та сповіщає його у разі незадовільної осанки, а також надає зображення перегляду зображення з постійним аналізом у проміжку декількох секунд

10

АПРОБАЦІЯ РОБОТИ

Статті:

1. Бріт Я.О. Формалізація задачі керування в інтелектуальних інформаційних технологіях на принципах нечіткої логіки// Зв'язок. №4 (152), 2021. *робота подана до друку*

Тези доповідей:

1. Бріт Я.О. Перспективи використання сучасних методів машинного навчання для покращення умов сидячих робочих місць шляхом аналізу та контролю осанки людини. // ЗБІРНИК МАТЕРІАЛІВ IX ВСЕУКРАЇНСЬКОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ МОЛОДИХ ВЧЕНИХ «НАУКОВА МОЛОДЬ-2021». – Київ: ДУ «ІГНС НАН України», 2021. – С.58-62.

11

ДЯКУЮ ЗА УВАГУ!

12