Зміст

Зміст

ВСТУП

Машинне навчання (ML) — це одна з найпопулярніших тем у розробці програмного забезпечення за останні кілька років. Машинне навчання відкриває багато нових можливостей для розробників, власників програм і кінцевих споживачів. Від більшої персоналізації до більш розумних рекомендацій, вдосконалених функцій пошуку, інтелектуальних помічників і додатків, які можуть бачити, чути та реагувати — технологія ML може покращити додаток і досвід його використання різними способами.

Машинне навчання є підмножиною штучного інтелекту (AI). Це дає комп'ютерам можливість навчатися на даних і поступово підвищувати якість результату на особливих завданнях — все це не покладаючись на програмування на основі чітких правил. Алгоритми машинного навчання знаходять природні закономірності всередині даних і приймають на їх основі майбутні рішення.

Темою даної роботи є класифікація емоцій людини на зображенні. Система з такою можливістю може стати в пригоді і для великих компаній, і для особистого користування задля підвищення емоційного інтелекту та навичок емпатії, які, на сьогоднішній день, займають ключове значення у розвитку особистості як в професійному так і суспільному повсякденному житті.

1 СУЧАСНІ ТЕХНОЛОГІЇ МАШИННОГО НАВЧАННЯ

1.1 Методи машинного навчання

Машинне навчання (далі ML) — це один з напрямів штучного інтелекту, особливістю якого є поступове покращення продуктивності у рішенні невідомих задач шляхом навчання в процесі застосування рішень множини подібних задач, без явного програмування. ML досліджує вивчення та побудову неявних алгоритмів, що можуть робити передбачення або приймати рішення на основі певних вхідних даних. Такий підхід дозволяє отримати надійні системи для вирішення складних задач, для яких неможливо побудувати явні алгоритми, зокрема ML використовують для фільтрації та класифікації багатовимірних даних (зображення, аудіофайли, тексти і т.д.), виявлення кібератак та забезпечення безпеки мереж, біоінженерії, для здійснення комп’ютерного зору та слуху тощо.

Для побудови таких систем використовуються засоби математичної статистики, численних методів, методів оптимізації, теорії ймовірностей, теорій графів та різних технік роботи з даними в цифровій формі.

У якості вхідних сигналів можуть бути використані різноманітні типи багатовимірних даних, такі як ознаковий опис, матриця відстаней, часовий ряд, відеоряд або зображення та більш складні випадки, коли використовують тексти, результати запитів до бази даних або дані у вигляді графу, але в такому разі їх, зазвичай, приводять до перших двох типів на етапі попередньої обробки даних та виділяння ознак.

При наданні вхідних даних у вигляді ознакового опису кожен об’єкт описується набором власних характеристик, тобто ознак, які можуть бути як числовими, так і категоріальними даними. Такий випадок зустрічається найчастіше.

Під часовим сигналом або рядом розуміють послідовність вимірювань у часі, а кожен вимір, у свою чергу, може бути як ознаковим описом, так і числом або вектором. Прикладом використання такого типу даних може бути навчання системи з використанням вимірів із різноманітних датчиків та сенсорів.

Окремим випадком є застосування матриці відстаней між об’єктами, де кожен елемент описується відстанями до кожного з інших об’єктів із навчальної вибірки.

Усі задачі машинного навчання, як правило, поділяють на дві широкі категорії, залежно від того, чи доступний системі, що навчається, навчальний «сигнал», або «зворотний зв'язок»: навчання з учителем, або контрольоване навчання та навчання без вчителя.

Навчання з вчителем — це процес навчання в ході якого випробувана система примусово навчається за допомогою прикладів «стимул-реакція». Між входами і еталонними виходами (стимул-реакція) може існувати деяка залежність, але вона невідома. Відома тільки кінцева сукупність прецедентів — пар «стимул-реакція», звана навчальною вибіркою. На основі цих даних потрібно відновити залежність (побудувати модель відносин стимул-реакція, придатних для прогнозування), тобто побудувати алгоритм, здатний для будь-якого об'єкта видати досить точну відповідь. Для вимірювання точності відповідей, так само як і в навчанні на прикладах, може вводитися функціонал якості.

Даний експеримент являє собою окремий випадок кібернетичного експерименту зі зворотним зв'язком. Постановка даного експерименту припускає наявність експериментальної системи, методу навчання і методу випробування системи або вимірювання характеристик.

Експериментальна система у свою чергу складається з випробовуваної (використовуваної) системи, простору стимулів одержуваних із зовнішнього

середовища та системи управління підкріпленням (регулятора внутрішніх параметрів). Як систему управління підкріпленням можна використати автоматичний пристрій, що регулює (наприклад, термостат), або людину-оператора (вчителя), здатну реагувати на реакції випробовуваної системи і стимули зовнішнього середовища шляхом застосування особливих правил підкріплення, що змінюють стан пам'яті системи. Схему системи навчання з учителем надано на рис.1.1.

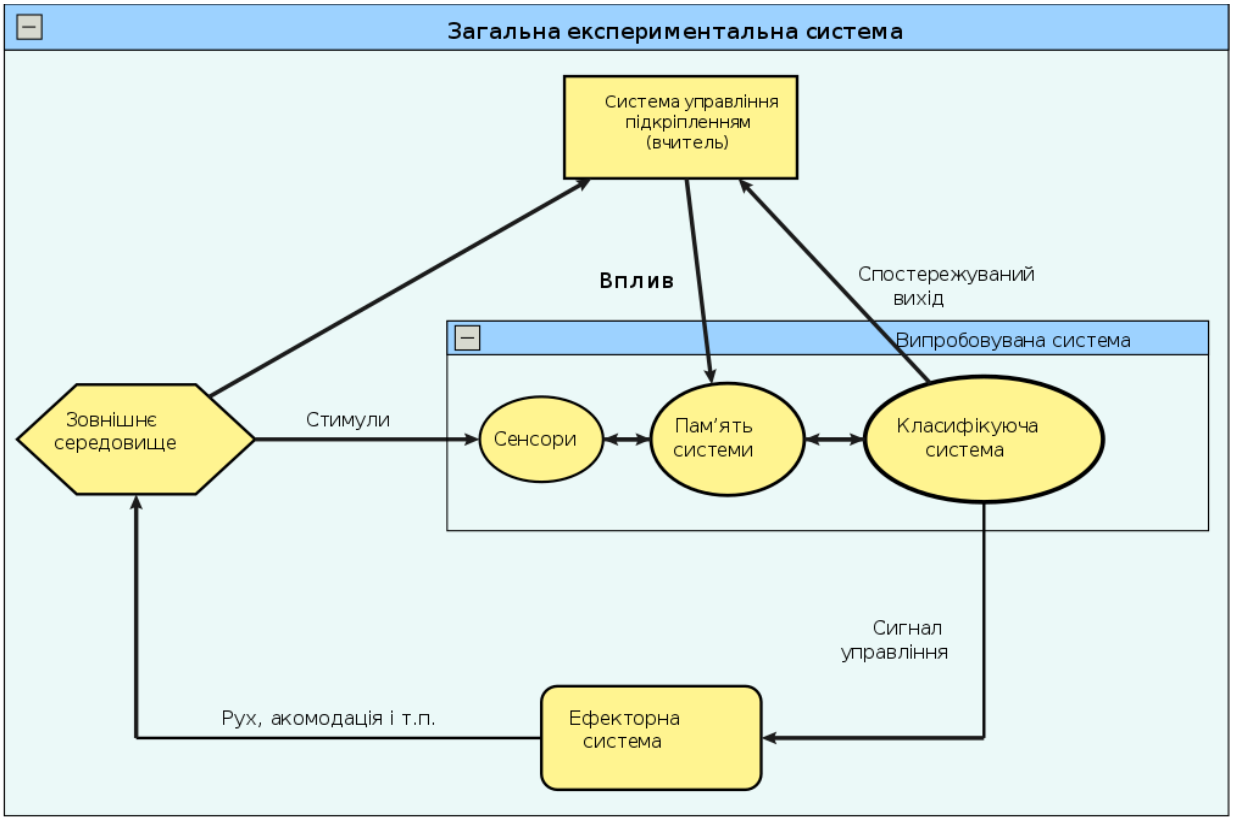

Рис.1.1 Навчання з учителем

Розрізняють два варіанти: реакція випробовуваної системи не змінює стан зовнішнього середовища та коли реакція системи змінює стимули зовнішнього середовища. Ці схеми вказують принципову схожість такої системи загального вигляду з біологічною нервовою системою.

Типи відгуків на навчання системи також можуть бути різноманітними. У задачах регресії та апроксимації це нескінченна множина відповідей, де вони є дійсними числами або векторами. У задачах класифікації та

розпізнавання образів множина можливих відповідей звичайна, а у задачах прогнозування відповіді характеризують майбутню поведінку процесу або явища.

Існує також поділ на S- та R-керовані системи. У S-керованої системи, тобто системи підкріплення з керуванням по стимулах, інформаційний канал від випробовуваної системи до системи підкріплення не функціонує. Незважаючи на це, дана категорія відноситься до навчання з учителем, оскільки в цьому випадку система підкріплення (вчитель) змушує випробувану систему виробляти реакції згідно певного правила, хоча й не береться до уваги наявність істинних реакцій випробовуваної системи.

У R-керованих систем, тобто систем підкріплення з керуванням по реакції, інформаційний канал від зовнішнього середовища до системи підкріплення не функціонує. Дана система, незважаючи на наявність системи управління, відноситься до спонтанного навчання, оскільки випробовувана система навчається автономно, під дією лише своїх вихідних сигналів незалежно від їх «правильності». При такому методі навчання для управління зміною стану пам'яті не потрібно ніякої зовнішньої інформації.

Навчання без вчителя — процес, під час якого алгоритми автоматичного навчання без нагляду виводять рішення з набору даних без використання відомих або позначених результатів. На відміну від контрольованого машинного навчання, ці методи не можуть бути безпосередньо застосовані до регресії або проблеми класифікації, тому що система не має уявлення щодо того, якими можуть бути значення вихідних даних, що унеможливлює навчання алгоритму. Навчання без нагляду може використовуватися для виявлення базової структури даних.

Неконтрольоване машинне навчання має на меті розкрити раніше невідомі закономірності в даних, але частіше ці моделі є поганими наближеннями до того, чого можна досягти при використанні контрольованого навчання. Крім того, оскільки система не знає, якими

повинні бути результати, не існує способу визначити, наскільки вони точні, що робить кероване машинне навчання більш придатним до реальних проблем. Некероване навчання доцільно використовувати тоді, коли немає даних про бажані результати, наприклад, визначення цільового ринку для абсолютно нового продукту. Проте, якщо задача полягає в кращому розумінні існуючої бази споживачів, навчання під наглядом є оптимальною методикою.

Деякі приклади застосування методів безконтрольного машинного навчання включають наступні задачі:

1) *Кластеризація* дозволяє автоматично розділяти набір даних на групи відповідно до схожості. Однак, часто кластерний аналіз переоцінює подібність між групами і не розглядає точки даних як індивіди. З цієї причини аналіз кластерів є поганим вибором для програм, таких як сегментація клієнтів та орієнтизація (targeting).

2) *Виявлення аномалій* може автоматично виявити нетипові дані у вхідному наборі. Це корисно для виявлення шахрайських транзакцій, виявлення несправних частин апаратних засобів або виявлення викидів, викликаних помилкою людини під час введення даних.

3) *Виокремлення асоціацій* визначає набори елементів, які часто зустрічаються поруч у наборі даних. Роздрібна торгівля часто використовує це для аналізу кошику покупця, оскільки дозволяє аналітикам виявляти товари, які часто купуються одночасно, і розробляти більш ефективні стратегії маркетингу та мерчандайзингу.

4) *Моделі латентної змінної* зазвичай використовуються для попередньої обробки даних, таких як зменшення кількості ознак у наборі даних (зменшення розмірності) або розкладання набору даних на декілька компонентів.

Шаблони, які розкриваються за допомогою методів некерованого машинного навчання також можуть стати в нагоді при подальшому впровадженні методів керованого машинного навчання. Наприклад, можна використовувати метод без вчителя для виконання кластерного аналізу даних, а потім скористатися кластером для кожного рядка у таблиці даних як додатковою функцією в моделі керованого навчання (напів-кероване навчання). Іншим прикладом є модель виявлення шахрайства, яка використовує показники виявлення аномалій як додаткову функцію.

Напівавтоматичне навчання або часткове навчання—це комбінація методів машинного навчання з учителем та без нього. У напів-контрольованому навчанні алгоритм вивчає дані, які включають як позначені, так і непозначені дані, зазвичай переважно непозначені.

Якщо є достатньо позначених даних, щоб створити точну модель, і немає можливостей або ресурсів, щоб отримати більше даних, можна використовувати методи напівавтоматичного навчання, щоб збільшити розмір навчальної вибірки.

Наприклад, це можна використати при розробці моделі для великого банку, призначеної для виявлення шахрайства. Деякі шахрайські дії вже відомі, але інші випадки шахрайства пройшли через перевірку непоміченими. Можна позначати набір даних за допомогою випадків шахрайства, про які відомо, але решта даних залишатиметься не позначеними. Для маркування даних можна використовувати алгоритм напівкерованого навчання, а також перенавчити модель з новим набором даних. Потім потрібно застосувати повторно навчену модель до нових даних, точніше виявляючи шахрайство. Однак не існує способу перевірити, що алгоритм згенерував позначення, які є на 100% точними, що призводить до менш надійних результатів, ніж традиційні керовані методи.

1.2 Алгоритми машинного навчання

Алгоритми є покроковими обчислювальними процедурами для вирішення проблеми, подібно до схем прийняття рішень, які використовуються для обробки інформації, математичного розрахунку та інших пов'язаних операцій. Машинне навчання спирається на алгоритми побудови моделей, які виявляють закономірності в даних. Таких алгоритмів існує досить багато:

- Лінійна регресія
- Логістична регресія
- Дерева рішень
- Метод опорних векторів, або SVM
- Наївний баєсів класифікатор
- Метод k-найближчих сусідів, або k-nn
- Кластеризація методом k-середніх, або k-means
- Random forest
- Алгоритми зменшення розмірності
- Алгоритми підвищення градієнта

Лінійна регресія використовується для оцінки реальних значень (вартість будинків, кількості дзвінків, загального обсягу продажів тощо) на основі безперервної змінної. Тут встановлюється зв'язок між незалежними та залежними змінними. При використанні лінійної регресії взаємозв'язок між даними моделюється за допомогою лінійних функцій, а невідомі параметри моделі оцінюються за вхідними даними. При розрахунках параметрів моделі лінійної регресії зазвичай застосовується метод найменших квадратів (МНК), але також можуть бути використані інші методи.

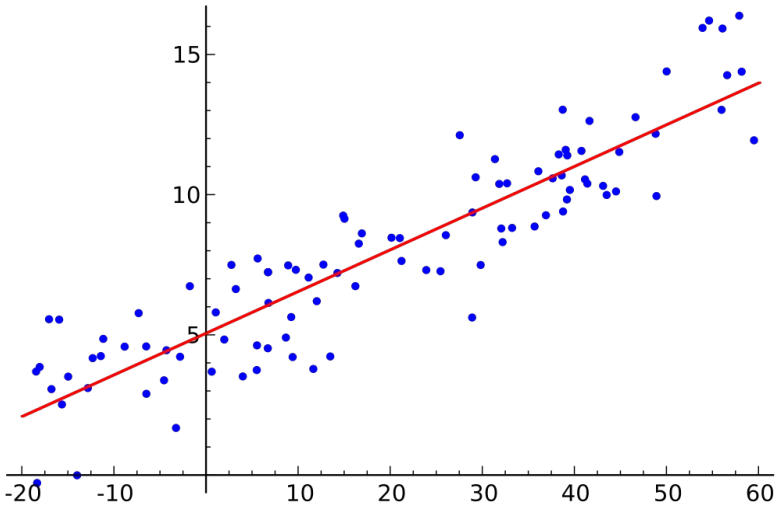

Рис. 1.2 Приклад класичної лінійної регресії з одною незалежною змінною

Загалом лінійна регресійна модель визначається у виді: $y=\beta_0+\beta_1 x_0+\ldots+\beta_k x_k+u$, де y — залежна пояснювальна змінна, $(x_1, x_2, \ldots, x_k)$ — незалежні пояснювальні змінні, u — випадкова похибка, розподіл якої в загальному випадку залежить від незалежних змінних але математичне сподівання якої дорівнює нулю. Приклад класичної лінійної регресії наведено на рис. 1.2.

Логістична регресія — це різновид множинної регресії, загальне призначення якої полягає в аналізі зв'язку між декількома незалежними змінними (званими також регресорами або предикторами) і залежною змінною. Бінарна логістична регресія застосовується у випадках, коли залежна змінна є категоріальною, тобто може приймати тільки два значення, 0 або 1, true або false. За допомогою логістичної регресії можна оцінювати вірогідність того, що подія настане для конкретного об’єкта або вірогідність його належності до певного класу.

В множинній лінійній регресії передбачається, що залежна змінна є лінійною функцією незалежних змінних. Її також можна застосовувати для оцінки ймовірності результату події, обчисливши стандартні коефіцієнти регресії. Проте це неминуче призведе до моделі з прогнозованими значеннями більшими за 1 і меншими за 0, які не припустимі для початкової задачі. Таким чином, множинна регресія ігнорує обмеження на діапазон

значень для вихідного результату. Залежність, що зв'язує ймовірність події і величину *y*, показано на рис. 1.3.

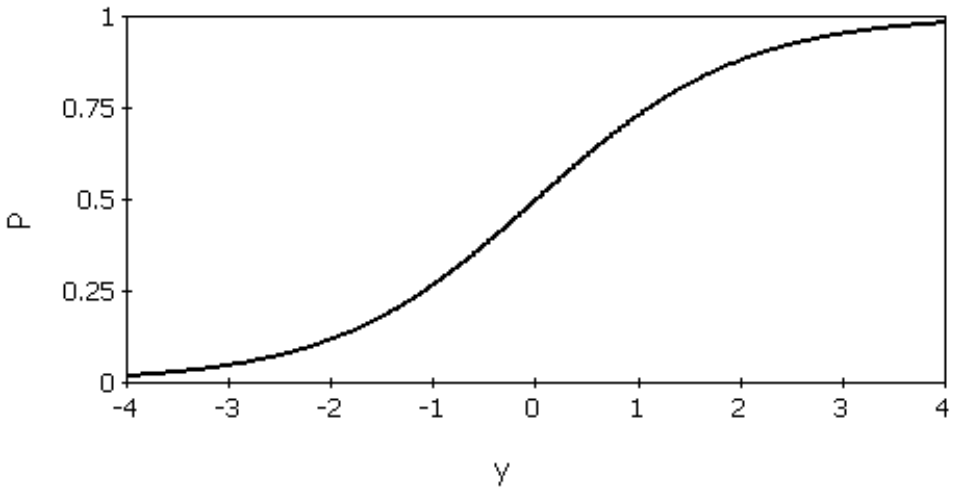

Рис. 1.3 Логістична крива

Для вирішення цієї проблеми завдання регресії може бути сформульоване інакше: замість передбачення бінарної змінної, потрібно передбачити безперервну змінну зі значеннями на відрізку [0,1] при будь-яких значеннях незалежних змінних. Це досягається застосуванням наступного регресійного рівняння (логіт-перетворення): $P=\frac{1}{1+e^{-y}}$, де P — ймовірність того, що подія відбудеться, або об’єкт належить до запитуваного класу. Теоретично P може приймати будь-яке значення. Оскільки логістичне перетворення вирішує проблему про обмеження на 0-1 ділянці для початкової залежної змінної (ймовірності), то ці перетворені значення можна використовувати в звичайному лінійному регресійному рівнянні. А саме, якщо провести логістичне перетворення обох частин описаного вище рівняння, можна отримати стандартну модель лінійної регресії.

Дерево рішень — це тип алгоритму навчання з учителем, який в основному використовується для задач класифікації. Даний алгоритм працює як для категоріальних, так і для безперервних залежних змінних та є деревоподібним графіком або моделлю рішень та їх можливих наслідків, включаючи результати випадкових подій. Це один із способів відображення

алгоритму, який містить тільки оператори умовного керування “if … then … else …”, а тому досить легко програмно описується.

Дерево рішень є структурою (рис. 1.4), подібною до блок-схеми, в якій кожен внутрішній вузол являє собою "тест" на атрибут, а кожна гілка являє собою результат тесту, і кожен лист вузла являє собою назву класу (рішення, прийняте після обчислення всіх атрибутів). Шляхи від кореня до листа являють собою правила класифікації.

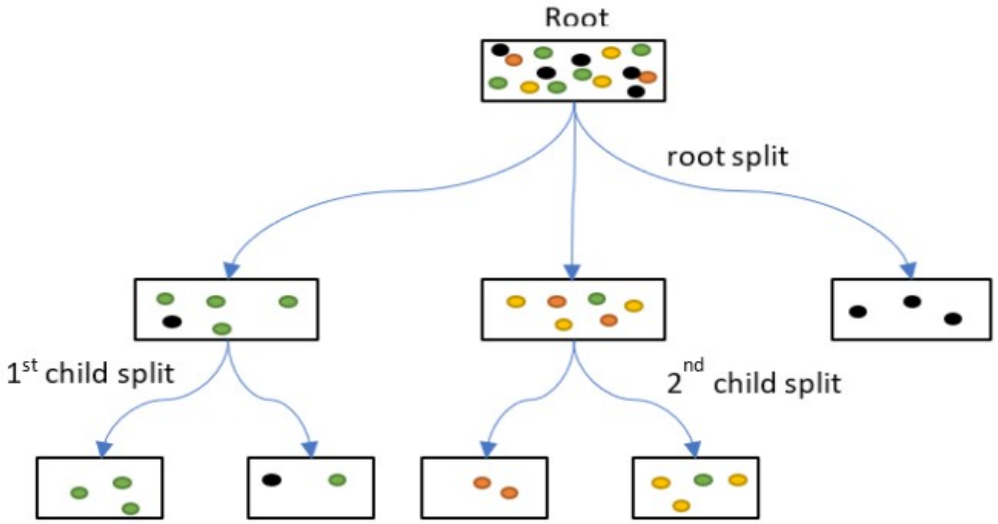

Рис. 1.4 Схема дерева рішень

Алгоритми навчання на основі дерев вважаються одними з найкращих і в основному використовуваних методів навчання з учителем. Методи, засновані на деревах, дозволяють моделям виконувати прогнозування з високою точністю, стабільністю і легкістю інтерпретації. На відміну від лінійних моделей, вони добре відображають нелінійні відносини. Вони пристосовані до вирішення будь-яких проблем (класифікація або регресія). Алгоритми дерев рішень називаються CART (Classification and Regression Trees).

Загальні терміни, які використовуються для дерев рішень:

- Root Node (кореневий вузол): представляє всю сукупність або зразок, і далі ділиться на два або більше однорідних набори.
- Splitting (розщеплення): це процес поділу вузла на два або більше підвузлів.)
- Decision Node (рішення вузла): коли підвузол розбивається на інші підвузли, то він називається рішенням вузла.
- Leaf/Terminal Node (лист/кінцевий вузол): вузли, що більше не розділяються називаються листами або кінцевими вузлами.
- Pruning (обрізка): процес виділення підвузлів із рішень вузла, протилежний до процесу розщеплення.
- Branch/Sub-Tree (гілка/під-дерево): підсекція всього дерева.
- Parent and Child Node (батьківський і дочірній вузол): вузол, який поділяється на підвузли, тоді як підвузли є дочірніми для батьківського вузла.

До переваг даного алгоритму відноситься простота у розумінні, швидкість, менша вибагливість до “чистоти” даних, можливість обробляти будь-які змінні, майже нульову кулькіть гіперпараметрів, які потрібно налаштовувати, а також те, що нелінійні відносини між параметрами не впливають на роботу дерева.

Проте дерева рішень мають суттєвий недолік у тому, що схильні до перенавчання, що повинно бути виправлено шляхом встановлення обмежень на параметри моделі та обрізку. Також вони можуть бути нестабільними, оскільки невеликі зміни даних можуть призвести до створення абсолютно іншого дерева, та дають відносно малу точність, у порівнянні з іншими моделями машинного навчання.

Random Forest є гнучким, простим у використанні алгоритмом машинного навчання, який є достатньо продуктивним, навіть без

налаштування гіперпараметрів у більшості випадків. Він також є одним з найбільш використовуваних алгоритмів, тому що є простим може бути використаним як для класифікації, так і для регресійних задач.

Random Forest — це контрольований алгоритм навчання. «Ліс», який він будує, — це ансамбль дерев рішень, які в більшості випадків навчаються методом «bagging». Загальною ідеєю методу bagging є те, що комбінація моделей навчання збільшує загальний результат. Простими словами random forest створює кілька дерев рішень і об'єднує їх разом, щоб отримати більш точний і стабільний прогноз.

Випадковий ліс є схожим до дерева рішень, але він додає додаткову випадковість моделі, під час вирощування дерев. Замість того, щоб шукати найважливішу функцію під час розбиття вузла, він шукає кращу функцію серед випадкової підмножини функцій. Це призводить до широкого розмаїття, що, зазвичай, призводить до отримання кращої моделі. Тому у випадковому лісі алгоритм розщеплення вузла враховує лише випадкову підмножину ознак. Також можна зробити дерева більш випадковими, додатково використовуючи випадкові пороги для кожної функції, а не шукати найкращі пороги (як звичайне дерево рішень).

Інша перевага алгоритму випадкового лісу полягає в тому, що можна легко виміряти відносну важливість кожної ознаки на прогнозі (наприклад використовуючи бібліотеку Sklearn).

Випадковий ліс являє собою набір дерев рішень, але є деякі відмінності. Якщо ввести навчальний набір з функціями та мітками в дерево рішень, воно сформулює певний набір правил, які будуть використовуватися для прогнозування. Для порівняння, алгоритм випадкового лісу випадковим чином обирає спостереження та функції для побудови декількох дерев рішень, а потім усереднює результати.

Інша відмінність від класичного дерева рішень полягає в тому, що

«глибокі» дерева рішень можуть зазнавати перенавчання. Random Forest запобігає перенавчанню у більшості випадків, створюючи випадкові підмножини функцій і будуючи невеликі дерева, використовуючи ці підмножини, а потім піддерева поєднуються.

Даний алгоритм вважається зручним і простим у використанні алгоритмом, оскільки гіперпараметри за замовчуванням часто дають хороший результат прогнозування. Кількість гіперпараметрів також не дуже висока і вони є зрозумілими. Основне обмеження випадкового лісу полягає в тому, що велика кількість дерев може зробити алгоритм повільним і неефективним для прогнозів у реальному часі. Загалом, ці алгоритми швидкі для навчання, але досить повільні для створення прогнозів після того, як вони пройшли навчання. Більш точний прогноз вимагає більшої кількості дерев, що призводить до більш повільної моделі. У більшості реальних додатків алгоритм випадкового лісу є досить швидким, але, безумовно, можуть виникнути ситуації, коли продуктивність під час виконання дуже важлива, а тому інші підходи краще підійдуть для вирішення початкової задачі.

Алгоритм випадкових лісів використовується в багатьох різних областях, таких як банківська справа, фондовий ринок, медицина та електронна комерція.

Метод опорних векторів — це метод аналізу даних для класифікації та регресійного аналізу за допомогою моделей з керованим навчанням з пов'язаними алгоритмами навчання, які називаються опорно-векторними машинами (support vector machines, SVM). Належить до сімейства лінійних класифікаторів і може також розглядатися як особливий випадок регуляризації за Тихоновим. Особливою властивістю методу опорних векторів є постійне зменшення емпіричної похибки класифікації і збільшення проміжку, тому метод також відомий як метод класифікатора з максимальним проміжком.

Основна ідея методу — переведення вихідних векторів в простір більш високої розмірності і пошук розділяючої гіперплощини з максимальним проміжком у цьому просторі. Дві паралельні гіперплощини будуються по обидва боки гіперплощини, що розділяє класи. Розділяючою гіперплощиною буде гіперплощина, що максимізує відстань до двох паралельних гіперплощин. Алгоритм працює в припущенні, що чим більша різниця або відстань між цими паралельними гіперплощинами, тим менше буде середня помилка класифікатора.

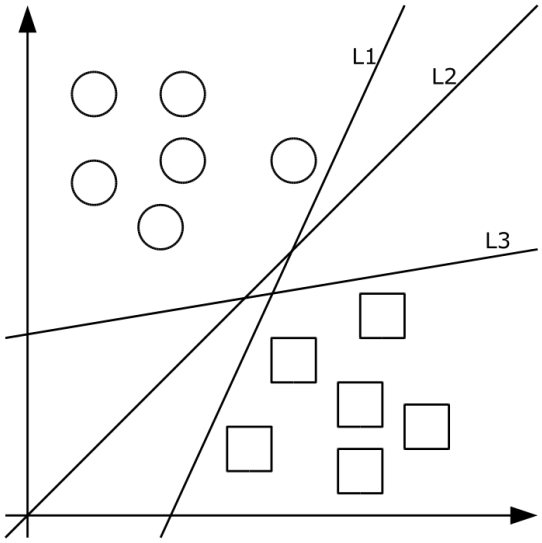

Рис.1.5 Класифікуючі розділяючі гіперплощини

Кожен об'єкт даних представляється як вектор (точка) в *p*-вимірному просторі (упорядкований набір *p* чисел). Кожна з цих точок належить тільки одному з двох класів. Питання полягає в тому, чи можна розділити точки гіперплощиною розмірності (*p*-1). Це — типовий випадок лінійної роздільності. Шуканих гіперплощин може бути багато, тому вважають, що максимізація проміжку між класами сприяє більш точній класифікації. Якщо сума відстаней до гіперплощини від двох найближчих до неї точок, що лежать по різні боки від неї, максимальна, то така гіперплощина називається оптимальною розділяючою гіперплощиною, а відповідний їй лінійний

класифікатор називається оптимально розділюючим класифікатором. На рис. 1.5 зображено декілька класифікуючих розділяючих гіперплощин, з яких оптимальною є лише L_2.

Наївний баєсів класифікатор — ймовірнісний класифікатор, що використовує теорему Баєса для вирішення задач класифікації.

Ідеальний байєсів класифікатор в якомусь сенсі є оптимальним. Його результат не може бути поліпшено, тому що у всіх випадках, коли можлива однозначна відповідь, він її дасть — а в тих випадках, коли відповідь неоднозначна, результат кількісно характеризує міру цієї неоднозначності. Разом із цим, в оптимальності криється основний недолік даного алгоритму: для його побудови потрібна вибірка, що містить всі можливі комбінації змінних, розмір якої експоненціально зростає із зростанням числа змінних, що називається “прокляттям розмірності”. Саме наївний баєсів класифікатор вирішує цю проблему, він дозволяє не вивчати взаємодію всіх можливих поєднань змінних, обмежившись лише впливом кожної змінної окремо, що скорочує розмір вибірки від експоненційного до лінійного.

Прикладом використання даного алгоритму є баєсівська фільтрація спаму. Під час навчання фільтра для кожного зустрітого в листах слова обчислюється і зберігається його «вага» — оцінка ймовірності того, що лист з цим словом — спам. У найпростішому випадку в якості оцінки використовується частота — відношення появи слова в спамі до появи всього. У більш складних випадках можлива попередня обробка тексту: приведення слів в початкову форму, видалення службових слів, обчислення «ваги» для цілих фраз, транслітерація та інше.

При повторній перевірці листа, ймовірність спаму обчислюється за формулою для безлічі гіпотез. В даному випадку гіпотези це слова, і для кожного слова достовірність гіпотези $P(A_i) = N_{word_i} / N_{words\,total}$ — частка цього слова в листі, а залежність події від гіпотези $P(B \mid A_i)$ — обчислена раніше

«вага» слова. Тобто «вага» листа в даному випадку — усереднена «вага» всіх його слів.

Віднесення листа до «спаму» чи «не-спаму» проводиться за тим, чи перевищує його «вага» певну межу, задану користувачем (зазвичай беруть 60-80%). Після прийняття рішення про лист, в базі даних оновлюються «ваги» для слів, що входили в нього.

Метод k-найближчих сусідів (k-nearest neighbor method або k-nn) — простий непараметричний класифікаційний метод, де для класифікації об'єктів у рамках простору властивостей використовуються відстані, пораховані до усіх інших об'єктів з навчальної вибірки. Далі відбираються k об’єктів, відстань до яких мінімальна, а клас визначається через клас, який зустрічається найчастіше серед k найближчих сусідів.

Алгоритм може бути застосовано до багатовимірних вибірок. Для цього перед застосуванням потрібно визначити функцію відстані, класичний варіант такої функції — евклідова метрика.

Різні атрибути можуть мати різний діапазон представлених значень у вибірці (наприклад атрибут А представлено в діапазоні від 0.1 до 0,5, а атрибут Б представлено в діапазоні від 1000 до 5000), тому значення дистанції можуть сильно залежати від атрибутів з великими діапазонами. Тому дані зазвичай підлягають нормалізації методами мінімакс-нормалізації і Z-нормалізації.

Деякі значущі атрибути можуть бути важливішими за інші, тому для кожного атрибута може бути задано у відповідності певна вага (наприклад обчислена за допомогою тестової вибірки і оптимізації помилки відхилення). Таким чином, кожному атрибуту *k* буде поставлено у відповідність вага z_k, так що значення атрибута будуть потрапляти в діапазон $[0; z_k \, max(k)]$ (для нормалізованих значень по мінімакс-методу). Наприклад, якщо атрибуту присвоєно вагу 2,7, то його нормалізовано-зважене значення буде лежати в

діапазоні $[0; 2{,}7]$.

Метод k-means — найбільш простий, але в той же час досить неточний метод кластеризації в класичній реалізації. Він розбиває безліч елементів векторного простору на заздалегідь відоме число кластерів k. Дія алгоритму така, що він прагне мінімізувати середньоквадратичне відхилення на точках кожного кластера. Основна ідея полягає в тому, що на кожній ітерації переобчислюється центр мас для кожного кластера, отриманого на попередньому кроці, потім вектори розбиваються на кластери знову відповідно до того, який з нових центрів виявився ближчим за обраною метрикою. Алгоритм завершується, коли на якийсь ітерації не відбувається зміни кластерів.

Проблеми алгоритму k-means:

- Необхідно заздалегідь знати кількість кластерів.
- Алгоритм дуже чутливий до вибору початкових центрів кластерів. Класичний варіант передбачає випадковий вибір кластерів, що дуже часто може бути джерелом похибки. Як варіант вирішення, необхідно проводити дослідження об'єкта для більш точного визначення центрів початкових кластерів.
- Не справляється із задачею, коли об'єкт належить до різних кластерів в рівній мірі або не належить жодному.

Під зменшенням розмірності (dimensionality reduction) в машинному навчанні мається на увазі зменшення числа ознак набору даних. Наявність в ньому надлишкових, неінформативних або слабо інформативних ознак може знизити ефективність моделі, а після такого перетворення вона спрощується, відповідно зменшується розмір набору даних в пам'яті і прискорюється робота алгоритмів ML на ньому. Зменшення розмірності може бути здійснено методами вибору ознак (feature selection) або виділення ознак (feature extraction).

Методи вибору ознак залишають деяку підмножину вихідного набору ознак, позбавляючись від зайвих. Основні переваги цього класу алгоритмів:

- Зменшення ймовірності перенавчання.
- Збільшення точності передбачення моделі.
- Скорочення часу навчання.
- Збільшення семантичного розуміння моделі.

Всі методи вибору ознак можна розділити на 5 типів, які відрізняються алгоритмами вибору зайвих ознак: фільтри, обгорткові методи, вбудовані методи, а також гібридні та ансамблеві методи. Фільтри вимірюють релевантність ознак на основі функції μ, а потім вирішують за правилом k, які ознаки залишити в результуючій множині ознак. Фільтри можуть бути одномірними та багатомірними. Одномірні фільтри визначають релевантність однієї ознаки по відношенню до вихідних міток. В такому випадку зазвичай вимірюють "якість" кожної ознаки і видаляють гірші. У багатомірних фільтрів функція μ визначає релевантність деякої підмножини вихідної множини ознак щодо вихідних міток.

Обгорткові методи (wrapper methods) знаходять підмножини шуканих ознак послідовно, використовуючи деякий класифікатор як джерело оцінки якості обраних ознак, цей процес є циклічним і триває до тих пір, поки не будуть досягнуті поставлені умови зупинки. Обгорткові методи враховують залежності між ознаками, що є перевагою в порівнянні з фільтрами, до того ж показують велику точність, але обчислення займають тривалий час, і підвищується ризик перенавчання.

Група вбудованих методів (embedded methods) схожа на обгорткові методи, але для вибору ознак використовується безпосередньо структура деякого класифікатора. У обгорткових методах класифікатор служить тільки для оцінки роботи на даній множині ознак, тоді як вбудовані методи використовують деяку інформацію про ознаки, яку класифікатори

привласнюють під час навчання.

Гібридні методи (Hybrid methods) комбінують декілька різних методів вибору ознак, наприклад, деяку множину фільтрів, а потім запускають обгортковий або вбудований метод. Таким чином, гібридні методи поєднують в собі переваги відразу декількох методів, і на практиці підвищують ефективність вибору ознак.

Ансамблеві методи (Ensemble methods) застосовуються більше для наборів даних з дуже великим числом ознак. В даному підході для початкової множини ознак створюється кілька підмножин ознак, і ці групи якимось чином об'єднуються, щоб отримати набір найбільш релевантних ознак. Це досить гнучка група методів, тому що для неї можна застосовувати різні способи вибору ознак і об'єднання їх підмножин.

Іншим способом зменшити розмірність вхідних даних є виділення ознак. Ці методи певним чином складають з вихідних ознак нові, котрі також повністю описують простір набору даних, але зменшуючи його розмірність і втрачаючи в репрезентативності даних, тому що стає незрозуміло, за що відповідають нові ознаки. Всі методи feature extraction можна розділити на лінійні і нелінійні.

Одним з найвідоміших методів лінійного виділення ознак є PCA (Principal Component Analysis, метод головних компонент). Основною ідеєю цього методу є пошук такої гіперплощини, на якій при ортогональної проекції всіх ознак максимізується дисперсія. Дане перетворення може бути зроблено за допомогою сингулярного розкладання матриць і створює проекцію тільки на лінійні багатовимірні площини, тому і метод знаходиться в категорії лінійних.

До нелінійних методів, можуть бути віднесені методи, що відображають початковий простір ознак на нелінійні поверхні або топологічні різноманіття. Одним з таких алгоритмів є t-SNE (t-distributed

Stochastic Neighbor Embedding, тобто стохастичне вкладення сусідів з t-розподілом). Даний метод складається з двох етапів: спочатку будується розподіл ймовірностей по всім парам точок набору даних, кожна умовна ймовірність $p_{j|i}$ якого означає наскільки точка X_j близька до точки X_i при гаусовому розподілі навколо X_i. Даний розподіл як метрику схожості використовує евклідова відстань. Алгоритм намагається отримати відображення з точок розмірності $\mathbb{R}^k$ в меншу розмірність $\mathbb{R}^d$, для цього вводиться ще одby розподіл, що описує наскільки точки з нового простору схожі одна на одну, але використовуючи при цьому t-розподіл Стьюдента з одним ступенем свободи. Як метрика схожості двох розподілів використовується дивергенція Кульбака-Лейблера, і щоб знайти точки нової розмірності d запускається градієнтний спуск для мінімізації цієї величини.

Посилення градієнтів є технікою для регресійних і класифікаційних задач, яка виробляє модель прогнозування у вигляді ансамблю слабких моделей прогнозування, зокрема дерев рішень. Вона будує модель у фазовому режимі, і узагальнює їх, дозволяючи оптимізувати довільну диференційовану функцію втрат.

Ця методика використовує логіку, в якій наступні предиктори вчаться на помилках попередніх предикторів. Таким чином, спостереження мають нерівну ймовірність появи в наступних моделях, а найчастіше з’являються моделі з найбільшою похибкою. (Таким чином, спостереження не вибираються на основі процесу завантаження, а на основі помилки). Прогнози можуть бути обрані з ряду моделей, таких як дерева рішень, регресори, класифікатори тощо. Оскільки нові предиктори вчаться на помилках, вчинених попередніми предикторами, потрібно менше часу/ітерацій для досягнення близьких до реальних прогнозів. Але потрібно ретельно обирати критерії зупинки, адже може статися перенавчання.

1.3 Особливості обробки зображень

У зв’язку з тим, що комп’ютер не здатен сприймати зображення так, як це робить людина, для того, аби навчити систему класифікувати зображення, потрібно їх привести у числовий вигляд. Це можна зробити двома шляхами — використовуючи відтінки сірого та використовуючи RGB кольори.

Під час використання першого способу, зображення буде переведено у відтінки сірого (діапазон сірих відтінків від білого до чорного). Комп'ютер призначатиме кожному пікселю значення, засноване на тому, наскільки він темний. Всі числа поміщаються в масив, і комп'ютер виконує обчислення на цьому масиві. Приклад конвертування зображення наведено на рис. 1.6.

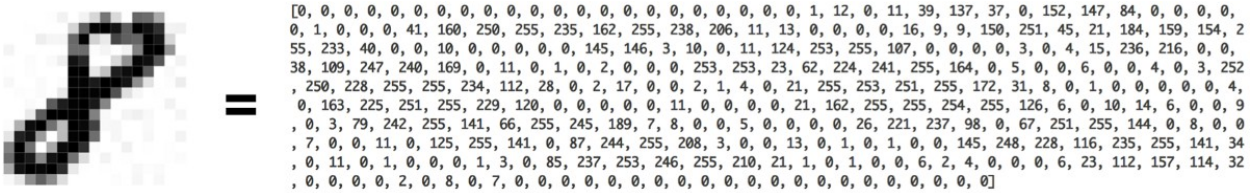

Рис. 1.6 Перетворення зображення у послідовність числових значень

Кольори можуть бути представлені у вигляді значень RGB (комбінація червоного, зеленого та синього від 0 до 255). Система може витягти RGB-значення для кожного пікселя і також зберігати результат у масив для подальшої інтерпретації. Коли система інтерпретує нове зображення, вона перетворює зображення на масив, а потім порівнює сукупності чисел з вже відомими об'єктами. Після цього відбувається розподіл оцінок довіри для кожного класу. Клас з найвищим показником довіри зазвичай є прогнозованим.

Однією з найпопулярніших методик, що використовуються для підвищення точності класифікації зображень, є Convolutional Neural Networks (CNN). Замість того, щоб подавати ціле зображення як масив чисел, зображення розбивається на декілька плиток, а машина намагається передбачити, результат для кожної з них окремо. На основі прогнозування

результатів всіх плиток робиться висновок щодо повного зображення. Це дозволяє комп'ютеру паралелізувати операції і виявляти об'єкт незалежно від того, де він знаходиться на зображенні.

Як і для інших випадків машинного навчання, точність вихідної системи прямо-пропорційна до величини вибірки. Але зазвичай датасети зображень містять не достатню їх кількість, до того ж вони можуть бути недостатньо різноманітними, що призводить до зменшення якості моделі. Виходом із цієї ситуації є виконання різноманітних операцій із зображеннями, наприклад, змінення розмірів, дзеркальні відображення, повороти, зміна кольорів (якщо використовуються кольорові зображення), штучне додавання шумів, тощо. На рис. 1.7 наведено одне зображення, яке було модифіковано різноманітними способами.

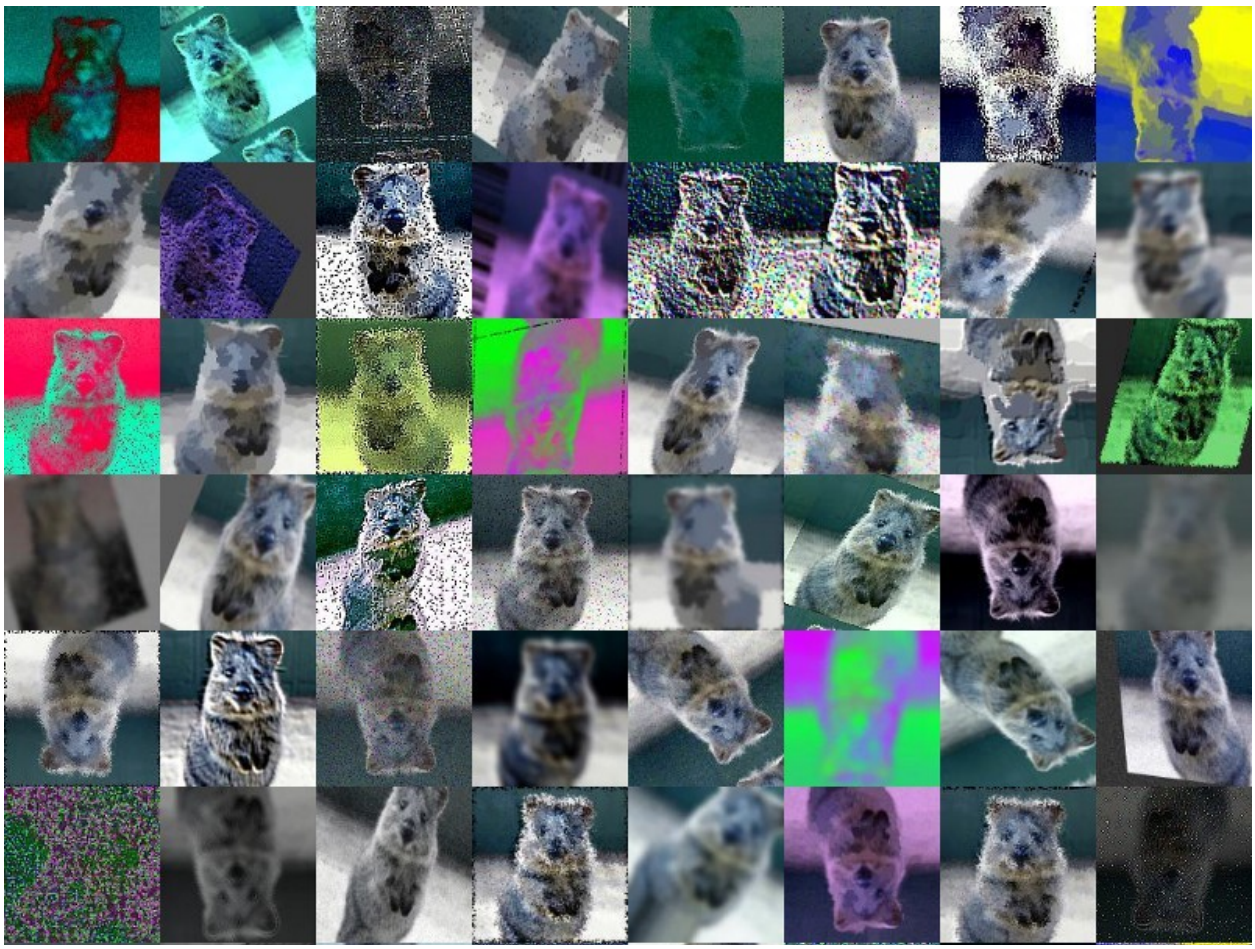

Рис. 1.7 Штучне збільшення навчальної вибірки

У деяких задачах, навпаки, кожне зображення потрібно привести до одного виду. Наприклад це стосується розпізнання конкретних людей на зображенні. Це потрібно тому, що після того як на зображенні буде знайдено лице, необхідно встановити чи є схожа людина у базі даних. Для цього після виявлення лиця на зображенні необхідно його обрізати, масштабувати та відцентрувати так, щоб не порушити паралельність ліній, приклад наведено на рис. 1.8.

Рис. 1.8 Центрування зображення для подальшої обробки

Після виконання простих операцій із зображенням необхідно виділити ознаки, за якими буде прийнято рішення щодо належності цього зображення до відповідного класу. Проте через нескінченну кількість зображень та їх велику різноманітність неможливо точно визначити ці ознаки, вони сильно залежать від задачі або типу застосування конкретної системи. Тому ознака визначається як деяка «цікава» частина зображення.

Оскільки ознаки використовуються як відправні точки та основні примітиви для наступних алгоритмів, загальний алгоритм часто буде лише настільки точним, наскільки точним є його детектор ознак. Основною якісною властивістю детектора ознак є повторюваність — виявлення однієї й тієї ж ознаки на двох або більше різних зображеннях однієї й тієї ж сцени. У зв’язку із вищеописаними особливостями, систему машинного навчання, яка працює із зображеннями, можна порівняти із чорною скринькою, адже неможливо дізнатися які саме ознаки використовуються.

Ознаки на зображеннях можуть бути контурами, особливими точками

або кутами, областями особливих точок, або хребтами.

Контури є точками, де присутня межа між двома областями зображення. Загалом контур може бути майже будь-якої форми та являє собою одновимірну структуру. На практиці контури зазвичай визначаються як множини точок зображення, які мають значну величну градієнта. Крім того, деякі поширені алгоритми з'єднують точки високого градієнта лінією для формування більш повного опису контуру. Ці алгоритми зазвичай накладають деякі обмеження на властивості контуру, такі як форма, плавність та значення градієнта. Приклад роботи детектора контурів наведено на рис. 1.9.

Рис. 1.9 Застосування до зображення методу знаходження контурів

Терміни кути та особливі точки використовуються часом як взаємозамінні, й відносяться до точкових ознак зображення, які мають локальну двовимірну структуру. Термін «кут» виник тому, що перші алгоритми спочатку виконували виявлення контурів, а потім аналізували контури для знаходження швидких змін у напрямку (кутів). Ці алгоритми згодом розвинулися до того, що явне виявлення контурів стало більше не потрібним, наприклад, при аналізі високих значень кривини градієнту зображення. Потім було відмічено, що так звані кути також виявлялися на тих частинах зображення, що не є кутами в традиційному розумінні (наприклад, можуть виявлятися невеликі яскраві плями на темному тлі). Ці

точки часто називаються особливими точками, але термін «кут» може використовуватись як традиційний.

Плями забезпечують додатковий опис структур зображення в термінах областей, на противагу до кутів, що є більш подібним до точок. Але описи плям часто можуть містити особливу точку (локальний максимум відповіді оператора, або центр мас), що означає, що багато детекторів плям можуть також розглядатися як оператори особливих точок. Детектори плям можуть виявляти такі області в зображенні, які є занадто згладженими для виявлення детектором кутів.

При знаходженні кутів, детектор зреагує на точки, що є контрастними у зменшеному зображенні, але можуть бути згладженими в початковому зображенні. В цьому випадку різниця між детекторами кутів та плям стає невиразною. В значній мірі це розрізнення може бути виправлено включенням відповідного поняття масштабу.

Для видовжених об'єктів використовується поняття хребтів. Хребтовий описувач, обчислюваний із зображення у відтінках сірого, можна розглядати як узагальнення серединної осі. З практичної точки зору хребет можна розглядати як одновимірну криву, що представляє вісь симетрії та має атрибут локальної ширини хребта, пов'язаний із кожною точкою хребта. Проте виділяти хребтові ознаки із загальних класів зображень у відтінках сірого алгоритмічно складніше, ніж контурні, кутові або плямові ознаки. Хребтові описувачі часто використовуються для виділяння доріг з аерофотознімків та судин у медичних зображеннях.

Після виділення ознак створюється ознаковий вектор — n-вимірний числовий вектор, що описує певний об’єкт. Часто ознаковий вектор може містити надто велику кількість ознак і вповільнювати роботу системи, у такому випадку до нього застосовують методики зменшення розмірності.

1.4 Практичні приклади застосування машинного навчання у роботі із зображеннями

Технології розпізнавання облич застосовуються для забезпечення безпеки в місцях великого скупчення людей, в системах охорони, для уникнення незаконного проникнення на територію об'єкта, пошуку зловмисників, пошуку підозрілих і потенційно небезпечних відвідувачів, для верифікація банківських карт, здійснення онлайн-платежів, у фототехніці, криміналістиці, в мобільних телефонах, соціальних мережах і т.д.

Серед найбільш успішних проектів по розпізнаванню осіб на державному рівні можна виділити найбільшу систему ідентифікації, яка була впроваджена Державним департаментом США. Її база даних містить 117 мільйонів фотографій дорослих американців, які, як правило, взяті з водійських прав. Крім цього успішно застосовується програма ФБР — Next Generation Identification, яка включає в себе розпізнавання обличчя, дактилоскопію і традиційну біометрику.

В Австралії з 2007 року діє програма SmartGate — це автоматизована система самообслуговування для здійснення прикордонного контролю, що керується австралійською прикордонною службою і розташована на контрольно-пропускних пунктах в залах прибуття у австралійських міжнародних аеропортах. SmartGate дозволяє власникам австралійських електронних паспортів, а також власникам електронних паспортів ряду інших країн, швидше проходити міграційний контроль. Електронне проходження паспортного контролю підвищує безпеку подорожей та заощаджує час службовців митниці. SmartGate використовує технологію розпізнавання облич для перевірки особистості мандрівника за даними, що зберігаються в чіпі в його біометричного паспорту, також особу перевіряють по міграційним базам даних. Аналогічна система також є у Великобританії — ePassport gates.

У Китаї активно застосовуються різні системи спостереження з

можливістю розпізнавання облич. За даними британської загальнонаціональної громадської телерадіомовної організації BBC, на вулицях Китаю, станом на кінець 2017 року, встановлено 170 мільйонів камер відеоспостереження, а до 2020 року кількість камер планують підвищити до 400 мільйонів. Більша частина з цих камер оснащена штучним інтелектом, який розпізнає обличчя громадян і порівнює їх з обличчями злочинців з державної бази даних. Також дане програмне забезпечення здатне визначити стать людини, її вік і національну приналежність.

Спочатку 2018 року служба новин Китаю повідомила, що співробітники митної служби, що працюють на високошвидкісній залізничній станції Чженчжоу в провінції Хенань, почали користуватися розумними окулярами — «GLXSS ME», які були вироблені китайською компанією «LLVision». Компанія LLVision стверджує, що на розпізнавання обличчя потрібно менше однієї секунди, а одночасно може бути здійснено захоплення до 10 облич.

Також в Китаї діє система, яка створена для здійснення контролю за виконанням правил дорожнього руху. У Шеньчжені це дозволило визначити 14 тисяч порушників правил дорожнього руху лише на одному з перехресть, а у Нінбо за півроку за допомогою цих систем вдалося зафіксувати 7800 випадків порушення правил дорожнього руху як пішоходами, так і велосипедистами.

Окрім застосування у безпеці, системи з функцією розпізнавання обличчя здатні збільшити ефективність комерційного підприємства в кілька разів. Вони можуть здійснювати авторизацію співробітників в інформаційних системах підприємства тільки при проході співробітників на робочі місця через біометричний пропуск, для забезпечення контролю часу і запобігання передачі пропусків стороннім особам, доступ співробітників по обличчю на територію, в офіс, закриті приміщення, а також пасивне спостереження в закритих зонах для виявлення сторонніх осіб.

Алгоритми машинного навчання широко застосовуються у сфері розваг. Платформи соціальних мереж ввели можливості розпізнавання облич декілька років тому. Прикладом цього є анімовані фільтри Snapchat, в яких використовувалася технологія розпізнавання облич, це дозволило користувачам додавати різноманітні маски, химерні фільтри та ефекти покращення зовнішності на фото. Надалі схожі функції були введені в відому соціальну мережу Instagram тощо.

Існує також система під назвою «DeepFace» — це система розпізнавання облич з глибинним навчанням, створена дослідницькою групою в Facebook. Вона ідентифікує людські обличчя в цифрових зображеннях використовуючи 9-шарову нейронну мережу з більш ніж 120 мільйонами значень ваг. Дана нейронна мережа була навчена за допомогою 4 мільйонів зображень, завантажених користувачами соціальної мережі Facebook. За даними компанії, точність системи складає 97% і є вищою, ніж точність системи, яку використовує ФБР — Next Generation Identification.

Окрім DeepFace, технології розпізнавання облич активно використовуються в різних програмах для зберігання фотографій, таких як Picasa, iCloud, Google Photos та інших. У цих програмах відбувається розпізнавання на фотографіях однакових облич та їх автоматичне групування в окремі альбоми для кожної людини.

У цифрових фотоапаратах вже більше 15 років широко застосовується функція визначення обличчя. Ця функція потрібна для автоматичної настройки параметрів фотографії, таких як фокусування та експозиція, а також для реалізації функції автоматичного спуску затвора при виявленні в кадрі посмішки.

Протягом останніх кількох років, в мобільних телефонах почали удосконалювати захист персональних даних за допомогою розблокування пристрою за обличчям власника. Компанія Apple в 2017 році представила функцію Face ID на смартфоні iPhone X.

У 2018 році Вчені зі Стенфордського університету розробили Superpower Glass — систему взаємодії з гарнітури Google Glass, яка допомагає дітям з аутизмом навчитися розпізнавати міміку інших людей.

Камера окулярів Google Glass знімає обличчя людей в поле зору дитини і передає цю інформацію в додаток, який здатен розпізнати вісім основних емоцій. Коли емоцію визначено, дитина чує її назву через динамік гарнітури і бачить відповідне зображення в правому куті окулярів.

У ході дослідження ефективності даної системи було виявлено, що при регулярних заняттях із цими окулярами, діти покращували свої соціальні та когнітивні навички.

1. 5 Постановка задачі

2 ПРОГРАМНІ ЗАСОБИ ДЛЯ МАШИННОГО НАВЧАННЯ

2.1 Порівняльні характеристики мов програмування у галузі

Від вибору мови програмування для подальшої розробки напряму залежать такі фактори, як швидкодія, доступність готових бібліотек та фреймворків, а також кількість матеріалів для навчання та отримання підтримки зі сторони спільноти розробників.

Рис.2.1 Дані дослідження "Developer Economics"

Наразі найбільшою популярністю користуються такі мови програмування як Python, C/C++, Java, R та JavaScript. Кожна з них має свої недоліки та переваги, а також деякі особливості застосування.

Згідно з даними "Developer Economics", які було зібрано з більш ніж двох тисяч анкет data scientists та machine learning розробників, дослідники порівняли перші п'ять результатів у рейтингу за кількома критеріями, розглянемо ці дані на рис.2.1.

По-перше, розглянемо загальну популярність мов для машинного навчання. Python є лідером у списку, при цьому 57% data scientists і ML розробників використовують його, а 33%—надають йому пріоритет під час розробки. Така ситуація склалася насамперед через великий прогрес фреймворків глибинного навчання на Python за останні 5 років, включаючи випуск TensorFlow та широкий вибір інших бібліотек.

Python часто порівнюється з R, але вони не можуть бути порівнянні з точки зору популярності: R є четвертим у загальному використанні (31%) і п'ятим у пріоритизації (5%). R фактично є мовою з найнижчим співвідношенням пріоритизації до використання серед п'яти лідерів (17%). Це означає, що в більшості випадків R є додатковою мовою, а не основним вибором. Це ж співвідношення для Python становить 58%, що є найбільшим серед п'яти мов-лідерів, а це свідчить про те, що тенденції використання Python є абсолютно протилежними тенденціям використання R. Окрім цього, Python також є основним вибором для більшості розробників.

C/C++ є не дуже далекими від Python, як у популярності (43%), так і за пріоритетами (19%). Java слідує за C/C++ та має 41% та 16% відповідно. Водночас JavaScript є п'ятим у використанні, з невеликим відривом від R за пріоритизацією (7%) і за популярністю (28%).

Респондентів питали також про інші мови, що використовуються в машинному навчанні, включаючи Julia, Scala, Ruby, Octave, MATLAB і SAS, але всі вони мали показники нижче 5% за пріоритетністю і нижче 26% від використання, тож не були включені у дослідження.

Дані показують, що найбільш вирішальним фактором при виборі мови для машинного навчання є тип проекту—область застосування. У розробників питали про 17 різних областей застосування, а також надавали можливість респондентам розповісти, що вони все ще вивчають варіанти, не активно працюючи в будь-якій області. У кінцевому дослідженні

пропонуються три найпопулярніші області для кожної мови: ті, де розробники надають пріоритети кожній мові найбільше і найменше.

Науковці з машинного навчання, що працюють над аналізом настроїв тексту, надають найбільші пріоритети Python (44%) і R (11%), а найменші—JavaScript (2%) і Java (15%), ніж розробники, що працюють в інших областях. На відміну від Python, Java, має високий пріоритет серед тих, хто працює над мережевою безпекою, захистом від кібер-атак і виявленням шахрайства, двома областями, де Python є найменш пріоритетним. Алгоритми захисту мережі та виявлення шахрайства побудовані або використовуються в основному у великих компаніях, особливо у фінансових установах, де Java є найпопулярнішою. У напрямках, які менш орієнтовані на великі підприємства, такі як обробка природної мови (NLP) та аналіз настроїв, розробники обирають Python, який пропонує простіший і швидший спосіб побудови високоефективних алгоритмів, завдяки великій колекції спеціалізованих бібліотек та швидкій точці входу.

Штучний інтелект (AI) в іграх (29%) та керування роботами (27%)—це дві області, в яких C/C++ використовуються найбільшою мірою, що обумовлено необхідним рівнем контролю, високою продуктивністю та ефективністю. В цих областях мова програмування більш низького рівня, як C/C++, яка поставляється з дуже складними бібліотеками AI, є звичайним вибором, тоді як R, призначений для статистичного аналізу та візуалізації, вважається в основному неактуальним. AI в іграх (3%) і керування роботами (1%)—дві області, де R має найменший пріоритет, подібно до розпізнавання усної мови (3%).

Окрім аналізу настроїв, R також користується відносно високим пріоритетом—у порівнянні з іншими областями застосування—у біоінженерії та біоінформатиці (11%), а Java має в цих напрямках найгірші показники (13%). Враховуючи давнє використання R у біомедичній статистиці, як всередині, так і за межами академічних кіл, не дивно, що вона

є однією з областей, де R використовується найбільше. Також, дані дослідження показують, що розробники, які є новими у галузі машинного навчання та досі вивчають варіанти, надають JavaScript більших пріоритетів, ніж іншим мовам (11%), а Java найменших—(13%). У багатьох випадках це розробники, які експериментують з машинним навчанням, використовуючи API для третіх сторін у веб-додатках.

По-друге, в даній області застосування, професійна підготовка також має вирішальне значення у виборі мови для машинного навчання. Найбільший пріоритет Python приділяється тими, для кого наука про дані та машинне навчання є основною професією або галуззю (38%). Те ж саме не можна сказати про R, який в основному має високий пріоритет від аналітиків даних і статистиків (14%), оскільки мова була спочатку створена для цих областей, замінивши S.

Front-end веб-розробники розширюють своє використання JavaScript для машинного навчання, 16% визначають його пріоритетним для цієї мети, залишаючись у стороні від складних та не пристосованих для цього напрямку C/C ++ (8%).

Ситуація серед інженерів вбудованих обчислювальних апаратів та електроніки склалася навпаки, тут C/C++ надають більшу перевагу ніж іншим мовам, уникаючи при цьому JavaScript, Java і R. Враховуючи інвестиції розробників в освоєння C/C++, для них не має сенсу використовувати мову, яка погіршить їх рівень контролю над програмою. До того ж, інженери часто працюють над подібними до їх галузі навчальними проектами, такими як проекти аналітики IoT, де апаратні засоби можуть змусити їх обрати саме більш низькорівневу мову програмування.

Для Java, більшість складають розробники front-end комп'ютерних додатків, яких значно більше, ніж серед інших мов (21%). Розробники на підприємствах, як правило, використовують Java у всіх проектах, включаючи машинне навчання. Вплив компанії в цьому випадку також видно з третього

фактора, який сильно співвідноситься з визначенням мовних пріоритетів—причиною для входу в машинне навчання. Найбільш поширеною причиною для вибору Java (27%) є пропозиція або вимога від компанії. Це найменш бажана мова (14%) серед тих, хто потрапив у ML тільки через власний інтерес, на відміну від Python (38%).

R обирають, у більшості випадків (7%), через те, що він використовується під час навчання в учбових закладах.

C/C++ є більш пріоритетним для тих, хто хоче покращити свої існуючі програми/проекти з машинним навчанням (20%) і менше для тих, хто сподівається створити нові висококонкурентні програми на основі машинного навчання (14%). Ці показники вказують на те, що C/C++ використовується в основному в інженерних проектах, а також у програмах IoT або AR/VR, які, швидше за все, вже написані на C/C++, до яких додаються функціональні можливості, що підтримуються ML. При створенні нової програми з нуля—особливо з використанням NLP для чат-ботів—немає особливої причини використовувати C/C++.

Нарешті, розробники, які потрапили в машинне навчання, щоб збільшити свої шанси на забезпечення високооплачуваних проектів, віддають перевагу JavaScript більше, ніж іншим мовам (8%).

Важливим фактором для вибору мови програмування є її швидкодія. Лідером у цьому питанні є C/C++. На другому місці перебуває Java. Ці мови є компільованими, тобто код зводиться у байт-код, перш ніж зберігатися як виконуваний файл. Натомість, R, JS та Python є інтерпретованими мовами, тобто код зберігається в тому ж форматі, в якому його було введено. Компільовані програми зазвичай працюють швидше, ніж інтерпретовані, оскільки інтерпретовані програми повинні переводитися у машинні команди кожного разу, коли вони виконуються. Проте для Python існує розширення Cython—яке, по суті, еквівалентне Python зі статичним типізацією, що дозволяє розробникам значно збільшити швидкість виконання програми.

Найбільша кількість бібліотек та пакетів інструментів сьогодні є у Python та R. Python також є мовою з найменшою точкою входу, тобто для написання програм на Python не потрібно глибоких знань цієї мови, що робить її найкращим вибором для початківця у ML. У цьому проекті також використано Python.

2.2 Фреймворки, бібліотеки та API для машинного навчання

Фреймворки машинного навчання постійно розвиваються. Штучний інтелект у поєднанні з правильною системою глибокого навчання збільшив загальний розмір того, чого можуть досягти різні підприємства, не виходячи за межі своїх предметних областей. Крім того, з постійно зростаючою кількістю організацій, які намагаються розширити спектр своїх послуг та можливостей, необхідно щоб будь-яка така компанія асимілювала як машинне навчання, так і прогностичні дослідження. Розглянемо найпопулярніші фреймворки та юіюліотеки для машинного навчання.

Рис. 2.2 Компанії, що застосовують TensorFlow

TensorFlow — це відкрита програмна бібліотека для машинного навчання, здатна вирішувати цілу низку задач, розроблена компанією Google для використання всередині компанії. 9 листопада 2015 року її було

випущено як відкрите програмне забезпечення. Вона може працювати на декількох центральних та графічних процесорах (включно з додатковими розширеннями CUDA для обчислень загального призначення на графічних процесорах).

В даний час TensorFlow займає перше місце в списку фреймворків для машинного навчання. Більшість розробників використовують TensorFlow, оскільки вона має велику спільноту підтримки і багато вбудованих функцій. Цей фреймворк використовується такими компаніями, як Google, Airbus, Airbnb, Coca-Cola, AMD, Nvidia, IBM та іншими (рис. 2.2), здебільшого через надзвичайно адаптовану інженерну структуру, великий спектр можливостей та підтримуваних архітектур. Зокрема, Google використовує TensorFlow для розпізнавання усного мовлення, перекладу, поліпшення результатів пошуку в Інтернеті та рекламного таргетингу.

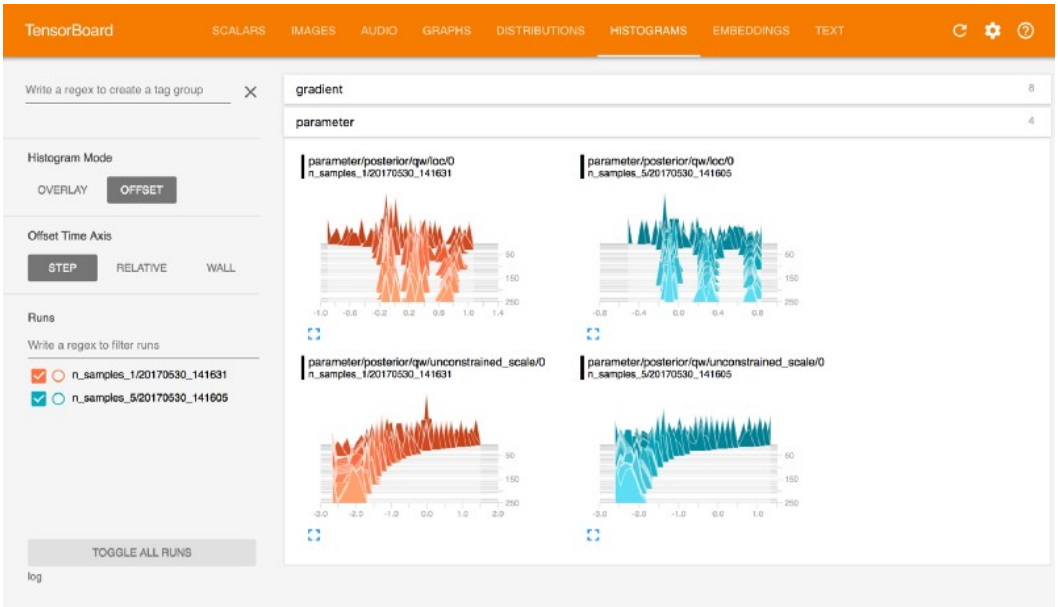

Рис. 2.3 Візуалізація характеристик моделі за допомогою TensorBoard

TensorFlow впроваджує різноманітні інструменти, які широко використовуються. Наприклад TensorBoard використовується для візуалізації інформації щодо системи, він демонструє процеси тренування та оцінки моделі, служить для швидкого оформлення нових розрахунків та тестів (рис. 2.3). Також TensorBoard має журнал подій, що дозволяє легко орієнтуватися

у процесах навчання та робити швидкі перевірки.

TensorFlow забезпечує прикладний програмний інтерфейс для Python, а також для C++, Haskell, Java та Go, а також доступний для 64-розрядних систем: Linux, macOS, Windows, та для мобільних обчислювальних платформ, включно з Android та iOS.

AlexNet (One Weird Trick paper) - Input 128x3x224x224

Library	Time (ms)	forward (ms)	backward (ms)
CuDNN-R3 (Torch)	96	32	64
Nervana (Neon)	101	32	69
CuDNN-R2 (Torch)	231	70	161
TensorFlow	326	96	230

Overfeat [fast] - Input 128x3x231x231

Library	Time (ms)	forward (ms)	backward (ms)
CuDNN-R3 (Torch)	326	113	213
fbfft (Torch)	342	114	227
CuDNN-R2 (Torch)	810	234	576
TensorFlow	1084	316	768

Рис. 2.4 Порівняння продуктивності

Дана бібліотека надає інтерактивний багатоплатформний інтерфейс програмування, який є масштабованим і набагато більш стабільним у порівнянні з іншими доступними бібліотеками машинного навчання, які наразі мають більш експериментальний характер.

Особливістю TensorFlow є модульність, що значно підвищує гнучкість системи та дозволяє використовувати різноманітні удосконалення або розширення. Такі розширення створюють як звичайні користувачі, так і працівники компанії, тому постійне оновлення — це ще одна перевага даного інструменту.

До недоліків бібліотеки потрібно віднести відносно низьку

продуктивність, у порівнянні із іншими бібліотеками, такими як Torch та Neon, порівняння продуктивності зображено на рис.2.4.

TensorFlow має широку архітектуру, до якої входять такі компоненти як:

- Servables — це центральні елементарні підрозділи TensorFlow Serving. TensorFlow Servables — це об'єкти, які клієнти використовують для виконання обчислень. Розмір Servables гнучкий та може включати в себе що завгодно, від однієї таблиці пошуку на основі однієї моделі до великої системи на кортежі моделей. Servables можуть бути будь-якого типу та інтерфейсу, що забезпечує гнучкість та майбутні покращення, такі як введення експериментальних API та асинхронних режимів роботи.
- Servable Versions — це компоненти, що здатні обробляти одну або кілька версій Servables протягом всього життєвого циклу одного екземпляра сервера. Це відкриває можливості для нових конфігурацій алгоритму, ваг та інших даних, які будуть завантажені з плином часу. Вони також дозволяють одночасно завантажувати більш ніж одну версію Servables, підтримуючи поступове розгортання та експериментування. Також клієнти можуть запитувати або останню версію, або певний ідентифікатор версії для конкретної моделі.
- Servable Streams — Послідовність версій придатна для сортування за збільшенням номеру версії.
- Models — являє собою модель у вигляді одного або декількох servables. Модель машинного навчання може включати в себе один або більше алгоритмів (включаючи визначені ваги). Servable може також служити часткою моделі, наприклад, велика таблиця пошуку може бути подана у розбитому на багато екземплярів стані.

- Loaders —це завантажувачі, керують життєвим циклом servable. API Loader забезпечує спільну інфраструктуру, незалежну від конкретних алгоритмів навчання, даних або випадків використання продуктів. Зокрема, завантажувачі стандартизують API для завантаження та вигрузку Servables.
- Sources — джерела є простими правилами та модулями, які знаходять і забезпечують servables. Кожне джерело надає нуль або більше доступних потоків. Для кожного servable потоку джерело поставляє один екземпляр Loader для кожної версії, яка може бути завантажена.
- Manager — менеджери Tensorflow опрацьовують повний життєвий цикл Servables, включаючи: завантаження Servables обслуговування Servables та їх вигрузку. Менеджери спостерігають за джерелами та відстежують всі версії. Менеджер намагається виконати запити джерел, але може відмовитися від завантаження потрібної версії. Менеджери також можуть відкласти “розвантаження”. Наприклад, менеджер може дочекатися вигрузки, доки не завершиться завантаження нової версії, на основі політики, яка гарантує, що принаймні одна версія завантажується в будь-який час.

AWS (Amazon Web Services) — широкий і різноманітний набір сервісів машинного навчання та AI для бізнесу. Надає можливість обрати один з попередньо підготовлених сервісів AI для роботи з комп'ютером, мовами, рекомендаціями і прогнозами. З допомогою сервісу Amazon SageMaker можна швидко створювати, навчати і розгортати масштабні моделі машинного навчання, або створювати користувацькі моделі з підтримкою всіх популярних платформ з відкритим кодом.

За даними AWS, їх перевстановлені оптимізовані алгоритми, здатні в 10 разів підвищити продуктивність при використанні сервісу Amazon SageMaker. Окрім цього існує магазин алгоритмів AWS, у якому можна

придбати різноманітні додаткові моделі та алгоритми для власного використання.

Amazon SageMaker автоматично налаштовує і оптимізує алгоритми TensorFlow, Apache MXNet, PyTorch, Chainer, Scikit-learn, SparkML, Horovod, Keras і Gluon. Поширені алгоритми машинного навчання інтегруються в систему і настроюються так, щоб забезпечити необхідні показники масштабованості, швидкості і точності. Для цього використовується понад 200 додаткових попередньо навчених моделей і алгоритмів, які доступні на AWS Marketplace. Крім того, можна розмістити в контейнері Docker і використовувати будь-який інший алгоритм або платформу.

AWS має велику кількість навчальних матеріалів на декількох мовах, а також готові Jupyter ноутбуки, в яких вже є блоки коду для первинної обробки даних та навчання моделей. Даний сервіс дозволяє створювати високоточні набори навчальних даних та економити до 70% на витратах на маркування даних за допомогою сервісу Amazon SageMaker Ground Truth.

Особливістю навчання моделі за допомогою AWS є можливість автоматично оптимізувати використання GPU на 75% (за даними з сайту AWS) за допомогою Amazon Elastic Inference, який дозволяє використовувати лише необхідні ресурси GPU. Також багато важливих функцій можна зробити в один клік, наприклад розгортання з використанням кластерів та автоматичного масштабування, розподілених в декількох зонах, щоб забезпечити високу продуктивність і доступність.

Сервіс надає можливість керованого A / B-тестування до п'яти різних моделей, що дає гнучкі можливості для проведення експериментів.

Попередньо навчені сервіси AI AWS надають готові інтелектуальні функції, які можна використовувати у власних додатках і робочих процесах. Сервіси AI можна легко інтегрувати з додатками для використання в стандартних сценаріях використання, наприклад для створення

персоналізованих рекомендацій, модернізації контакт-центру, підвищення безпеки та захищеності і збільшення активності клієнтів. Також є готові сценарії для аналізування відео та фото, документів, усного мовлення, транскрибування та перекладу. Для використання сервісів AI на AWS не потрібно досвід в сфері машинного навчання.

Сервіси AI AWS використовують такі компанії як MotorolaSolutions, NASA, Duolingo, Finra, vidmob тощо.

Проект *Apache Singa* — програмне забезпечення для Machine Learning, розробка якого буда ініційована групою DB System у 2014 році. Вони зосередилися на розподіленому глибокому навчанні шляхом розбиття моделі. Apache Singa забезпечує просту модель програмування. Крім того, працює через кластер машин. В основному дане ПО використовується в обробці природних мов (НЛП) і розпізнаванні зображень. Прототип Singa, прийнятий компанією Apache Incubator у березні 2015 року. Він забезпечує гнучку архітектуру для масштабованого розподіленого навчання, а також розширюється для роботи з широким спектром обладнання.

Apache Singa було розроблено за інтуїтивною моделлю програмування, заснованої на абстракції шарів. Підтримується широкий спектр популярних моделей глибокого навчання. Також ці моделі базуються на гнучкій архітектурі. При цьому, Singa запускає різні синхронні, асинхронні та гібридні рамки навчання. Програмний стек Singa складається з трьох основних компонентів: Core, IO і Model. Основний компонент стосується управління пам'яттю та операцій тензора, IO містить класи для читання і запису даних на диск і в мережу, а модель включає в себе структури даних і алгоритми для моделей машинного навчання.

Її основні особливості:

- Включає тензорну абстракцію для сильної підтримки більш просунутих моделей машинного навчання.

- Підтримує абстракцію апаратного забезпечення для роботи на різноманітних пристроях.
- Імпровізована прив’язка Python, а також містить більш глибокі моделі навчання, як VGG і ResNet
- Включає вдосконалені класи IO для читання, запису, кодування та декодування файлів і даних.

Caffe — це фреймворк для глибинного навчання, первинно розроблений Янці Дзя, як частина його докторської праці в Каліфорнійському університеті в Берклі, наразі у цього проекту багато учасників, і його розміщено на GitHub. Вона є відкритою, з ліцензією BSD та написана мовою C++ з інтерфейсом для Python.

Caffe підтримує багато різних типів архітектур глибинного навчання, орієнтованих на класифікацію та сегментування зображень. Вона підтримує конструкції згорткових та рекурентних нейронних мереж, а також довгої короткочасної пам’яті та повноз'єднаних нейронних мереж. Caffe підтримує прискорення на основі графічного процесора із застосуванням Cuda Nvidia та може обробляти близько 60 мільйонів зображень на день на графічному процесорі K40. Проте, не так просто перетворити гіперпараметри на нього програмно.

Torch — відкрита бібліотека для машинного навчання, система для наукових обчислень та мова сценаріїв на основі мови програмування Lua. Пропонує широкий спектр алгоритмів для глибинного машинного навчання і використовує мову сценаріїв LuaJIT та реалізацію мовою C в основі.

Основним пакетом бібліотеки Torch є однойменний компонент torch. Він забезпечує гнучкий N-вимірний масив, або тензор, який підтримує основні процедури для індексування, розшаровування, транспозиції, приведення типів, зміни розмірів, розподілення зберігання та клонування. Цей об'єкт використовується більшістю інших пакетів, і відтак є центральним

об'єктом бібліотеки. Тензор також підтримує математичні операції, такі як max, min, sum, статистичні розподіли, такі як рівномірний, нормальний та поліноміальний, та операції основних підпрограм лінійної алгебри, такі як скалярний добуток, матрично-векторне множення, матрично-матричне множення, матрично-векторний скалярний добуток та матричний скалярний добуток.

У складі бібліотеки також є пакет nn, який застосовують для побудови нейронних мереж. Його розділено на модульні об'єкти, які мають спільний інтерфейс Module. Модулі мають методи, які дозволяють їм виконувати пряме та зворотне поширення. Модулі можна з'єднувати, застосовуючи модулеві компонувальники, щоб створювати складні, підігнані під задачу графи. Простіші модулі, такі як Linear, Tanh і Max, складають модулі основних складових. Цей модульний інтерфейс забезпечує автоматичне диференціювання градієнтів першого порядку.

Torch використовує група дослідження штучного інтелекту компанії Facebook, IBM, Яндекс та Дослідницький інститут Ідіап. Torch було розширено для використання під Android і iOS. Його використовували для відтворення апаратних реалізацій потоків даних, подібних тим, що відбуваються у нейронних мережах.

Theano — це ще одна популярна бібліотека Python для чисельних обчислень і схожа на NumPy. Theano дозволяє ефективно визначити, оптимізувати та оцінити математичні вирази, що включають багатовимірні масиви, а також служить основою для багатьох інших бібліотек та фреймворків.

Theano може виступати в якості основи для точних наукових обчислень. Перевагою Theano, є те, що він користується можливостями GPU комп'ютера. Це дає змогу робити обчислення великих даних до 100 разів швидше, ніж при роботі на CPU. Швидкість Theano робить її особливо цінною для глибокого навчання та інших складних завдань.

Sources

1) "Speed/accuracy trade-offs for modern convolutional object detectors."

Huang J, Rathod V, Sun C, Zhu M, Korattikara A, Fathi A, Fischer I, Wojna Z,

Song Y, Guadarrama S, Murphy K, CVPR 2017