МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ФАКУЛЬТЕТ АВТОМАТИЗАЦІЇ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інформаційних технологій

ОПОРНИЙ КОНСПЕКТ ЛЕКЦІЙ

з дисципліни

«Штучний інтелект та гібридні мережі»

з освітньо-кваліфікаційного рівня «Магістр»
для спеціальності
«Комп’ютерні науки»

Київ 2024

Штучний інтелект та гібридні мережі. Конспект лекцій.

Долгополов С. Ю. Штучний інтелект та гібридні мережі : опорний конспект для здобувачів вищої освіти денної форми навчання. Київ : КНУБА, 2024. 78 с.

Автор: Долгополов Сергій Юрійович, асистент кафедри ІТ.

ЗМІСТ

ВСТУП

Метою навчальної дисципліни є набуття студентами комплексних знань та практичних навичок у розробці, оптимізації та застосуванні алгоритмів машинного навчання та глибокого навчання для розв'язування реальних задач у сфері штучного інтелекту. Дисципліна охоплює основи Data Science, роботу з даними, розгляд основних методів Machine Learning та Deep Learning.

Основними завданнями дисципліни є:

1. Ознайомлення з основами Data Science та штучного інтелекту. Вивчення теоретичних основ, методів збору, обробки та аналізу даних, ознайомлення з етичними та соціальними аспектами використання ШІ.
2. Практичне застосування методів машинного та глибокого навчання. Розробка та тренування моделей на реальних наборах даних, використання ансамблевих методів та оптимізація моделей для покращення їх ефективності.
3. Розвиток навичок роботи з передовими інструментами та платформами. Оволодіння інструментами розробки, такими як Google Colab, Kaggle, Jupyter Lab, та мовами програмування R та Python.

Структура дисципліни «Штучний інтелект та гібридні мережі». Дисципліна вивчається протягом 1-го семестру, відповідно складається з одного семестрового модулю: 1 семестр, що включає теоретичні лекції, лабораторні заняття та курсову роботу, спрямовані на засвоєння основних концепцій та методів штучного інтелекту, Data Science, Machine Learning і Deep Learning.

Даний конспект лекцій «Штучний інтелект та гібридні мережі» покликаний зорієнтувати здобувачів вищої освіти на оволодіння навчальним матеріалом семестрового модулю, надаючи глибоке розуміння теоретичних основ штучного інтелекту, а також практичні навички використання сучасних інструментів і методів для моделювання та розв'язання реальних задач. Матеріал модулю охоплює вступ в штучний інтелект, основи обробки та аналізу даних, а також розробку та оптимізацію моделей машинного та глибокого навчання, з метою підготовки студентів до самостійного вирішення задач в області штучного інтелекту.

МОДУЛЬ 1. ОСНОВИ DATA SCIENCE ТА ARTIFICIAL INTELLIGENCE

Лекція 1. Вступ в курс лекцій «Штучний інтелект та гібридні мережі».

Що ж таке штучний інтелект?

Це питання, яке протягом десятиліть хвилює уми науковців, філософів та фантастів. Якщо говорити просто, ШІ – здатність машини імітувати когнітивні функції людини, такі як навчання, розв'язання проблем, розпізнавання образів та прийняття рішень. Уявіть собі комп'ютер, який може не просто виконувати закладені в нього інструкції, а й аналізувати інформацію, робити висновки та адаптуватися до нових умов. Це і є та мета, до якої прагне ШІ.

Коротка історія ШІ

Концепція «штучного розуму» зародилася ще в давнину, коли люди мріяли про створення механічних істот, здатних думати та діяти самостійно. Однак, справжній науковий підхід до проблеми ШІ сформувався лише в середині XX століття. Знаковим моментом стала Дартмутська конференція 1956 року, де група вчених, серед яких були Джон Маккарті, Марвін Мінський, Клод Шеннон та Натаніель Рочестер, офіційно започаткувала дослідження в галузі штучного інтелекту.

Перші програми ШІ були здебільшого спрямовані на розв'язання логічних задач та ігри. Наприклад, програма «Логік-теоретик» могла доводити математичні теореми, а «Шахова програма» змагалася з людьми в шахи. Ці ранні успіхи породили хвилю оптимізму, однак незабаром стало зрозуміло, що створення справжнього «розуму» – значно складніше завдання, ніж здавалося на перший погляд.

Зими штучного інтелекту

Історія ШІ – це не лише тріумфи, а й періоди розчарувань, відомі як «зими ШІ». Надмірні очікування, невиправдані прогнози та технічні обмеження призводили до скорочення фінансування та зниження інтересу до досліджень. Однак, попри ці труднощі, наука про ШІ продовжувала розвиватися, і кожен новий виток прогресу приносив нові відкриття та можливості.

Сучасний ренесанс ШІ

Сьогодні ми є свідками справжнього ренесансу ШІ. Цьому сприяли кілька факторів:

- **Доступність обчислювальних потужностей.** Сучасні комп'ютери та графічні процесори здатні обробляти величезні обсяги даних, необхідні для навчання складних моделей ШІ.
- **Великі дані (Big Data).** Інтернет та цифрові технології породили експоненціальне зростання даних, які слугують «паливом» для алгоритмів машинного навчання.
- **Прорив у галузі глибокого навчання (Deep Learning).** Багатошарові нейронні мережі, здатні навчатися на складних даних, продемонстрували вражаючі результати в різноманітних задачах, від розпізнавання зображень до перекладу мов.

Інтелектуальний аналіз даних

В основі сучасного ШІ лежить інтелектуальний аналіз даних (Data Mining). Сучасний світ генерує колосальні об'єми даних щосекунди: від транзакцій у супермаркетах до постів у соціальних мережах. Інтелектуальний аналіз даних – мистецтво видобувати з цієї лавини інформації цінні знання, закономірності та інсайти, які допомагають приймати обґрунтовані рішення.

Уявіть, що ви власник інтернет-магазину. Аналізуючи дані про покупки клієнтів, їхні перегляди та вподобання, ви можете виявити приховані тенденції, наприклад, які товари часто купують разом, або які групи клієнтів схильні до імпульсивних покупок. Ця інформація дозволить вам оптимізувати асортимент, налаштувати персоналізовані рекомендації та, зрештою, збільшити прибуток.

Етика та соціальні аспекти ШІ

Розвиток ШІ ставить перед людством не лише технічні, а й етичні та соціальні питання. Чи може машина бути «справедливою»? Як запобігти упередженості алгоритмів? Хто несе відповідальність за помилки, допущені ШІ? Ці питання стають дедалі актуальнішими, адже ШІ все глибше проникає в наше життя.

Наприклад, алгоритми ШІ вже сьогодні використовуються для прийняття рішень про видачу кредитів, найм на роботу та навіть у судочинстві. Якщо ці алгоритми навчаються на упереджених даних, вони можуть відтворювати та посилювати існуючу нерівність. Уявіть собі систему, яка систематично відмовляє в кредитах жінкам або представникам певних етнічних груп лише тому, що історично склалося так, що вони мали менше можливостей для отримання кредитів.

Інший аспект – вплив ШІ на ринок праці. Автоматизація та роботизація можуть призвести до зникнення цілих професій, що може спричинити соціальну напругу та нерівність. Тому важливо вже сьогодні думати про те, як адаптувати суспільство до цих змін, наприклад, через перекваліфікацію та створення нових робочих місць.

Інструментарій ШІ

Тепер, коли ми заклали теоретичний фундамент, поговоримо про практичні інструменти, які ми будемо використовувати в нашому курсі. Розробка моделей ШІ – це, перш за все, програмування. І тут у нас є два основних гравці: **Python** та **R**.

Python – універсальна мова програмування, яка завоювала світ ШІ завдяки своїй простоті, читабельності та багатій екосистемі бібліотек. NumPy, Pandas, Scikit-learn, TensorFlow, Keras, PyTorch – ці назви стануть для вас звичними, адже це потужні інструменти для обробки даних, машинного та глибокого навчання.

R – мова, створена спеціально для статистичного аналізу та візуалізації даних. Вона має потужні можливості для роботи з даними, включаючи спеціалізовані пакети для статистичного моделювання та машинного навчання. R особливо популярний серед дослідників та аналітиків даних.

Середовища розробки

Але де ж писати код? Тут на сцену виходять середовища розробки. Ми розглянемо три популярні платформи:

- **Google Colab.** Хмарне середовище, яке надає безкоштовний доступ до обчислювальних ресурсів, включаючи GPU. Ідеально підходить для глибокого навчання та роботи з великими даними. Не потрібно нічого встановлювати – все, що вам потрібно, це обліковий запис Google.
- **Kaggle.** Платформа для змагань з машинного навчання, яка також надає середовище для розробки з безкоштовним доступом до GPU та TPU. Kaggle – це не тільки інструмент, а й спільнота, де можна вчитися у досвідчених фахівців та ділитися своїми знаннями.
- **Jupyter Lab.** Інтерактивне середовище розробки, яке дозволяє поєднувати код, текст, зображення та результати обчислень в одному документі – Jupyter Notebook. Це ідеальний інструмент для дослідницької роботи, прототипування та створення звітів.

Контрольні запитання.

1. Які основні задачі курсу «Штучний інтелект та гібридні мережі» та як вони пов’язані з іншими освітніми компонентами? Наведіть приклади міждисциплінарних зв’язків.
2. Дайте визначення штучного інтелекту (ШІ). Чим відрізняється «сильний» ШІ від «слабкого»? Які сучасні досягнення в галузі ШІ ви вважаєте найбільш вражаючими і чому?
3. Що таке інтелектуальний аналіз даних (Data Mining)? Які його основні завдання та методи? Наведіть приклад практичного застосування Data Mining з вашої галузі інтересів.
4. Які етичні та соціальні аспекти використання ШІ викликають найбільше занепокоєння? Як можна мінімізувати потенційні ризики, пов’язані з впровадженням ШІ?
5. Порівняйте можливості та обмеження середовищ розробки Google Colab, Kaggle та Jupyter Lab. Яку мову програмування, R чи Python, ви б обрали для розробки моделей ШІ і чому?

Лекція 2. Data Science та підготовка даних.

Data Science: наука про дані

Уявіть собі величезну бібліотеку, наповнену книгами, написаними невідомою мовою. Ви не можете прочитати жодного слова, але інтуїтивно відчуваєте, що в цих книгах приховані безцінні знання. Data Science – це і є та сама наука, яка дає нам ключ до розуміння мови даних. Вона поєднує в собі методи з математичної статистики, машинного навчання та програмування, щоб видобувати з сирих даних корисну інформацію, виявляти закономірності та робити прогнози.

Фахівці з Data Science, яких часто називають «дата-сайєнтистами», схожі на сучасних алхіміків. Вони перетворюють терабайти незрозумілої інформації на золото інсайтів, які допомагають компаніям приймати кращі рішення, оптимізувати процеси та створювати нові продукти. Наприклад, Netflix використовує Data Science, щоб рекомендувати фільми, які вам сподобаються, Amazon – щоб передбачати попит на товари, а банки – щоб виявляти шахрайські транзакції.

Збір даних: де шукати скарби?

Перш ніж аналізувати дані, їх потрібно зібрати. І тут перед нами відкривається цілий всесвіт можливостей. Дані можуть надходити з найрізноманітніших джерел:

- **Внутрішні бази даних компаній:** інформація про клієнтів, транзакції, складські запаси тощо.
- **Інтернет:** вебсайти, соціальні мережі, форуми – справжня скарбниця інформації про поведінку та вподобання людей.
- **Сенсори та пристрої Інтернету речей (IoT):** дані з датчиків температури, вологості, руху та інших параметрів навколишнього середовища.
- **Відкриті дані:** урядові портали, наукові публікації, статистичні збірники – безкоштовні джерела інформації, доступні кожному.

Збір даних може здійснюватися різними способами: від ручного введення до автоматичного парсингу вебсторінок. Важливо забезпечити повноту, достовірність та релевантність даних, адже від їх якості залежить успіх всього подальшого аналізу.

Структуровані дані: коли все розкладено по поличках

Зібрані дані можуть бути представлені в різних форматах, але для машинного навчання найзручніше працювати зі структурованими даними. Такі дані організовані у вигляді таблиць, де

кожен рядок представляє окремий об’єкт (наприклад, клієнта, товар або транзакцію), а кожен стовпець – його атрибут (наприклад, вік, ціну або дату).

Уявіть собі таблицю з інформацією про клієнтів інтернет-магазину. Кожен рядок – це окремий клієнт, а стовпці містять такі дані, як ім’я, вік, стать, місто проживання, сума покупок тощо. Така таблиця є прикладом структурованих даних. Кожен елемент у ній має чітко визначене місце та тип.

Структуровані дані поділяються на різні типи, залежно від характеру інформації:

- **Числові:** кількісні дані, які можна виміряти (наприклад, вік, зріст, вартість).
- **Категоріальні:** дані, які представляють належність об’єкта до певної категорії (наприклад, стать, колір, тип товару).
- **Бінарні:** окремий випадок категоріальних даних, коли є лише дві можливі категорії (наприклад, «так»/»ні», «правда»/»брехня»).
- **Дата і час:** інформація про часові проміжки або конкретні моменти часу.

Розуміння типів даних критично важливе для вибору правильних методів аналізу. Наприклад, для числових даних можна розраховувати середнє значення, а для категоріальних – частоту зустрічальності кожної категорії.

Оцінка центрального положення

Уявіть, що ви хочете зрозуміти, який середній вік клієнтів вашого інтернет-магазину. Або яка середня зарплата в IT-компанії. Або який найпопулярніший товар у вашому асортименті. У всіх цих випадках вам потрібно знайти міру центрального положення – значення, яке найкраще представляє «типовий» елемент набору даних.

Найпоширеніші міри центрального положення:

- **Середнє арифметичне (Mean):** сума всіх значень, поділена на їх кількість. Чутливе до викидів – екстремально великих або малих значень. Наприклад, якщо в компанії з 9 співробітниками, які заробляють по 50000 гривень, з’являється директор із зарплатою 1000000 гривень, середнє значення різко зросте, хоча зарплата більшості співробітників залишиться незмінною.
- **Медіана (Median):** значення, яке знаходиться посередині впорядкованого набору даних. Половина значень менша за медіану, а інша половина – більша. Не чутлива до викидів. У прикладі з компанією медіана залишиться близькою до 50000 гривень, навіть після появи високооплачуваного директора.

- **Мода (Mode):** значення, яке зустрічається найчастіше. Може бути декілька мод (мультимодальний розподіл). Корисна для категоріальних даних. Наприклад, якщо в магазині найчастіше купують футболки розміру «М», то «М» – це мода.

Вибір міри центрального положення залежить від типу даних та мети аналізу. Медіана краще підходить для даних з викидами, а мода – для категоріальних даних.

Аналіз викидів: шукаємо аномалії

Викиди – значення, які значно відрізняються від решти даних. Вони можуть бути результатом помилок вимірювання, незвичайних подій або просто рідкісних, але реальних випадків. Наприклад, у наборі даних про вік клієнтів викидом може бути значення 120 років, яке, ймовірно, є помилкою. Або, наприклад, середній зріст дорослих жінок в Україні становить 166 см, але ми можемо знайти жінку зростом 205 см. Це буде викидом, але цілком реальною людиною.

Викиди можуть спотворювати результати аналізу, тому їх важливо виявляти та обробляти. Для виявлення викидів використовують різні методи, наприклад, візуалізацію даних за допомогою діаграм розмаху («ящик з вусами») або статистичні тести, такі як тест Граббса.

Після виявлення викидів з ними потрібно щось робити. І тут немає універсального рішення. Іноді викиди видаляють, іноді замінюють на інші значення (наприклад, на медіану), а іноді залишають без змін, якщо вони є важливими для розуміння даних.

Варіативність: міра розкиду даних

Центральне положення дає нам уявлення про «типове» значення, але не говорить про те, наскільки дані розкидані навколо цього значення. Для вимірювання розкиду даних використовують міри варіативності.

- **Розмах (Range):** різниця між максимальним та мінімальним значенням. Дуже чутливий до викидів.
- **Дисперсія (Variance):** середня сума квадратів відхилень кожного значення від середнього арифметичного. Чим більша дисперсія, тим сильніше дані розкидані.
- **Стандартне відхилення (Standard Deviation):** квадратний корінь з дисперсії. Має ту ж розмірність, що й вихідні дані, тому його легше інтерпретувати. Приблизно 68% даних лежать в межах одного стандартного відхилення від середнього, 95% – в межах двох стандартних відхилень, і 99.7% – в межах трьох стандартних відхилень (правило трьох сигм).

Уявіть, що ви порівнюєте дві групи студентів за результатами тесту. Середній бал в обох групах однаковий, але в першій групі стандартне відхилення набагато менше. Це означає, що результати студентів у першій групі більш однорідні, а в другій групі є як сильні, так і слабкі студенти.

Часові ряди

Особливий тип даних – часові ряди. Вони представляють собою послідовність значень, виміряних через рівні проміжки часу. Наприклад, щоденні ціни на акції, щомісячні продажі товарів, погодинна температура повітря.

Аналіз часових рядів дозволяє виявляти тренди, сезонні коливання та циклічні патерни. Наприклад, можна передбачити, коли буде пік продажів морозива (влітку) або коли зросте попит на теплий одяг (взимку).

Для аналізу часових рядів використовуються спеціальні методи, такі як ковзне середнє, експоненційне згладжування та авторегресійні моделі.

Очищення даних

Перш ніж аналізувати дані, їх потрібно очистити від помилок, пропусків та неузгодженостей. Цей процес називається очищенням даних (Data Cleaning) і є одним з найважливіших етапів підготовки даних.

Уявіть, що ви отримали дані з опитування, де в деяких анкетах не заповнено поле «вік», а в інших замість віку вказано «не знаю». Або, скажімо, є два однакові записи з різними даними. Такі дані непридатні для аналізу, їх потрібно очистити.

Очищення даних включає в себе:

- **Обробку пропусків:** пропущені значення можна видалити, замінити на середнє, медіану або моду, або заповнити за допомогою алгоритмів машинного навчання.
- **Виправлення помилок:** неправильні значення можна виправити вручну або за допомогою автоматичних правил.
- **Усунення дублікатів:** однакові записи потрібно видалити або об’єднати.
- **Приведення даних до єдиного формату:** наприклад, перетворення всіх дат в один формат, а всіх текстових значень – в нижній регістр.

Ретельне очищення даних – запорука якісного аналізу. «Сміття на вході – сміття на виході» – це золоте правило Data Science.

Візуалізація даних

Людина краще сприймає інформацію візуально, тому візуалізація даних – потужний інструмент аналізу та комунікації. Графіки та діаграми допомагають швидко зрозуміти основні закономірності, виявити тренди та аномалії.

Для візуалізації даних використовуються різні бібліотеки, такі як:

- **Pandas:** окрім функцій для обробки даних, Pandas має базові можливості для побудови графіків, наприклад, гістограм, діаграм розсіювання та лінійних графіків.
- **Matplotlib:** низькорівнева бібліотека, яка дає повний контроль над усіма елементами графіка. Дозволяє створювати найрізноманітніші візуалізації, від простих лінійних графіків до складних 3D-діаграм.
- **Seaborn:** високорівнева бібліотека, побудована на основі Matplotlib. Пропонує зручний інтерфейс для створення статистичних графіків, таких як розподіли, парні діаграми та теплові карти.

Вибір інструменту залежить від типу даних та мети візуалізації. Наприклад, для відображення розподілу однієї змінної підійде гістограма, для порівняння двох груп – діаграма розмаху, а для вивчення взаємозв'язку між двома змінними – діаграма розсіювання.

Нормалізація та стандартизація

Часто дані мають різний масштаб. Наприклад, вік може змінюватися від 0 до 100, а дохід – від 0 до кількох мільйонів. Якщо не привести дані до єдиного масштабу, алгоритми машинного навчання можуть надавати більшого значення змінним з більшим діапазоном значень, що призведе до некоректних результатів.

Для вирішення цієї проблеми використовують нормалізацію та стандартизацію даних.

- **Нормалізація (Min-Max Scaling):** лінійне перетворення даних, яке приводить їх значення до діапазону від 0 до 1.

$$x_{scaled} = \frac{x - min(x)}{max(x) - min(x)}$$

- **Стандартизація (Z-score):** перетворення даних таким чином, що середнє значення стає рівним 0, а стандартне відхилення – 1.

$$x_{scaled} = \frac{x - mean(x)}{std(x)}$$

Вибір методу залежить від конкретної задачі. Нормалізація підходить, коли важливо зберегти відносні відстані між об’єктами, а стандартизація – коли дані мають нормальний розподіл.

Контрольні запитання

1. Опишіть основні етапи процесу збору та обробки даних. Які труднощі можуть виникнути на кожному етапі і як їх можна подолати?
2. Що таке структуровані дані? Наведіть приклади різних типів структурованих даних. Як тип даних впливає на вибір методів їхнього аналізу?
3. Поясніть різницю між середнім значенням, медіаною та модою. У яких випадках доцільно використовувати кожну з цих мір центрального положення? Наведіть приклади з реального життя.
4. Що таке викиди в даних і як їх можна виявити? Опишіть методи обробки викидів. Який вплив можуть мати викиди на результати аналізу даних?
5. Що таке нормалізація та стандартизація даних? У чому різниця між цими методами? Наведіть приклади методів нормалізації (Min-Max Scaling, Z-score) та поясніть, коли доцільно використовувати кожен із них?

Лекція 3. Основи Machine Learning та Deep Learning.

Machine Learning

Уявіть, що ви намагаєтеся навчити собаку команді «сидіти». Ви показуєте їй, як це робити, даєте ласощі за правильне виконання і повторюєте вправу знову і знову. З часом собака починає розуміти, чого ви від неї хочете, і виконує команду без підказки. Це і є навчання – процес, в результаті якого організм набуває нових знань і навичок на основі досвіду.

Machine Learning – область штучного інтелекту, яка наділяє комп'ютери здатністю навчатися на даних, не будучи явно запрограмованими. Замість того, щоб вручну прописувати правила для кожної можливої ситуації, ми даємо машині дані та алгоритм, який дозволяє їй самостійно виявляти закономірності, будувати моделі та робити прогнози.

Ідея машинного навчання не нова. Ще в 1959 році Артур Семюель, один з піонерів ШІ, визначив ML як «галузь досліджень, яка дає комп'ютерам здатність навчатися, не будучи явно запрограмованими». Однак справжній розквіт ML почався лише в останні десятиліття, з появою великих даних і потужних обчислювальних ресурсів.

Три кити машинного навчання: навчання з учителем, без учителя та з підкріпленням

Залежно від типу доступних даних і поставленого завдання, машинне навчання поділяється на три основні типи:

1. **Навчання з учителем (Supervised Learning):** найпоширеніший тип ML. Алгоритм навчається на «розмічених» даних, де кожному об'єкту відповідає правильна відповідь («мітка»). Наприклад, ми можемо навчити алгоритм розпізнавати котів на фотографіях, показавши йому тисячі зображень котів і собак, де кожне зображення підписано «кіт» або «собака». Мета алгоритму – навчитися знаходити закономірності, які відрізняють котів від собак, щоб потім правильно класифікувати нові зображення.

Задачі навчання з учителем поділяються на два класи:

- **Класифікація (Classification):** передбачення категорії об'єкта. Наприклад, визначення виду тварини на фотографії, фільтрація спаму в електронній пошті або діагностика захворювання за медичними показниками.
- **Регресія (Regression):** передбачення числового значення. Наприклад, прогнозування ціни на нерухомість, попиту на товари або температури повітря на завтра.

Приклади алгоритмів навчання з учителем: лінійна регресія, логістична регресія, метод опорних векторів, дерева рішень, випадковий ліс.

2. **Навчання без учителя (Unsupervised Learning):** алгоритм навчається на «нерозмічених» даних, де немає правильних відповідей. Мета алгоритму – самостійно знайти структуру в даних, виділити групи схожих об'єктів або виявити аномалії. Уявіть, що вам дали коробку з різнокольоровими кубиками, кульками та пірамідками, і попросили розсортувати їх, не знаючи, скільки груп і за якою ознакою сортувати. Алгоритм навчання без учителя зробить це самостійно, ґрунтуючись на схожості об'єктів за формою, розміром, кольором тощо.

Основні задачі навчання без учителя:

- **Кластеризація (Clustering):** розбиття даних на групи (кластери) схожих об'єктів. Наприклад, сегментація клієнтів за купівельною поведінкою, групування новинних статей за темами або виявлення спільнот у соціальних мережах.
- **Зниження розмірності (Dimensionality Reduction):** зменшення кількості ознак, що описують дані, без втрати важливої інформації. Використовується для спрощення моделей, візуалізації даних та виявлення прихованих факторів. Наприклад, стиснення зображень, виділення основних тем у тексті або зменшення кількості змінних у медичних дослідженнях.
- **Пошук асоціативних правил (Association Rule Learning):** виявлення стійких взаємозв'язків між об'єктами. Наприклад, аналіз покупок у супермаркеті («клієнти, які купують пиво, часто купують і чіпси»), рекомендація товарів в інтернет-магазинах або виявлення взаємозв'язків між симптомами захворювань.

Приклади алгоритмів навчання без учителя: метод k-середніх, ієрархічна кластеризація, метод головних компонент, алгоритм Apriori.

3. **Навчання з підкріпленням (Reinforcement Learning):** алгоритм навчається, взаємодіючи з середовищем і отримуючи винагороду або покарання за свої дії. Мета алгоритму – навчитися обирати таку стратегію поведінки, яка максимізує сумарну винагороду. Це схоже на дресирування тварин, де правильні дії заохочуються, а неправильні – караються.

Навчання з підкріпленням використовується в задачах, де потрібно приймати послідовні рішення, наприклад, в іграх, робототехніці та управлінні ресурсами. Наприклад, алгоритм може навчитися грати в шахи, керувати роботом-пилососом або оптимізувати роботу системи охолодження в дата-центрі.

Основні поняття навчання з підкріпленням:

- **Агент (Agent):** той, хто навчається і приймає рішення.
- **Середовище (Environment):** те, з чим взаємодіє агент.
- **Стан (State):** поточна ситуація, в якій знаходиться агент.
- **Дія (Action):** те, що робить агент.
- **Винагорода (Reward):** зворотний зв'язок від середовища, який оцінює дії агента.

Приклади алгоритмів навчання з підкріпленням: Q-learning, SARSA, Deep Q-Network.

Тренувальні та тестувальні дані

Щоб навчити модель машинного навчання, нам потрібні дані. Але як перевірити, чи дійсно модель навчилася узагальнювати інформацію, а не просто запам'ятала навчальні приклади? Для цього дані поділяють на дві частини: тренувальний набір (Training Set) і тестувальний набір (Test Set).

- **Тренувальний набір:** дані, на яких навчається модель. Алгоритм аналізує ці дані, виявляє закономірності та налаштовує свої параметри.
- **Тестувальний набір:** дані, які модель не бачила під час навчання. Використовуються для оцінки якості моделі та її здатності узагальнювати інформацію на нових даних.

Уявіть, що ви готуєтеся до іспиту. Тренувальний набір – це конспекти лекцій і приклади завдань, які ви вивчаєте. Тестувальний набір – це сам іспит, де вам доведеться вирішувати нові завдання, яких ви раніше не бачили. Якщо ви просто завчите конспекти, але не зрозумієте матеріал, ви, швидше за все, провалите іспит. Так само і модель машинного навчання: якщо вона просто запам'ятає тренувальні дані, але не навчиться узагальнювати, вона покаже погані результати на тестувальних даних.

Методи розділення даних

Існує кілька методів розділення даних на тренувальний і тестувальний набори:

- **Випадкове розділення (Random Split):** дані випадковим чином діляться на дві частини в заданому співвідношенні, наприклад, 80% на навчання і 20% на тестування. Найпростіший і найпоширеніший метод.
- **Стратифіковане розділення (Stratified Split):** використовується, коли потрібно зберегти пропорції класів у тренувальному і тестувальному наборах. Наприклад, якщо в даних 90% об'єктів належать до класу «А» і 10% – до класу «Б», то в тренувальному і

тестувальному наборах буде збережено таке ж співвідношення. Це важливо для задач класифікації з незбалансованими даними.

Вибір методу розділення залежить від конкретної задачі та характеристик даних. Важливо, щоб тестувальний набір був репрезентативним, тобто відображав основні властивості генеральної сукупності даних.

Deep Learning

Deep Learning – це підмножина машинного навчання, яка використовує глибокі нейронні мережі для навчання на даних. Глибокі – означає, що мережа має багато шарів, кожен з яких виконує свою частину роботи з обробки інформації. Уявіть собі конвеєр, де на кожному етапі відбувається певна операція з обробки сировини, поки на виході не вийде готовий продукт. Так само і в глибоких нейронних мережах: кожен шар витягує з даних все більш складні ознаки, поки на виході не вийде результат.

Концепції глибокого навчання

Ідея нейронних мереж була натхненна роботою людського мозку, який складається з мільярдів нейронів, з'єднаних між собою синапсами. Кожен нейрон отримує сигнали від інших нейронів, обробляє їх і передає далі. Навчання в мозку відбувається шляхом зміцнення або ослаблення зв'язків між нейронами.

Штучна нейронна мережа – це математична модель, яка імітує цю структуру. Вона складається з вузлів (штучних нейронів), з'єднаних між собою зв'язками (синапсами). Кожен зв'язок має вагу, яка визначає силу його впливу на наступний нейрон.

Архітектури глибоких нейронних мереж

Існує багато різних архітектур глибоких нейронних мереж, кожна з яких найкраще підходить для вирішення певного типу завдань:

- **Перцептрон (Perceptron):** найпростіша нейронна мережа, яка складається з одного шару нейронів. Використовується для вирішення задач лінійної класифікації.
- **Багатошаровий перцептрон (Multilayer Perceptron, MLP):** складається з декількох шарів нейронів, з'єднаних між собою. Може вирішувати більш складні задачі, ніж одношаровий перцептрон.
- **Згорткові нейронні мережі (Convolutional Neural Networks, CNN):** спеціалізуються на обробці зображень. Використовують операцію згортки для виділення локальних ознак, таких як краї, кути та текстури. Згорткові шари чергуються з шарами субдискретизації

(pooling), які зменшують розмірність даних і роблять мережу більш стійкою до невеликих зрушень і поворотів зображення. CNN показали вражаючі результати в задачах розпізнавання об'єктів, класифікації зображень і сегментації зображень.

- **Рекурентні нейронні мережі (Recurrent Neural Networks, RNN):** призначені для обробки послідовностей даних, таких як текст, аудіо та відео. Мають зворотні зв'язки, які дозволяють їм зберігати інформацію про попередні вхідні дані. Однак, стандартні RNN погано навчаються на довгих послідовностях через проблему зникаючого градієнта.
    - **Довга короткострокова пам'ять (Long Short-Term Memory, LSTM):** спеціальний тип RNN, розроблений для вирішення проблеми зникаючого градієнта. LSTM мають комірки пам'яті, які можуть зберігати інформацію протягом тривалого часу, і вентилі, які контролюють потік інформації.
    - **Двонаправлена RNN (Bidirectional RNN, BRNN):** обробляють послідовність даних в обох напрямках – від початку до кінця і від кінця до початку. Це дозволяє їм враховувати контекст з обох сторін кожного елемента послідовності.

Практичне застосування Deep Learning

Глибоке навчання зробило революцію в багатьох областях, де потрібна обробка складних даних:

- **Обробка природної мови (Natural Language Processing, NLP):** машинний переклад, розпізнавання мови, генерація тексту, аналіз тональності, чат-боти. Наприклад, Google Translate використовує глибокі нейронні мережі для перекладу між мовами, а Siri і Alexa – для розпізнавання голосових команд.
- **Комп'ютерний зір (Computer Vision):** розпізнавання об'єктів на зображеннях і відео, класифікація зображень, сегментація зображень, детектування облич, розпізнавання емоцій. Наприклад, безпілотні автомобілі використовують комп'ютерний зір для орієнтування в просторі, а системи безпеки – для розпізнавання облич.
- **Ігри:** навчання агентів для гри в складні ігри, такі як Go, Dota 2 і StarCraft II. Наприклад, AlphaGo від DeepMind перемогла чемпіона світу з Go, використовуючи глибоке навчання з підкріпленням.
- **Медицина:** діагностика захворювань за медичними зображеннями, прогнозування ризику розвитку захворювань, розробка нових ліків. Наприклад, глибокі нейронні мережі можуть виявляти рак на рентгенівських знімках з точністю, що перевищує точність досвідчених лікарів.

- **Фінанси:** прогнозування курсів акцій, виявлення шахрайства, оцінка кредитного ризику. Наприклад, банки використовують глибоке навчання для виявлення аномальних транзакцій, які можуть свідчити про шахрайство.

Типові виклики Deep Learning

Незважаючи на вражаючі успіхи, глибоке навчання стикається з низкою викликів:

1. **Перенавчання (Overfitting):** модель занадто добре підлаштовується під тренувальні дані і погано узагальнює на нові дані. Це схоже на студента, який завчив відповіді на всі питання в підручнику, але не може відповісти на питання, сформульоване інакше.

Боротися з перенавчанням можна за допомогою:

- **Регуляризації:** додавання штрафів за занадто складні моделі.
- **Збільшення обсягу даних:** чим більше даних, тим краще модель узагальнює.
- **Спрощення моделі:** зменшення кількості шарів або нейронів у мережі.
- **Dropout:** випадкове «вимкнення» деяких нейронів під час навчання, що змушує мережу вчитися розподіляти інформацію між різними нейронами.

2. **Недонавчання (Underfitting):** модель занадто проста і не може виявити складні закономірності в даних. Це схоже на студента, який вивчив лише поверхнево матеріал і не може відповісти навіть на прості питання.

Боротися з недонавчанням можна за допомогою:

- **Збільшення складності моделі:** додавання більшої кількості шарів або нейронів.
- **Використання більш складних функцій активації.**
- **Збільшення часу навчання.**

3. **Вибір архітектури моделі:** не існує універсальної архітектури, яка підходила б для всіх завдань. Вибір архітектури – це мистецтво, яке вимагає експериментального досвіду та творчості.

Контрольні запитання

1. Дайте визначення машинного навчання (Machine Learning). Чим воно відрізняється від традиційного програмування? Наведіть приклади задач, які можна вирішити за допомогою ML.
2. Опишіть основні типи машинного навчання: навчання з учителем, без учителя, з підкріпленням. Наведіть приклади задач для кожного типу навчання.
3. Поясніть, що таке тренувальний та тестувальний набори даних і для чого вони використовуються. Опишіть методи розділення даних на тренувальний і тестувальний набори (випадкове розділення, стратифіковане розділення).
4. Що таке глибоке навчання (Deep Learning)? Чим воно відрізняється від класичного машинного навчання? Які переваги та недоліки глибокого навчання?
5. Опишіть основні архітектури глибоких нейронних мереж: перцептрони, конволюційні та рекурентні нейронні мережі. Які задачі найкраще вирішуються за допомогою кожної з цих архітектур?

МОДУЛЬ 2. ПІДГОТОВКА ТА ОПТИМІЗАЦІЯ МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ

Лекція 4. Архітектура та проєктування нейронних мереж

Від перцептрона до складних систем: еволюція нейромереж

Почнемо з найпростішого – одношарового перцептрона. Його можна порівняти з фундаментом, на якому зводяться складніші конструкції. Розроблений Френком Розенблаттом у 1958 році, одношаровий перцептрон став першою реалізацією штучної нейронної мережі. Він складається з одного шару нейронів, кожен з яких отримує вхідні сигнали, зважує їх, підсумовує та застосовує функцію активації, видаючи результат.

Попри простоту, одношаровий перцептрон має обмежені можливості. Він здатний вирішувати лише задачі лінійної класифікації, тобто такі, де дані можна розділити на класи прямою лінією. Наприклад, він може навчитися розрізняти червоні та зелені яблука за їхнім кольором, але не зможе класифікувати об’єкти, які неможливо розділити лінійно.

Багатошаровий перцептрон (MLP)

Щоб подолати обмеження одношарового перцептрона, дослідники додали до нього приховані шари нейронів, розташовані між вхідним і вихідним шарами. Так народився багатошаровий перцептрон (Multilayer Perceptron, MLP). Кожен нейрон прихованого шару отримує сигнали від нейронів попереднього шару, обробляє їх і передає далі. Завдяки такій багатошаровій структурі, MLP може апроксимувати будь-яку неперервну функцію, що робить його універсальним інструментом для вирішення широкого кола завдань.

MLP – це повністю зв’язана мережа (Fully Connected, FC). Кожен нейрон одного шару з’єднаний з кожним нейроном наступного шару. Збільшення кількості шарів і нейронів у кожному шарі дозволяє MLP навчатися все складнішим залежностям у даних. Однак зі зростанням складності мережі зростає і ризик перенавчання, коли модель занадто добре підлаштовується під тренувальні дані і погано узагальнює на нові.

Конволюційні нейронні мережі (CNN)

Коли мова заходить про обробку зображень, на сцену виходять конволюційні нейронні мережі (Convolutional Neural Networks, CNN). Їхня архітектура натхненна будовою зорової кори

головного мозку тварин. CNN використовують операцію згортки для виділення локальних ознак зображення, таких як краї, кути та текстури.

Уявіть, що ви дивитеся на фотографію через маленьке віконце, яке переміщуєте по всьому зображенню. Це віконце – ядро згортки (фільтр), яке є набором ваг. Згортка полягає в тому, що фільтр ковзає по зображенню, і для кожного положення виконується поелементне множення значень пікселів зображення на ваги фільтра, а потім результати підсумовуються. Результатом є карта ознак, яка показує, наскільки сильно певна ознака виражена в різних частинах зображення.

Згорткові шари в CNN чергуються з шарами субдискретизації (пулінгу), які зменшують розмірність даних, узагальнюючи інформацію з сусідніх пікселів. Найпоширеніший тип пулінгу – max-pooling, який вибирає максимальне значення з невеликої області. Пулінг робить мережу більш стійкою до невеликих зрушень і поворотів зображення.

CNN чудово зарекомендували себе в задачах розпізнавання об’єктів, класифікації та сегментації зображень. Наприклад, вони використовуються в системах розпізнавання облич, безпілотних автомобілях, медичній діагностиці та багатьох інших областях.

Рекурентні нейронні мережі (RNN)

Якщо CNN спеціалізуються на обробці зображень, то рекурентні нейронні мережі (Recurrent Neural Networks, RNN) – на обробці послідовностей даних, таких як текст, аудіо та відео. Їхня ключова особливість – наявність зворотних зв’язків, які дозволяють нейронам зберігати інформацію про попередні вхідні дані.

Уявіть, що ви читаєте речення. Щоб зрозуміти значення кожного слова, вам потрібно пам’ятати попередні слова. Так само і RNN: кожен нейрон отримує не тільки поточний вхідний сигнал, але й інформацію про попередній стан мережі. Це дозволяє RNN враховувати контекст і навчатися залежностей між елементами послідовності.

Однак стандартні RNN мають обмеження – вони погано навчаються на довгих послідовностях через проблему зникаючого або вибухового градієнта. Суть проблеми полягає в тому, що при зворотному поширенні помилки градієнт може або занадто швидко зменшуватися (зникати), або занадто швидко збільшуватися (вибухати), що унеможливлює ефективне навчання мережі.

Для вирішення цієї проблеми були розроблені спеціальні типи RNN, такі як:

- **Довга короткострокова пам’ять (Long Short-Term Memory, LSTM):** LSTM мають спеціальні комірки пам’яті, які можуть зберігати інформацію протягом тривалого часу, і вентилі, які контролюють потік інформації, що надходить у комірку, виходить з неї та стирається. Вентилі – структури, які на основі вхідних даних і поточного стану комірки

приймають рішення, яку інформацію зберегти, а яку забути. Завдяки цьому, LSTM можуть ефективно навчатися на довгих послідовностях даних.

- **Двонаправлені RNN (Bidirectional RNN, BRNN):** обробляють послідовність даних в обох напрямках – від початку до кінця і від кінця до початку. Дозволяє враховувати контекст з обох сторін кожного елемента послідовності, що покращує якість навчання.

RNN та їхні модифікації широко використовуються в задачах машинного перекладу, розпізнавання мови, генерації тексту, аналізу тональності та багатьох інших.

Повнозв'язні нейронні мережі (FFNN)

Повнозв'язні нейронні мережі (Feedforward Neural Networks, FFNN) – найбільш загальний і поширений тип нейронних мереж. Інформація в них поширюється строго в одному напрямку – від вхідного шару до вихідного, без зворотних зв'язків. Їхня архітектура може бути найрізноманітнішою: від простих одношарових перцептронів до складних багатошарових мереж з безліччю прихованих шарів.

FFNN – універсальні солдати в арсеналі машинного навчання. Їх можна використовувати для вирішення широкого кола завдань, включаючи класифікацію, регресію, кластеризацію та зниження розмірності. Вибір кількості шарів і нейронів у кожному шарі залежить від складності завдання та обсягу доступних даних.

Генеративні змагальні нейронні мережі (GAN)

Генеративні змагальні нейронні мережі (Generative Adversarial Networks, GAN) – один із найцікавіших і найперспективніших напрямків у глибокому навчанні. Їхня архітектура разюче відрізняється від усього, що ми розглядали раніше. GAN складається з двох нейронних мереж – генератора і дискримінатора, які змагаються одна з одною.

- **Генератор:** створює нові дані, схожі на ті, на яких він навчався. Наприклад, якщо GAN навчається на зображеннях облич, генератор буде намагатися створювати нові, реалістичні зображення облич.
- **Дискримінатор:** намагається відрізнити справжні дані від тих, що створив генератор.

Навчання GAN відбувається ітеративно: генератор намагається обдурити дискримінатор, а дискримінатор намагається викрити обман. У процесі змагання обидві мережі стають кращими, і врешті-решт генератор починає створювати дані, які дискримінатор не може відрізнити від справжніх.

GAN знайшли застосування в найрізноманітніших областях, включаючи генерацію зображень, відео, музики та тексту. Наприклад, вони можуть створювати реалістичні фотографії людей, яких ніколи не існувало, або домальовувати відсутні частини зображень.

Функції активації

Функції активації – важливий компонент нейронних мереж. Вони вносять нелінійність в роботу мережі, дозволяючи їй навчатися складним залежностям у даних. Якби нейрони просто підсумовували вхідні сигнали, нейронна мережа, незалежно від кількості шарів, залишалася б лінійною моделлю.

Функція активації застосовується до зваженої суми входів кожного нейрона, перетворюючи її на вихідний сигнал. Розглянемо найпоширеніші функції активації:

- **Linear (лінійна):** найпростіша функція активації. Видає те саме значення, яке отримала на вході. Використовується рідко, в основному у вихідному шарі для задач регресії.

$$f(x) = x$$

- **Sigmoid (сигмоїда):** перетворює вхідний сигнал на значення в діапазоні від 0 до 1. Раніше широко використовувалася, але зараз поступається іншим функціям через проблему зникаючого градієнта в глибоких мережах.

$$f(x) = \frac{1}{1 + exp(-x)}$$

- **Tanh (гіперболічний тангенс):** схожа на сигмоїду, але видає значення в діапазоні від -1 до 1. Також страждає від проблеми зникаючого градієнта.

$$f(x) = \frac{exp(x) - exp(-x)}{exp(x) + exp(-x)}$$

- **ReLU (Rectified Linear Unit):** найпопулярніша функція активації на сьогодні. Видає вхідне значення, якщо воно більше 0, і 0 в іншому випадку. Проста в обчисленні і не страждає від проблеми зникаючого градієнта для позитивних значень.

$$f(x) = max(0, x)$$

- **Leaky ReLU:** модифікація ReLU, яка видає невелике негативне значення для негативних входів, що допомагає уникнути проблеми «мертвих нейронів», коли нейрон перестає навчатися, видаючи нульовий вихід.

$$f(x) = max(ax, x)$$

де a – невеликий коефіцієнт, наприклад, 0.01.

- **Softmax:** використовується у вихідному шарі для задач класифікації з декількома класами. Перетворює вектор значень на вектор ймовірностей, де кожен елемент представляє ймовірність належності об'єкта до певного класу. Сума елементів вектора дорівнює 1.

$$f(x_i) = \frac{exp(x_i)}{\sum exp(x_j)}$$

для всіх j.

- **Softplus:** гладка апроксимація функції ReLU.

$$f(x) = ln(1 + exp(x))$$

Просунуті функції активації

Окрім перерахованих вище, існують і більш екзотичні функції активації, які можуть бути корисними в певних ситуаціях:

- **Swish:** відносно нова функція активації, яка показує кращі результати, ніж ReLU, у деяких задачах.

$$f(x) = x * sigmoid(x)$$

- **ELU (Exponential Linear Unit):** схожа на Leaky ReLU, але використовує експоненціальну функцію для негативних значень.

$$f(x) = x$$

якщо $x > 0$.

$$a * (exp(x) - 1)$$

якщо $x \leq 0$.

- **PReLU (Parametric ReLU):** модифікація Leaky ReLU, де коефіцієнт нахилу для негативних значень є параметром, який навчається під час навчання мережі.

$$f(x) = max(ax, x)$$

де a – параметр, що навчається.

- **SELU (Scaled Exponential Linear Unit):** функція активації, яка автоматично нормалізує виходи нейронів, що може покращити стабільність і швидкість навчання.

$$f(x) = scale * (max(0, x) + min(0, \alpha * (exp(x) - 1)))$$

- **Maxout:** узагальнення ReLU і Leaky ReLU. Вибирає максимальне значення з декількох лінійних функцій.

$$f(x) = max(w_1 * x + b_1, w_2 * x + b_2, \ldots, w_n * x + b_n)$$

- **Exponential:** просто експоненційна функція.

$$f(x) = exp(x)$$

Вибір функції активації залежить від типу нейронної мережі, завдання, яке вона вирішує, та особливостей даних. Немає універсальної функції активації, яка підходила б для всіх випадків.

Transformer

Нарешті, не можна не згадати про архітектуру Transformer, яка зробила революцію в обробці природної мови (NLP). Transformer відмовився від рекурентних і згорткових шарів на користь механізму, який називається «увага» (attention).

Механізм уваги дозволяє моделі зосереджуватися на найбільш важливих частинах вхідної послідовності при обробці кожного її елемента. Уявіть, що ви перекладаєте речення з англійської на українську. Замість того, щоб обробляти слова послідовно, ви можете спочатку подивитися на

все речення цілком, щоб зрозуміти його загальний зміст, а потім зосередитися на окремих словах, які є найбільш важливими для перекладу.

Transformer складається з декількох шарів кодувальника і декодувальника. Кодувальник обробляє вхідну послідовність і перетворює її на набір векторів, які називаються «приховане представлення». Декодувальник використовує ці вектори разом з механізмом уваги для генерації вихідної послідовності.

На основі архітектури Transformer були створені такі потужні мовні моделі, як:

- **BERT (Bidirectional Encoder Representations from Transformers):** розроблена Google. Використовує двонаправлений кодувальник для навчання мовних представлень. BERT навчений на величезному корпусі тексту і може бути донавчений для вирішення різних завдань NLP, таких як класифікація тексту, відповіді на питання і розпізнавання іменованих сутностей.
- **GPT (Generative Pre-trained Transformer):** розроблена OpenAI. Використовує тільки декодувальник і навчається генерувати текст, передбачаючи наступне слово в послідовності.

Контрольні запитання

1. Порівняйте архітектури одношарового та багатошарового перцептронів. Які обмеження має одношаровий перцептрон і як їх долає багатошаровий?
2. Опишіть принцип роботи конволюційних нейронних мереж (CNN). Які переваги надає використання операції згортки для обробки зображень? Наведіть приклади практичного застосування CNN.
3. У чому полягає особливість рекурентних нейронних мереж (RNN)? Як RNN використовують зворотний зв’язок для обробки послідовностей? Наведіть приклади задач, для яких доцільно використовувати RNN.
4. Що таке функції активації і для чого вони використовуються в нейронних мережах? Опишіть властивості та області застосування таких функцій активації, як Sigmoid, Tanh, ReLU.
5. Поясніть принцип роботи архітектури Transformer. У чому її переваги перед рекурентними мережами для обробки тексту?

Лекція 5. Оптимізація та налаштування моделей

Оптимізація

Уявіть, що ви стоїте на вершині гори і хочете спуститися в долину. Ваша мета – знайти найшвидший і найбезпечніший шлях вниз. Так само і в машинному навчанні: ми хочемо знайти такі параметри моделі, які мінімізують помилку на навчальних даних. Цей процес пошуку оптимальних параметрів називається оптимізацією.

Функція, яку ми намагаємося мінімізувати, називається функцією втрат (loss function). Вона показує, наскільки сильно модель помиляється на кожному прикладі з навчального набору. Наприклад, для задачі регресії часто використовується середньоквадратична помилка (Mean Squared Error, MSE), яка обчислює середній квадрат різниці між передбаченим і реальним значенням. Для задач класифікації поширена перехресна ентропія (Cross-Entropy), яка вимірює розбіжність між розподілом передбачених імовірностей і реальним розподілом класів.

Градієнтний спуск

Найпоширеніший метод оптимізації – градієнтний спуск (Gradient Descent). Він працює ітеративно, крок за кроком оновлюючи параметри моделі в напрямку, протилежному градієнту функції втрат. Градієнт – вектор, який вказує напрямок найшвидшого зростання функції. Відповідно, антиградієнт вказує напрямок найшвидшого спадання.

Уявіть собі м’ячик, який скочується по схилу. Він рухається в напрямку найкрутішого спуску, ведений силою тяжіння. Так само і градієнтний спуск «скочується» по поверхні функції втрат, рухаючись у напрямку, де функція зменшується найшвидше.

Оптимізатори

Існує безліч різновидів градієнтного спуску, які називаються оптимізаторами. Кожен з них має свої особливості та найкраще підходить для певних типів задач і архітектур нейронних мереж. Розглянемо найпопулярніші з них:

- **Стохастичний градієнтний спуск (Stochastic Gradient Descent, SGD):** класичний оптимізатор. На кожній ітерації оновлює параметри моделі на основі градієнта, обчисленого для одного випадково обраного прикладу з навчального набору. SGD простий у реалізації, але може сильно коливатися, особливо на початку навчання. Його можна порівняти з мандрівником, який спускається з гори в тумані, роблячи невеликі, але не завжди впевнені кроки.
- **SGD з моментом (SGD with Momentum):** модифікація SGD, яка додає інерцію до руху в напрямку антиградієнта. Момент накопичує попередні градієнти і згладжує

коливання, дозволяючи оптимізатору швидше сходитися до мінімуму. Можна уявити це як лижника, який розганяється на схилі, набираючи швидкість і долаючи невеликі перешкоди. Параметр **Momentum** (зазвичай позначається як γ або β_1), який приймає значення від 0 до 1, контролює силу інерції. Чим він більший, тим сильніший вплив попередніх градієнтів.

- **Adagrad (Adaptive Gradient Algorithm):** адаптивний алгоритм, який підлаштовує швидкість навчання (learning rate) для кожного параметра окремо, залежно від історії його градієнтів. Параметри, які рідко оновлюються, отримують більшу швидкість навчання, а ті, які оновлюються часто, – меншу. Adagrad добре підходить для розріджених даних, де багато параметрів мають нульові значення. Але у нього є недолік: з часом швидкість навчання може стати занадто малою, що призведе до зупинки навчання.
- **RMSprop (Root Mean Square Propagation):** вдосконалення Adagrad, яке вирішує проблему занадто швидкого зменшення швидкості навчання. RMSprop використовує ковзне середнє квадратів градієнтів замість накопичення всіх попередніх квадратів. Цей оптимізатор часто є хорошим вибором за замовчуванням. RMSprop, подібно до Adagrad, адаптує швидкість навчання для кожного параметра, але замість накопичення суми квадратів градієнтів, він використовує експоненціально зважене ковзне середнє. Параметр **decay rate** (зазвичай позначається як ρ або β_2) контролює швидкість затухання попередніх градієнтів.
- **Adadelta:** ще одне розширення Adagrad, яке усуває необхідність вручну встановлювати швидкість навчання. Замість накопичення квадратів градієнтів, Adadelta обмежує вікно накопичення попередніх градієнтів фіксованим розміром. Цей метод робить оптимізатор менш чутливим до вибору гіперпараметрів.
- **Adam (Adaptive Moment Estimation):** один із найпопулярніших оптимізаторів на сьогодні. Поєднує в собі ідеї моменту і RMSprop. Adam обчислює як ковзне середнє градієнтів, так і ковзне середнє їхніх квадратів, і використовує їх для адаптації швидкості навчання для кожного параметра. Adam, як правило, працює добре в широкому діапазоні задач і часто є найкращим вибором серед адаптивних методів.
- **Nadam (Nesterov-accelerated Adaptive Moment Estimation):** модифікація Adam, яка включає момент Нестерова. Момент Нестерова – вдосконалена версія стандартного моменту, яка спочатку робить крок у напрямку попереднього накопиченого градієнта, а потім обчислює градієнт у новій точці. Зазвичай сходиться трохи швидше, ніж Adam.

Вибір оптимізатора

Не існує універсального оптимізатора, який ідеально підходив би для всіх задач. Вибір оптимізатора залежить від типу даних, архітектури нейронної мережі, розміру навчального набору та інших факторів.

Ось кілька загальних рекомендацій:

- Для невеликих наборів даних і неглибоких мереж добре працює **SGD з моментом**.
- Для глибоких мереж часто краще підходять адаптивні методи, такі як **Adam** або **RMSprop**.
- **Adam** часто є хорошим вибором за замовчуванням, оскільки він поєднує в собі переваги інших методів.
- Якщо дані розріджені, варто спробувати **Adagrad** або **RMSprop**.
- **Nadam** може бути трохи швидшим за **Adam**, але не завжди.

Найкращий спосіб вибрати оптимізатор – поекспериментувати з різними варіантами та порівняти їхню ефективність на вашій задачі.

Додаткові параметри

Оптимізатори мають додаткові параметри, які впливають на їхню поведінку. Розглянемо найважливіші з них:

- **Швидкість навчання (Learning Rate):** мабуть, найважливіший гіперпараметр. Визначає розмір кроку, який робить оптимізатор у напрямку антиградієнта. Занадто велика швидкість навчання може призвести до того, що оптимізатор «перестрибне» мінімум і не зможе знайти оптимальне рішення. Занадто мала швидкість навчання призведе до повільного навчання і може застрягти в локальному мінімумі. Зазвичай швидкість навчання вибирають в діапазоні від 0.0001 до 0.1.

Існують різні стратегії зміни швидкості навчання під час навчання:

- **Постійна швидкість навчання:** найпростіший варіант, коли швидкість навчання залишається незмінною протягом усього навчання.
- **Спадаюча швидкість навчання:** швидкість навчання поступово зменшується з часом, наприклад, за експоненціальним законом або за розкладом. Це дозволяє оптимізатору робити великі кроки на початку навчання і дрібні кроки ближче до мінімуму.
- **Адаптивна швидкість навчання:** швидкість навчання автоматично підлаштовується під час навчання, як у методах Adagrad, RMSprop і Adam.

- **Параметри Beta (β_1 та β_2):** використовуються в оптимізаторах Adam і Nadam для керування експоненціальним зваженим ковзним середнім градієнтів та їхніх квадратів. β_1 зазвичай встановлюється в 0.9, а β_2 – в 0.999.
- **Momentum (γ або β_1):** використовується в SGD з моментом і Nadam для накопичення попередніх градієнтів. Зазвичай встановлюється в 0.9 або 0.99.

Гіперпараметри

Окрім параметрів оптимізатора, у моделі машинного навчання є й інші параметри, які не навчаються під час оптимізації, але суттєво впливають на її ефективність. Їх називають гіперпараметрами.

До гіперпараметрів належать:

- **Архітектура мережі:** кількість шарів, кількість нейронів у кожному шарі, тип функції активації.
- **Параметри оптимізатора:** швидкість навчання, момент, параметри Beta.
- **Розмір пакета (batch size):** кількість прикладів, які обробляються за одну ітерацію оптимізатора.
- **Кількість епох (epochs):** кількість повних проходів по всьому навчальному набору даних.
- **Параметри регуляризації:** коефіцієнт регуляризації L1 або L2, dropout rate.

Методи налаштування гіперпараметрів: від ручного пошуку до автоматизації

Налаштування гіперпараметрів – важливий етап у розробці моделей машинного навчання. Від нього залежить, наскільки добре модель буде працювати на нових даних. Розглянемо основні методи налаштування гіперпараметрів:

- **Ручне налаштування (Manual Search):** найпростіший, але найменш ефективний метод. Дослідник вручну підбирає значення гіперпараметрів, ґрунтуючись на своєму досвіді та інтуїції. Цей підхід вимагає глибокого розуміння роботи моделі та може займати багато часу.
- **Пошук по сітці (Grid Search):** автоматизований метод, який перебирає всі можливі комбінації значень гіперпараметрів із заданого діапазону. Для кожної комбінації модель навчається і оцінюється на валідаційному наборі даних. Потім вибирається комбінація, яка дає найкращі результати. Grid Search гарантує, що буде знайдено найкращу комбінацію з заданого діапазону, але може бути дуже обчислювально дорогим, особливо коли кількість гіперпараметрів велика.

- **Випадковий пошук (Random Search):** альтернатива Grid Search, яка замість перебору всіх можливих комбінацій випадковим чином вибирає задану кількість комбінацій з заданого діапазону. Дослідження показують, що Random Search часто знаходить хороші комбінації гіперпараметрів швидше, ніж Grid Search, особливо коли є багато гіперпараметрів, які слабо впливають на продуктивність.
- **Байєсівська оптимізація (Bayesian Optimization):** більш просунутий метод, який використовує імовірнісну модель для апроксимації функції продуктивності моделі залежно від значень гіперпараметрів. На кожній ітерації байєсівська оптимізація вибирає наступну комбінацію гіперпараметрів, яка має найбільший потенціал для поліпшення результату, ґрунтуючись на поточній моделі. Цей підхід дозволяє знаходити хороші комбінації гіперпараметрів швидше, ніж Grid Search і Random Search, особливо коли функція продуктивності складна і дорога в обчисленні.

Контрольні запитання

1. Що таке оптимізація в контексті машинного навчання? Опишіть принцип роботи градієнтного спуску.
2. Порівняйте оптимізатори Adam, SGD, RMSprop. Які переваги та недоліки кожного з них? Коли доцільно використовувати той чи інший оптимізатор?
3. Що таке швидкість навчання (Learning Rate), момент (Momentum) та параметри Beta в оптимізаторах? Як вони впливають на процес навчання моделі?
4. Що таке гіперпараметри? Чим вони відрізняються від параметрів моделі? Наведіть приклади гіперпараметрів для нейронних мереж.
5. Опишіть методи налаштування гіперпараметрів: Grid Search, Random Search, Bayesian Optimization. Які переваги та недоліки кожного з цих методів?

Лекція 6. Навчання та валідація моделей

Параметри регуляризації

Навчання моделі – балансування між двома крайнощами: недонавчанням і перенавчанням. Недонавчання виникає, коли модель занадто проста і не здатна вловити складні залежності в даних. Перенавчання, навпаки, коли модель занадто добре підлаштовується під навчальні дані, вивчаючи не тільки загальні закономірності, а й шум і випадкові флуктуації. В результаті модель погано узагальнює на нові дані.

Щоб уникнути цих проблем, використовуються параметри регуляризації – своєрідні «ручки управління», які дозволяють тонко налаштувати процес навчання. Розглянемо найважливіші з них:

- **Epoch (Епоха):** один повний прохід по всьому навчальному набору даних. Кількість епох – важливий гіперпараметр, який визначає тривалість навчання. Занадто мала кількість епох може призвести до недонавчання, а занадто велика – до перенавчання.
- **Batch Size (Розмір пакета):** кількість прикладів, які обробляються за одну ітерацію оптимізатора (за один крок оновлення ваг). Великий розмір пакета прискорює навчання, але може вимагати більше пам’яті. Малий розмір пакета робить процес навчання більш стохастичним, що може бути корисним для виходу з локальних мінімумів, але може сповільнити навчання. Зазвичай розмір пакета вибирають ступенем двійки: 32, 64, 128, 256 і т.д.
- **Validation Split (Частка валідаційного набору):** частина навчального набору даних, яка використовується для оцінки ефективності моделі під час навчання та налаштування гіперпараметрів. Дозволяє відстежувати перенавчання і вибирати найкращу модель. Зазвичай для валідації виділяють 10-20% навчальних даних.
- **Dropout Rate (Коефіцієнт проріджування):** техніка регуляризації, яка полягає у випадковому «вимкненні» (обнуленні виходів) деякої частки нейронів під час кожного проходу навчання. Змушує мережу вчитися розподіляти інформацію між різними нейронами і робить її більш стійкою до шуму в даних. Значення коефіцієнта проріджування зазвичай вибирається в діапазоні від 0.2 до 0.5.
- **Weight Initialization (Ініціалізація ваг):** початкові значення ваг нейронної мережі можуть суттєво впливати на швидкість і якість навчання. Випадкова ініціалізація ваг невеликими значеннями допомагає уникнути симетрії в роботі нейронів і забезпечує ефективне навчання. Існують різні методи ініціалізації, такі як Xavier/Glorot initialization і He initialization, які враховують кількість входів і виходів нейрона.
- **Regularization Techniques (Методи регуляризації):** загальна назва технік, спрямованих на запобігання перенавчанню. До них, окрім Dropout, належать L1 і L2

регуляризація, які додають до функції втрат штраф за великі значення ваг. L1 регуляризація (сума модулів ваг) призводить до розрідженості моделі, тобто обнулення деяких ваг. L2 регуляризація (сума квадратів ваг) просто робить ваги меншими.

- **Learning Rate Decay (Зменшення швидкості навчання):** техніка, яка полягає в поступовому зменшенні швидкості навчання під час навчання. Дозволяє моделі робити великі кроки на початку навчання і більш точні кроки ближче до мінімуму. Існують різні розклади зменшення швидкості навчання, наприклад, експоненціальний, степеневий або зменшення на плато (коли точність на валідаційному наборі перестає покращуватися).
- **Gradient Clipping (Обрізка градієнта):** техніка, яка обмежує значення градієнта під час навчання, щоб запобігти вибуху градієнта, особливо в рекурентних нейронних мережах. Коли норма градієнта перевищує певний поріг, він масштабується вниз.

Методи ранньої зупинки та регуляризації: запобігаємо перенавчанню

Регуляризація допомагає запобігти перенавчанню, але існують і інші методи боротьби з цим явищем. Один з найпоширеніших – рання зупинка (Early Stopping).

Рання зупинка (Early Stopping): моніторинг продуктивності моделі на валідаційному наборі даних під час навчання. Навчання зупиняється, коли продуктивність на валідаційному наборі перестає покращуватися протягом певної кількості епох (параметр patience). Рання зупинка дозволяє уникнути перенавчання і знайти модель з найкращою узагальнюючою здатністю.

Критерії оцінки ефективності моделі: вимірюємо успіх

Щоб зрозуміти, наскільки добре працює наша модель, потрібні метрики оцінки якості. Вибір метрики залежить від типу задачі, яку вирішує модель. Розглянемо найпоширеніші метрики:

Задачі класифікації:

- **Accuracy (Точність):** частка правильних відповідей моделі. Найпростіша і найінтуїтивніша метрика, але може бути оманливою для незбалансованих класів (коли кількість прикладів у різних класах сильно відрізняється).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precision (Точність):** частка об’єктів, які модель класифікувала як позитивні і які дійсно є позитивними. Показує, наскільки можна довіряти моделі, коли вона класифікує об’єкт як позитивний.

$$Precision = \frac{TP}{TP + FP}$$

- **Recall (Повнота):** частка об’єктів, які модель правильно класифікувала як позитивні, відносно всіх об’єктів, які дійсно є позитивними. Показує, яку частку позитивних об’єктів модель здатна виявити.

$$Recall = \frac{TP}{TP + FN}$$

- **F1-Score (F1-міра):** середнє гармонійне точності і повноти. Є збалансованою метрикою, яка враховує як точність, так і повноту.

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

- **Confusion Matrix (Матриця невідповідностей):** таблиця, яка показує кількість істинно позитивних (TP), істинно негативних (TN), хибно позитивних (FP) і хибно негативних (FN) класифікацій. Дозволяє детально проаналізувати помилки моделі.
- **ROC-крива (Receiver Operating Characteristic curve):** графік, який показує залежність частки істинно позитивних класифікацій (True Positive Rate, TPR) від частки хибно позитивних класифікацій (False Positive Rate, FPR) при різних значеннях порогу класифікації.
- **AUC (Area Under the ROC Curve):** площа під ROC-кривою. Чим ближче AUC до 1, тим краще працює класифікатор.

Задачі регресії:

- **MAE (Mean Absolute Error):** середня абсолютна різниця між передбаченими і реальними значеннями.

$$MAE = \frac{1}{n} * \sum |y_i - \hat{y}_i|$$

- **MSE (Mean Squared Error):** середня квадратична різниця між передбаченими і реальними значеннями. Більш чутлива до великих помилок, ніж MAE.

$$MSE = \frac{1}{n} * \sum (y_i - \hat{y}_i)^2$$

- **RMSE (Root Mean Squared Error):** корінь квадратний з MSE. Має ту ж розмірність, що й вихідна змінна, що полегшує інтерпретацію.
- **Log Loss (Логарифмічна втрата):** використовується для оцінки якості ймовірнісних прогнозів. Вимірює розбіжність між розподілом передбачених імовірностей і реальним розподілом.

Стратегії валідації: перевіряємо на міцність

Валідація – процес оцінки узагальнюючої здатності моделі, тобто її здатності працювати на нових, небачених раніше даних. Існує декілька стратегій валідації:

- **Holdout Method (Відкладена вибірка):** найпростіший метод. Дані діляться на три частини: навчальний набір, валідаційний набір і тестовий набір. Модель навчається на навчальному наборі, її гіперпараметри налаштовуються на валідаційному наборі, а остаточна оцінка якості проводиться на тестовому наборі. Недолік методу полягає в тому, що результати можуть сильно залежати від того, як саме дані були розділені на три частини.
- **k-fold Cross-Validation (Крос-валідація по k блоках):** дані діляться на k рівних частин (блоків). Модель навчається k разів, кожен раз використовуючи один з блоків як валідаційний набір, а решту k-1 блоків – як навчальний набір. Результати усереднюються по всіх k ітераціях. Цей метод більш надійний, ніж Holdout, оскільки використовує весь набір даних як для навчання, так і для валідації.
- **Stratified K-Fold Cross-Validation (Стратифікована крос-валідація по k блоках):** різновид k-fold крос-валідації, який зберігає пропорції класів у кожному блоці. Корисний для незбалансованих даних.
- **LOOCV (Leave-One-Out Cross-Validation):** окремий випадок k-fold крос-валідації, коли k дорівнює кількості прикладів у наборі даних. Кожен приклад по черзі використовується як валідаційний, а решта даних – як навчальний набір. Дуже обчислювально дорогий метод, але може бути корисним для невеликих наборів даних.
- **Time Series Cross-Validation (Крос-валідація для часових рядів):** спеціальний метод валідації для даних, які мають часову структуру. Дані діляться на блоки послідовно в

часі, і модель навчається на даних, що передують валідаційному блоку. Цей метод дозволяє уникнути «заглядання в майбутнє», коли модель навчається на даних, які в реальному житті ще не були б доступні.

- **Bootstrap Method (Метод бутстрепу):** з набору даних випадковим чином з поверненням вибирається n вибірок (n – розмір вихідного набору). Кожна вибірка використовується як навчальний набір, а приклади, які не потрапили у вибірку, утворюють тестовий набір. Повторюється багато разів, і результати усереднюються. Дозволяє оцінити статистичну значущість результатів.
- **Monte Carlo Cross-Validation (Крос-валідація Монте-Карло):** також відома як Repeated random sub-sampling validation. Дані випадковим чином діляться на навчальний і валідаційний набори багато разів. Для кожної ітерації модель навчається і тестується, а результати усереднюються. Схожа на k-fold, але дозволяє більш гнучко контролювати розмір навчального і валідаційного наборів.

Контрольні запитання

1. Що таке епоха (epoch) і розмір пакета (batch size) в контексті навчання нейронних мереж? Як вони впливають на швидкість і якість навчання?
2. Поясніть, що таке перенавчання (overfitting) і недонавчання (underfitting). Як можна виявити ці проблеми і як з ними боротися?
3. Опишіть методи регуляризації: L1, L2, dropout, early stopping. Як вони допомагають запобігти перенавчанню?
4. Які критерії використовуються для оцінки ефективності моделей класифікації? Поясніть, що таке accuracy, precision, recall, F1-score.
5. Опишіть різні стратегії валідації: Holdout, k-fold cross-validation, Stratified k-fold cross-validation, LOOCV. Які переваги і недоліки кожної з них?

МОДУЛЬ 3. ПЕРЕДОВІ МЕТОДИ У MACHINE LEARNING ТА DEEP LEARNING

Лекція 7. Machine Learning з вчителем. Регресія та класифікація.

Регресія

Завдання регресії полягає в тому, щоб передбачити числове значення цільової змінної на основі значень однієї або декількох вхідних змінних (ознак). Наприклад, ми можемо спробувати передбачити ціну квартири за її площею, кількістю кімнат, відстанню від центру міста та іншими характеристиками. Або спрогнозувати зріст людини за її вагою, віком і статтю.

Найпростіший і найпоширеніший метод регресії – лінійна регресія (Linear Regression). Вона передбачає, що залежність між вхідними змінними та цільовою змінною є лінійною. Тобто, цільову змінну можна представити як лінійну комбінацію вхідних змінних з деякими коефіцієнтами (вагами).

Уявіть, що ми хочемо передбачити зріст людини за її вагою. Припустимо, що залежність між цими величинами лінійна. Тоді ми можемо побудувати пряму лінію, яка найкращим чином апроксимує цю залежність. Коефіцієнт нахилу цієї прямої і буде вагою, яка відповідає ознаці «вага».

Завдання лінійної регресії – знайти такі значення ваг, які мінімізують помилку між передбаченими і реальними значеннями цільової змінної. Для цього зазвичай використовується метод найменших квадратів (Ordinary Least Squares, OLS), який мінімізує суму квадратів різниць між передбаченими і реальними значеннями.

Однак, далеко не завжди залежність між змінними є лінійною. Іноді вона може бути криволінійною, наприклад, квадратичною, кубічною або мати ще складнішу форму. У таких випадках на допомогу приходить поліноміальна регресія (Polynomial Regression).

Поліноміальна регресія – розширення лінійної регресії, яке дозволяє моделювати нелінійні залежності. Вона вводить в модель поліноміальні члени, тобто степені вхідних змінних. Наприклад, замість простої лінійної залежності $y = w_0 + w_1 * x$ ми можемо використовувати квадратичну залежність $y = w_0 + w_1 * x + w_2 * x^2$ або кубічну $y = w_0 + w_1 * x + w_2 * x^2 + w_3 * x^3$.

Використання поліноміальної регресії високих ступенів може призвести до перенавчання, коли модель занадто добре підлаштовується під навчальні дані і погано узагальнює на нові. Щоб уникнути цього, застосовуються методи регуляризації, такі як Ridge і Lasso регресія.

Ridge регресія (Ridge Regression) додає до функції втрат штраф за великі значення ваг, а саме суму квадратів ваг, помножену на коефіцієнт регуляризації (λ). Цей штраф змушує модель вибирати менші значення ваг, що робить її простішою і більш стійкою до перенавчання.

Lasso регресія (Lasso Regression) діє аналогічно, але замість суми квадратів ваг використовує суму їхніх модулів. Крім регуляризації, Lasso регресія також може призводити до відбору ознак, тобто обнулення ваг деяких менш важливих ознак.

Вибір між Ridge і Lasso регресією залежить від конкретної задачі. Ridge регресія зазвичай працює краще, коли всі ознаки важливі, а Lasso регресія – коли є багато ознак, але тільки деякі з них дійсно впливають на цільову змінну.

Класифікація

Якщо регресія передбачає числові значення, то класифікація має справу з категоріальними даними. Завдання класифікації полягає в тому, щоб віднести об'єкт до одного з декількох заздалегідь визначених класів. Наприклад, ми можемо класифікувати електронні листи на спам і не спам, розпізнавати рукописні цифри від 0 до 9 або визначати породу собаки на фотографії.

Існує безліч алгоритмів класифікації, кожен з яких має свої сильні та слабкі сторони. Розглянемо деякі з найпопулярніших:

- **Метод k-найближчих сусідів (k-Nearest Neighbors, k-NN):** один із найпростіших і найінтуїтивніших алгоритмів класифікації. Ідея полягає в тому, що об'єкт класифікується на основі того, до якого класу належать його найближчі сусіди в навчальному наборі даних. Уявіть, що ви потрапили в незнайоме місто і хочете знайти хороший ресторан. Ви можете запитати у k перехожих, які ресторани їм подобаються, і вибрати той, який рекомендує більшість із них. Так само працює і k-NN: він «дивиться» на k найближчих сусідів об'єкта і відносить його до того класу, який переважає серед цих сусідів. Відстань між об'єктами зазвичай обчислюється за допомогою евклідової метрики, але можуть використовуватися й інші метрики.
- **Наївний баєсівський класифікатор (Naive Bayes):** базується на теоремі Баєса з теорії ймовірностей. Попри свою «наївність» (він передбачає, що всі ознаки незалежні), цей алгоритм часто показує хороші результати на практиці, особливо в задачах класифікації тексту. Наприклад, він може використовуватися для фільтрації спаму, аналізу тональності тексту або визначення тематики новинних статей.
- **Метод опорних векторів (Support Vector Machine, SVM):** будує гіперплощину, яка найкращим чином розділяє об'єкти різних класів у багатовимірному просторі ознак. SVM прагне знайти таку гіперплощину, щоб відстань від неї до найближчих об'єктів кожного класу (опорних векторів) була максимальною. Це робить SVM стійким до

шуму і викидів у даних. Алгоритм може використовуватися і для нелінійної класифікації, якщо застосовувати так зване «ядрове перетворення», яке відображає дані в простір вищої розмірності, де вони стають лінійно роздільними.

- **Дерева рішень (Decision Trees):** будують деревоподібну структуру, де кожен внутрішній вузол відповідає перевірці деякої ознаки, кожна гілка – результату перевірки, а кожен лист – передбаченому класу. Дерева рішень легко інтерпретуються і дозволяють зрозуміти, які ознаки є найбільш важливими для класифікації. Однак, вони схильні до перенавчання, особливо коли дерево дуже глибоке.
- **Логістична регресія (Logistic Regression):** незважаючи на свою назву, є алгоритмом класифікації, а не регресії. Вона моделює ймовірність належності об'єкта до певного класу за допомогою логістичної функції (сигмоїди). Логістична регресія, подібно до лінійної регресії, знаходить лінійну комбінацію ознак, але потім перетворює її за допомогою логістичної функції, щоб отримати значення в діапазоні від 0 до 1, яке інтерпретується як ймовірність.

Зміна порогу класифікації

Багато алгоритмів класифікації, такі як логістична регресія або SVM, видають не просто мітку класу, а ймовірність належності об'єкта до кожного класу. Щоб отримати остаточний прогноз, потрібно вибрати поріг класифікації – значення ймовірності, вище якого об'єкт буде віднесений до позитивного класу.

За замовчуванням поріг часто встановлюється на рівні 0.5, тобто об'єкт класифікується як позитивний, якщо ймовірність належності до позитивного класу перевищує 0.5. Однак, цей поріг не завжди є оптимальним.

Уявіть, що ви розробляєте систему для діагностики раку. Хибно негативний результат (пропуск хвороби) в цьому випадку набагато гірший, ніж хибно позитивний (помилкова тривога). Тому вам може знадобитися знизити поріг, щоб збільшити повноту (частку виявлених хворих), нехай і ціною зниження точності.

З іншого боку, якщо ви розробляєте систему для фільтрації спаму, хибно позитивний результат (позначення нормального листа як спаму) може бути більш небажаним, ніж хибно негативний (пропуск спам-листа). У цьому випадку вам може знадобитися підвищити поріг, щоб збільшити точність, нехай і ціною зниження повноти.

Вибір оптимального порогу залежить від конкретної задачі та від того, яка з помилок (хибно позитивна чи хибно негативна) є більш критичною.

Використання вагових коефіцієнтів

Часто в реальних задачах класифікації зустрічається дисбаланс класів, коли кількість об’єктів одного класу значно перевищує кількість об’єктів іншого класу. Наприклад, у задачі виявлення шахрайських транзакцій кількість шахрайських транзакцій зазвичай набагато менша, ніж кількість нормальних.

Якщо навчати модель на таких даних без урахування дисбалансу, вона може навчитися добре передбачати домінуючий клас, але погано розпізнавати рідкісний клас. Щоб подолати цю проблему, можна використовувати вагові коефіцієнти.

Ідея полягає в тому, щоб присвоїти кожному класу вагу, обернено пропорційну його частоті в навчальному наборі даних. Таким чином, рідкісний клас отримає більшу вагу, і модель буде приділяти йому більше уваги під час навчання.

Більшість бібліотек машинного навчання, таких як Scikit-learn, дозволяють задавати вагові коефіцієнти для класів при навчанні моделей.

Контрольні запитання

1. У чому полягає завдання регресії в машинному навчанні? Наведіть приклади практичних завдань, які можна вирішити за допомогою регресії.
2. Опишіть принцип роботи лінійної регресії. Як знаходиться оптимальна лінійна залежність між вхідними змінними та цільовою змінною?
3. Що таке поліноміальна регресія і коли її доцільно використовувати? Як поліноміальна регресія дозволяє моделювати нелінійні залежності?
4. У чому полягає завдання класифікації? Наведіть приклади практичних завдань класифікації.
5. Порівняйте методи класифікації: k-NN, Naive Bayes, SVM, Decision Trees, Logistic Regression. Які їхні переваги, недоліки та особливості застосування?

Лекція 8. Machine Learning без вчителя. Кластеризація та асоціація.

Кластеризація

Уявіть, що перед вами велика купа різнокольорових ґудзиків. Вони відрізняються за розміром, формою, матеріалом і кольором. Завдання кластеризації полягає в тому, щоб розділити ці ґудзики на групи (кластери) таким чином, щоб ґудзики всередині кожної групи були максимально схожі один на одного, а ґудзики з різних груп максимально відрізнялися.

Кластеризація – потужний інструмент аналізу даних, який дозволяє виявляти групи схожих об’єктів, виявляти аномалії та отримувати нові знання про структуру даних. Алгоритми кластеризації знаходять застосування в найрізноманітніших областях, від сегментації клієнтів у маркетингу до аналізу генів у біології.

Розглянемо найпопулярніші методи кластеризації:

- **K-Means (K-середніх):** один із найвідоміших і найпоширеніших алгоритмів кластеризації. Його мета – розбити множину об’єктів на K кластерів таким чином, щоб мінімізувати суму квадратів відстаней від кожного об’єкта до центру його кластера (центроїда). Алгоритм працює ітеративно, оновлюючи положення центроїдів на кожному кроці, поки вони не перестануть змінюватися. K-Means простий у реалізації та ефективний, але має свої недоліки. По-перше, він чутливий до вибору початкового положення центроїдів, що може призвести до потрапляння в локальний мінімум. По-друге, він передбачає, що кластери мають сферичну форму і приблизно однаковий розмір, що не завжди відповідає дійсності. По-третє, потрібно заздалегідь знати кількість кластерів K, що не завжди можливо.
- **DBSCAN (Density-Based Spatial Clustering of Applications with Noise):** алгоритм кластеризації, заснований на щільності. На відміну від K-Means, DBSCAN не вимагає заздалегідь задавати кількість кластерів і може виявляти кластери довільної форми. DBSCAN визначає кластери як області високої щільності, розділені областями низької щільності. Об’єкт вважається частиною кластера, якщо в його ε-околі знаходиться не менше MinPts інших об’єктів. Об’єкти, які не потрапляють у жоден кластер, вважаються шумом. DBSCAN ефективний для виявлення кластерів складної форми і аномалій, але може бути чутливим до вибору параметрів ε і MinPts.
- **Mean-Shift:** непараметричний алгоритм кластеризації, який не вимагає знання кількості кластерів. Ідея полягає в ітеративному зсуві кожної точки даних у напрямку локального максимуму щільності розподілу. Уявіть, що кожна точка даних випускає «сигнал», який притягує до себе сусідні точки. Згодом точки збираються в групи навколо локальних максимумів щільності, які і стають центрами кластерів. Mean-Shift добре підходить для

кластеризації даних, де кластери мають неправильну форму, але може бути обчислювально дорогим для великих наборів даних.

- **Agglomerative Clustering (Агломеративна кластеризація):** ієрархічний алгоритм кластеризації, який починає з того, що кожен об'єкт вважається окремим кластером, а потім послідовно об'єднує найближчі кластери, поки всі об'єкти не опиняться в одному кластері. Результатом є деревоподібна структура (дендрограма), яка показує ієрархію кластерів. Агломеративна кластеризація дозволяє візуалізувати структуру даних і вибирати оптимальну кількість кластерів, розрізаючи дендрограму на певному рівні. Однак, алгоритм може бути обчислювально дорогим для великих наборів даних і чутливим до шуму.
- **Fuzzy C-Means (Нечіткі С-середні):** розширення алгоритму K-Means, яке дозволяє об'єктам належати до декількох кластерів одночасно з різним ступенем належності. Замість жорсткого приписування об'єкта до одного кластера, Fuzzy C-Means обчислює ймовірність належності об'єкта до кожного кластера. Це може бути корисним, коли кластери перекриваються або коли об'єкти знаходяться на межі між кластерами.

Асоціативні правила

Інший важливий напрямок навчання без учителя – пошук асоціативних правил (Association Rule Learning). Його мета – виявити стійкі взаємозв'язки між об'єктами або подіями в наборі даних.

Класичний приклад – аналіз покупок у супермаркеті. Алгоритми пошуку асоціативних правил можуть виявити, що клієнти, які купують пиво, часто купують і чіпси, або що покупка підгузків часто супроводжується покупкою дитячого харчування. Такі знання можуть бути використані для оптимізації розміщення товарів на полицях, розробки маркетингових акцій або рекомендації товарів покупцям.

Асоціативне правило має вигляд «Якщо А, то В», де А і В – набори об'єктів або подій. Наприклад, «Якщо {пиво, чіпси}, то {підгузки}».

Для оцінки сили асоціативного правила використовуються такі метрики:

- **Support (Підтримка):** частка транзакцій, які містять як А, так і В. Показує, наскільки часто зустрічається правило в даних.
- **Confidence (Достовірність):** ймовірність того, що В відбудеться, якщо відбулося А. Показує, наскільки точним є правило.
- **Lift (Підйом):** відношення достовірності правила до ймовірності В. Показує, у скільки разів ймовірність В збільшується за умови А.

Розглянемо основні алгоритми пошуку асоціативних правил:

- **Apriori:** один із найвідоміших алгоритмів. Він використовує ітеративний підхід для пошуку частих наборів об’єктів (тих, які зустрічаються в даних частіше, ніж заданий поріг підтримки). На кожній ітерації алгоритм генерує набори-кандидати більшого розміру на основі частих наборів, знайдених на попередній ітерації, і перевіряє їхню підтримку. Apriori ефективний для невеликих наборів даних, але може бути обчислювально дорогим для великих наборів з великою кількістю унікальних об’єктів.
- **Eclat (Equivalence Class Clustering and bottom-up Lattice Traversal):** використовує представлення даних у вигляді вертикального формату, де для кожного об’єкта зберігається список транзакцій, в яких він зустрічається. Eclat будує решітку наборів об’єктів і обходить її знизу вгору, обчислюючи підтримку кожного набору шляхом перетину списків транзакцій. Eclat може бути швидшим за Apriori, особливо для щільних даних.
- **FP-Growth (Frequent Pattern Growth):** використовує компактну структуру даних, звану FP-деревом (Frequent Pattern tree), для зберігання інформації про часті набори об’єктів. FP-дерево будується за один прохід по даних, а потім використовується для генерації частих наборів без необхідності повторного сканування даних. FP-Growth зазвичай швидший за Apriori і Eclat, особливо для великих наборів даних.

Зниження розмірності

Багато наборів даних містять велику кількість ознак, що може ускладнювати їхній аналіз і візуалізацію. Крім того, деякі ознаки можуть бути надлишковими або неінформативними. Зниження розмірності (Dimensionality Reduction) – сукупність методів, які дозволяють зменшити кількість ознак, зберігаючи при цьому найважливішу інформацію.

Зниження розмірності може бути використане для вирішення таких завдань:

- **Стиснення даних:** зменшення обсягу пам’яті, необхідної для зберігання даних.
- **Візуалізація даних:** представлення багатовимірних даних у двовимірному або тривимірному просторі для зручності візуального аналізу.
- **Попередня обробка даних:** видалення шуму, усунення мультиколінеарності та виділення найбільш інформативних ознак перед застосуванням інших алгоритмів машинного навчання.
- **Виявлення прихованих факторів:** ідентифікація невеликої кількості латентних змінних, які пояснюють більшу частину варіативності даних.

Одним із методів зниження розмірності є автоенкодери (Autoencoders).

Автоенкодери

Автоенкодер – нейронна мережа, яка навчається відновлювати вхідні дані на своєму виході. Мережа складається з двох частин: кодувальника (encoder) і декодувальника (decoder). Кодувальник перетворює вхідні дані в низьковимірне представлення (латентний простір), а декодувальник відновлює вихідні дані з цього представлення.

Навчаючись відновлювати вхідні дані, автоенкодер вчиться виділяти найважливіші ознаки і відкидати шум. Розмірність латентного простору зазвичай набагато менша за розмірність вхідних даних, що дозволяє використовувати автоенкодери для стиснення даних.

Крім стиснення, автоенкодери можуть використовуватися для вирішення інших завдань, таких як:

- **Видалення шуму з даних:** якщо навчити автоенкодер відновлювати «чисті» дані з «зашумлених», він навчиться видаляти шум.
- **Генерація нових даних:** навчений автоенкодер може генерувати нові дані, схожі на ті, на яких він навчався, шляхом подачі випадкових векторів у латентний простір і подальшого їх декодування.
- **Виявлення аномалій:** автоенкодер навчений на нормальних даних буде погано відновлювати аномальні дані, що дозволяє використовувати його для виявлення аномалій.

Контрольні запитання

1. Що таке кластеризація і чим вона відрізняється від класифікації? Наведіть приклади практичних завдань, які можна вирішити за допомогою кластеризації.
2. Опишіть принцип роботи алгоритму K-Means. Які його основні недоліки? Як можна визначити оптимальну кількість кластерів для K-Means?
3. Порівняйте алгоритми кластеризації: K-Means, DBSCAN, Mean-Shift, Agglomerative Clustering. Які їхні переваги, недоліки та особливості застосування?
4. Що таке асоціативні правила? Як вони використовуються для аналізу даних? Наведіть приклади практичного застосування асоціативних правил.
5. Опишіть алгоритми пошуку асоціативних правил: Apriori, Eclat, FP-Growth. У чому полягає їхня відмінність?

Лекція 9. Спеціалізовані методи Machine Learning. Узагальнення. Ансамблеві методи. Reinforcement Learning.

Методи узагальнення

Методи узагальнення (dimensionality reduction techniques), які ми частково розглянули в попередній лекції, відіграють важливу роль у машинному навчанні, дозволяючи не тільки стискати дані, але й виявляти приховані фактори, що лежать в основі спостережуваних явищ.

Уявіть, що ви фотографуєте тривимірний об'єкт з різних ракурсів. Кожна фотографія – двовимірне зображення, але всі вони містять інформацію про один і той самий об'єкт. Методи узагальнення дозволяють виявити ці три виміри, що лежать в основі безлічі двовимірних зображень.

Розглянемо деякі з найпопулярніших методів узагальнення:

- **Метод головних компонент (Principal Component Analysis, PCA):** один із найвідоміших і найпоширеніших методів. Його суть полягає в тому, щоб знайти такі нові змінні (головні компоненти), які є лінійними комбінаціями вихідних змінних і пояснюють більшу частину дисперсії даних. Головні компоненти впорядковуються за спаданням поясненої дисперсії, тобто перша головна компонента пояснює найбільшу частку дисперсії, друга – трохи меншу і так далі. Застосовуючи PCA, можна зменшити розмірність даних, відкинувши головні компоненти, які пояснюють незначну частку дисперсії. PCA широко використовується для стиснення даних, візуалізації, а також для попередньої обробки даних перед застосуванням інших алгоритмів машинного навчання.
- **Сингулярний розклад матриць (Singular Value Decomposition, SVD):** ще один потужний метод лінійної алгебри, який часто використовується для зниження розмірності. SVD розкладає матрицю даних на три матриці: U, Σ і V*. Матриця Σ містить сингулярні значення, які є мірою важливості відповідних компонент. Залишаючи тільки найбільші сингулярні значення і відповідні їм стовпці в матрицях U і V*, можна отримати низькорангове наближення вихідної матриці. SVD застосовується в рекомендаційних системах, обробці зображень і зниженні шуму.
- **Латентно-семантичний аналіз (Latent Semantic Analysis, LSA):** різновид SVD, який використовується для аналізу текстових даних. LSA будує матрицю «термін-документ», де рядки відповідають термінам, а стовпці – документам. Кожен елемент матриці відображає частоту вживання терміна в документі. Застосовуючи SVD до цієї матриці, LSA виявляє приховані семантичні зв'язки між термінами і документами, що дозволяє

групувати документи за темами, знаходити схожі документи і здійснювати семантичний пошук.

- **t-розподілене стохастичне вкладення сусідів (t-distributed Stochastic Neighbor Embedding, t-SNE):** нелінійний метод зниження розмірності, який особливо добре підходить для візуалізації багатовимірних даних. t-SNE прагне зберегти локальну структуру даних, тобто розмістити близькі один до одного об'єкти в багатовимірному просторі також близько один до одного в двовимірному або тривимірному просторі. Це дозволяє візуалізувати кластери, виявляти аномалії та досліджувати структуру даних.

Ансамблеві методи

Ансамблеві методи (Ensemble Methods) – потужний підхід у машинному навчанні, який ґрунтується на ідеї об'єднання декількох простих моделей (базових алгоритмів) в одну складну, яка працює краще, ніж кожна з них окремо. Це схоже на те, як група експертів, об'єднавши свої знання та досвід, може прийняти більш зважене і точне рішення, ніж кожен з них поодинці.

Ансамблеві методи дозволяють підвищити точність і стійкість моделей, а також зменшити ризик перенавчання. Існує декілька основних підходів до побудови ансамблів:

- **Випадковий ліс (Random Forest):** будує ансамбль дерев рішень, кожне з яких навчається на випадковій підмножині навчальних даних і випадковій підмножині ознак. Усереднення прогнозів окремих дерев дозволяє отримати більш точну і стабільну модель. Випадковий ліс добре працює з різними типами даних, стійкий до шуму і викидів, а також дозволяє оцінювати важливість ознак.
- **Градієнтний бустинг (Gradient Boosting):** будує ансамбль послідовно, де кожна наступна модель навчається виправляти помилки попередніх моделей. На кожному кроці алгоритм обчислює градієнт функції втрат і навчає нову модель, яка апроксимує цей градієнт. Потім прогнози нової моделі додаються до прогнозів попередніх моделей з деяким коефіцієнтом (швидкістю навчання). Градієнтний бустинг – один із найпотужніших методів машинного навчання, який показує високу точність на багатьох задачах.
    - **XGBoost (Extreme Gradient Boosting):** оптимізована реалізація градієнтного бустингу, яка відрізняється високою швидкістю роботи і ефективністю. XGBoost використовує ряд алгоритмічних і системних оптимізацій, таких як регуляризація, обробка пропусків у даних, паралельне обчислення та кешування, що робить його одним із найпопулярніших інструментів для змагань з машинного навчання.

- **LightGBM (Light Gradient Boosting Machine):** ще одна ефективна реалізація градієнтного бустингу, розроблена компанією Microsoft. LightGBM використовує техніку градієнтного одностороннього семплювання (Gradient-based One-Side Sampling, GOSS) і ексклюзивне зв'язування ознак (Exclusive Feature Bundling, EFB) для прискорення навчання і зменшення споживання пам'яті.
- **CatBoost (Categorical Boosting):** розроблений компанією Яндекс. Його ключова особливість – ефективна обробка категоріальних ознак. CatBoost використовує впорядкований бустинг (Ordered Boosting) і інноваційні алгоритми для перетворення категоріальних ознак у числові, що дозволяє йому показувати високу точність на даних з великою кількістю категоріальних ознак.

- **Адаптивний бустинг (AdaBoost, Adaptive Boosting):** один із перших алгоритмів бустингу. Він навчає послідовність слабких класифікаторів (зазвичай дерев рішень невеликої глибини), причому кожен наступний класифікатор фокусується на тих прикладах, на яких попередні класифікатори помилялися. Приклади отримують ваги, які збільшуються, якщо класифікатор помиляється, і зменшуються, якщо класифікатор дає правильну відповідь. Таким чином, AdaBoost змушує наступні класифікатори звертати більше уваги на складні приклади.

Навчання з підкріпленням

Навчання з підкріпленням (Reinforcement Learning, RL) – область машинного навчання, де агент навчається взаємодіяти з середовищем, щоб максимізувати сумарну винагороду. Це схоже на те, як дресирують тварин, заохочуючи їх за правильні дії і караючи за неправильні.

На відміну від навчання з учителем, в RL немає заздалегідь заданих правильних відповідей. Агент повинен самостійно досліджувати середовище, методом проб і помилок, і знаходити оптимальну стратегію поведінки.

Основні поняття навчання з підкріпленням:

- **Агент (Agent):** той, хто навчається і приймає рішення.
- **Середовище (Environment):** те, з чим взаємодіє агент.
- **Стан (State):** поточна ситуація, в якій знаходиться агент.
- **Дія (Action):** те, що робить агент.
- **Винагорода (Reward):** зворотний зв'язок від середовища, який оцінює дії агента.
- **Стратегія (Policy):** правило, яке визначає, яку дію повинен вибрати агент у кожному стані.
- **Функція цінності (Value Function):** оцінює, наскільки добре перебувати в даному стані або виконувати дану дію.

- **Функція Q (Q-function):** оцінює очікувану сумарну винагороду, яку отримає агент, якщо він почне з даного стану, виконає дану дію і далі буде слідувати певній стратегії.

Алгоритми навчання з підкріпленням можна розділити на дві основні групи:

- **Методи, засновані на цінності (Value-Based Methods):** навчаються оцінювати функцію цінності або Q-функцію, а потім вибирають дії, які максимізують очікувану винагороду. Прикладом є алгоритм Q-навчання (Q-learning).
- **Методи, засновані на стратегії (Policy-Based Methods):** навчаються безпосередньо апроксимувати стратегію, яка відображає стани в дії. Прикладом є алгоритм REINFORCE.

Існують також гібридні методи, які поєднують в собі підходи, засновані на цінності і на стратегії, наприклад, Actor-Critic.

Навчання з підкріпленням знаходить застосування в різних областях, таких як:

- **Ігри:** навчання агентів для гри в шахи, Го, покер, відеоігри.
- **Робототехніка:** навчання роботів виконувати різні завдання, такі як ходьба, маніпулювання об’єктами, навігація.
- **Управління ресурсами:** оптимізація роботи систем, таких як центри обробки даних, транспортні мережі, енергосистеми.
- **Фінанси:** розробка торгових стратегій, управління портфелем інвестицій.

Неврівноважені набори даних

Неврівноважені набори даних (Imbalanced Datasets) – проблема, яка часто зустрічається в реальних задачах машинного навчання, коли кількість прикладів у різних класах значно відрізняється. Наприклад, у задачі виявлення шахрайських транзакцій кількість шахрайських транзакцій зазвичай набагато менша, ніж кількість нормальних.

Якщо навчати модель на таких даних без урахування дисбалансу, вона може навчитися добре передбачати домінуючий клас, але погано розпізнавати рідкісний клас, що може призвести до серйозних наслідків.

Існує декілька підходів до вирішення проблеми неврівноважених даних:

- **Перевибірка (Oversampling):** збільшення кількості прикладів у рідкісному класі шляхом їх дублювання або генерації нових прикладів на основі існуючих. Найпростіший метод – випадкова перевибірка (Random Oversampling), коли приклади з рідкісного класу просто дублюються. Більш просунуті методи, такі як SMOTE (Synthetic

Minority Over-sampling Technique), генерують нові приклади шляхом інтерполяції між існуючими прикладами рідкісного класу.

- **Підвибірка (Undersampling):** зменшення кількості прикладів у домінуючому класі шляхом видалення деяких з них. Найпростіший метод – випадкова підвибірка (Random Undersampling), коли приклади з домінуючого класу випадковим чином видаляються. Існують і більш інтелектуальні методи, які видаляють менш інформативні приклади.
- **Синтетичне створення даних (Synthetic Data Generation):** генерація нових прикладів для рідкісного класу з використанням різних технік, таких як SMOTE, згаданий вище, або генеративні змагальні мережі (GANs).

Вибір методу боротьби з дисбалансом класів залежить від конкретної задачі та характеристик даних. Важливо пам’ятати, що не існує універсального рішення, і кожен метод має свої переваги та недоліки.

Контрольні запитання

1. Що таке методи узагальнення і для чого вони використовуються? Які методи узагальнення ви знаєте?
2. Опишіть принцип роботи методу головних компонент (PCA). Як PCA використовується для зниження розмірності даних?
3. Що таке ансамблеві методи? У чому їхня перевага над одиночними моделями? Наведіть приклади ансамблевих методів.
4. Порівняйте ансамблеві методи: Random Forest, XGBoost, LightGBM, CatBoost, AdaBoost. Які їхні особливості та області застосування?
5. Що таке навчання з підкріпленням (Reinforcement Learning)? Опишіть основні принципи та алгоритми RL. Наведіть приклади застосування RL.

МОДУЛЬ 4. ГЕНЕРАТИВНИЙ ШТУЧНИЙ ІНТЕЛЕКТ

Лекція 10. Введення в генеративні моделі. Класифікація та основні концепції.

Вітаю на десятій лекції і першій у третьому модулі нашого курсу! Сьогодні ми ступаємо на захопливу територію генеративного штучного інтелекту – області, де машини не просто аналізують дані, а й створюють щось нове: зображення, музику, текст, і навіть цілі світи. Якщо раніше ми вчили алгоритми розпізнавати і класифікувати, то тепер ми навчимо їх творити.

Генеративні моделі – справжній прорив у сфері штучного інтелекту. Вони здатні не просто імітувати людську творчість, а й генерувати оригінальні ідеї, розширюючи межі можливого. Уявіть собі комп'ютер, який може написати симфонію, намалювати картину в стилі Ван Гога або створити реалістичне зображення людини, якої ніколи не існувало. Все перелічене вже не фантастика, а реальність завдяки генеративним моделям.

Від аналізу до творення

Протягом десятиліть штучний інтелект був зосереджений переважно на аналізі даних. Алгоритми машинного навчання навчилися розпізнавати об'єкти на зображеннях, класифікувати тексти, прогнозувати події та виявляти аномалії. Але всі ці завдання зводилися до обробки існуючої інформації.

Генеративні моделі знаменують собою якісно новий етап розвитку ШІ – перехід від аналізу до творення. Тепер машини здатні не просто пасивно сприймати інформацію, а й активно генерувати новий контент, який може бути не гіршим, а іноді й кращим за створений людиною.

Генеративні моделі: що це таке і як вони працюють?

Генеративна модель – алгоритм, який навчається створювати нові дані, схожі на ті, на яких він навчався. Наприклад, генеративна модель, навчена на фотографіях котів, може згенерувати нове зображення кота, якого ніколи не існувало, але який буде виглядати цілком реалістично.

В основі генеративних моделей лежить ідея вивчення розподілу ймовірностей навчальних даних. Уявіть, що у вас є набір точок на площині. Генеративна модель намагається зрозуміти, за яким законом ці точки розподілені, і навчитися генерувати нові точки, які будуть відповідати цьому закону.

Класифікація генеративних моделей

Існує безліч різних типів генеративних моделей, які можна класифікувати за кількома ознаками. Одна з найпоширеніших класифікацій ділить їх на дві основні групи:

- **Явні моделі (Explicit Models):** моделі, які явно визначають функцію щільності ймовірності даних. Вони дозволяють обчислювати ймовірність будь-якого заданого прикладу даних. До цієї групи належать такі моделі, як авторегресійні моделі (Autoregressive Models), потокові моделі (Flow-based Models) та варіаційні автоенкодери (Variational Autoencoders, VAE).
- **Неявні моделі (Implicit Models):** моделі, які не визначають явно функцію щільності ймовірності, а навчаються генерувати дані, імітуючи процес, що породив навчальні дані. До цієї групи належать генеративні змагальні мережі (Generative Adversarial Networks, GANs).

Інша класифікація поділяє генеративні моделі на основі того, як вони представляють дані:

- **Дискретні моделі:** працюють з дискретними даними, такими як текст або послідовності символів.
- **Неперервні моделі:** працюють з неперервними даними, такими як зображення, аудіо або відео.

Варіаційні автоенкодери (VAE)

Варіаційні автоенкодери (Variational Autoencoders, VAE) – тип генеративних моделей, який поєднує в собі ідеї автоенкодерів і варіаційного висновку (variational inference). VAE, як і звичайні автоенкодери, складаються з кодувальника (encoder) і декодувальника (decoder).

Кодувальник перетворює вхідні дані x в низьковимірне латентне представлення z, а декодувальник відновлює дані x з латентного представлення z. Однак, на відміну від звичайних автоенкодерів, VAE не просто відображають вхідні дані в латентний простір, а моделюють розподіл ймовірностей у цьому просторі.

Кодувальник VAE апроксимує апостеріорний розподіл $p(z|x)$ латентних змінних z за даними x. Зазвичай передбачається, що цей розподіл є гаусовим з діагональною коваріаційною матрицею. Декодувальник, навпаки, апроксимує правдоподібність $p(x|z)$ даних x за латентними змінними z. Навчання VAE полягає в максимізації варіаційної нижньої межі (variational lower bound) на логарифм правдоподібності даних.

Контрольні запитання

1. Що таке генеративні моделі і чим вони відрізняються від дискримінативних моделей? Наведіть приклади завдань, які вирішуються за допомогою генеративних моделей.
2. Як можна класифікувати генеративні моделі? Опишіть основні типи генеративних моделей та їхні характеристики.
3. Поясніть принцип роботи варіаційних автоенкодерів (VAE). Чим VAE відрізняються від звичайних автоенкодерів?
4. Що таке латентний простір і як він використовується в генеративних моделях? Як можна використовувати латентний простір для маніпулювання згенерованими даними?
5. У чому полягає основна відмінність між VAE та GANs з точки зору принципу роботи та сфери застосування?

Лекція 11. Генеративні змагальні нейронні мережі (GANs): принцип роботи, архітектури, застосування.

Архітектура GANs

В основі GANs лежить геніально проста ідея: протистояння двох нейронних мереж – генератора (Generator) і дискримінатора (Discriminator), які навчаються в процесі змагання одна з одною.

- **Генератор (G):** отримує на вході випадковий шум z і намагається перетворити його на дані, схожі на ті, на яких він навчається (наприклад, зображення, текст або музику). Його мета – створити такі дані, які дискримінатор не зможе відрізнити від справжніх.
- **Дискримінатор (D):** отримує на вході дані – або справжні з навчального набору, або згенеровані генератором – і намагається визначити, чи є вони справжніми чи підробленими. Його мета – навчитися максимально точно розрізняти реальні дані від згенерованих.

Уявіть собі, що генератор – фальшивомонетник, який намагається створити підроблені купюри, а дискримінатор – досвідчений експерт, який намагається відрізнити їх від справжніх. Фальшивомонетник намагається зробити свої підробки якомога якіснішими, а експерт, своєю чергою, стає все більш уважним до деталей, щоб викрити обман.

Механізм змагання: навчання через протистояння

Навчання GANs відбувається ітеративно, у вигляді гри з нульовою сумою, де виграш одного гравця дорівнює програшу іншого. На кожній ітерації генератор намагається створити дані, які обдурять дискримінатор, а дискримінатор намагається викрити обман.

Процес навчання можна описати так:

1. Генератор отримує випадковий шум z і створює зразок даних G(z).
2. Дискримінатор отримує як справжні дані x з навчального набору, так і згенеровані дані G(z), і видає ймовірність того, що дані є справжніми (D(x) для справжніх даних і D(G(z)) для згенерованих).
3. Ваги дискримінатора оновлюються таким чином, щоб максимізувати ймовірність правильної класифікації справжніх і згенерованих даних (тобто максимізувати D(x) і мінімізувати D(G(z))).
4. Ваги генератора оновлюються таким чином, щоб мінімізувати ймовірність того, що дискримінатор розпізнає згенеровані дані як підроблені (тобто максимізувати D(G(z))).

У процесі навчання обидві мережі стають сильнішими: генератор вчиться створювати все більш реалістичні дані, а дискримінатор – все краще відрізняти їх від справжніх. В ідеалі, навчання сходиться до рівноваги Неша, коли генератор створює дані, які дискримінатор не може відрізнити від справжніх, і обидві мережі перестають покращуватися.

Функція втрат GANs

Функція втрат GANs відображає змагальний характер навчання. Вона складається з двох частин:

- **Втрати дискримінатора:** заохочують дискримінатор правильно класифікувати справжні дані як справжні (D(x) близьке до 1), а згенеровані дані як підроблені (D(G(z)) близьке до 0).
- **Втрати генератора:** заохочують генератор створювати дані, які дискримінатор класифікує як справжні (D(G(z)) близьке до 1).

Основні типи GANs

З моменту появи оригінальної архітектури GANs було запропоновано безліч її модифікацій, спрямованих на поліпшення стабільності навчання, якості генерації та розширення можливостей застосування. Розглянемо деякі з найважливіших типів GANs:

- **Deep Convolutional GAN (DCGAN):** одна з найуспішніших і найпопулярніших архітектур GANs для генерації зображень. DCGAN використовує згорткові шари як у генераторі, так і в дискримінаторі, що дозволяє їй ефективно працювати з просторовою структурою зображень. У генераторі DCGAN зазвичай використовуються транспоновані згорткові шари (transposed convolutional layers) для збільшення розмірності даних від латентного простору до розміру зображення. Дискримінатор, навпаки, використовує згорткові шари для зменшення розмірності даних і виділення ознак, що дозволяють відрізнити справжні зображення від згенерованих. DCGAN зробила значний внесок у покращення якості та реалістичності згенерованих зображень.
- **Conditional GAN (cGAN):** розширення GANs, яке дозволяє контролювати процес генерації даних, задаючи додаткові умови. Наприклад, можна навчити cGAN генерувати зображення рукописної цифри, вказавши, яку саме цифру потрібно згенерувати. У cGAN як генератор, так і дискримінатор отримують на вході не тільки випадковий шум або дані, але й додаткову інформацію (умову), яка описує бажані характеристики згенерованих даних.
- **CycleGAN:** архітектура, призначена для перенесення стилю між різними доменами без необхідності мати парні приклади даних. Наприклад, CycleGAN може навчитися

перетворювати фотографії коней на зебр, фотографії літа на зиму, або картини Моне на фотографії. CycleGAN складається з двох генераторів і двох дискримінаторів, які утворюють два цикли: один для перетворення з домену A в домен B, а інший – навпаки.

- **StyleGAN:** одна з найпередовіших архітектур GANs, яка дозволяє генерувати дивовижно реалістичні зображення з високою роздільною здатністю і контролювати різні аспекти стилю зображення, такі як зачіска, вираз обличчя, освітлення та ін. StyleGAN досягає цього завдяки використанню прогресивного нарощування роздільної здатності зображення в процесі навчання, а також завдяки введенню механізму адаптивної інстанційної нормалізації (AdaIN) у генератор.

Застосування GANs

GANs знайшли широке застосування в різних областях, де потрібна генерація даних:

- **Генерація зображень:** створення реалістичних зображень людей, об'єктів, пейзажів та ін.
- **Редагування зображень:** зміна стилю, кольору, освітлення, додавання або видалення об'єктів на зображеннях.
- **Підвищення роздільної здатності зображень (Super-Resolution):** збільшення роздільної здатності зображень зі збереженням деталей.
- **Генерація відео:** створення реалістичних відеороликів, анімацій, спецефектів.
- **Генерація музики:** створення нових музичних композицій у різних жанрах.
- **Генерація тексту:** створення зв'язного і осмисленого тексту, наприклад, новинних статей, віршів, сценаріїв.
- **Розробка ігор:** створення ігрового контенту, такого як персонажі, рівні, текстури.
- **Дизайн:** створення нових дизайнів одягу, взуття, меблів, інтер'єрів.
- **Медицина:** генерація синтетичних медичних даних для навчання моделей машинного навчання, а також для розробки нових ліків і методів діагностики.
- **Наука:** моделювання фізичних процесів, генерація нових матеріалів з заданими властивостями.

Проблеми та виклики GANs

Незважаючи на вражаючі успіхи, GANs все ще стикаються з низкою проблем:

- **Нестабільність навчання:** навчання GANs може бути складним і нестабільним, часто виникають проблеми зі збіжністю, коли генератор і дискримінатор не можуть досягти рівноваги.

- **Колапс мод (Mode Collapse):** ситуація, коли генератор навчається створювати лише обмежений набір зразків даних, ігноруючи інші частини розподілу реальних даних.
- **Складність оцінки:** оцінка якості згенерованих даних може бути суб'єктивною і складною, особливо коли мова йде про такі дані, як зображення або музика.
- **Обчислювальна дорожнеча:** навчання GANs може вимагати значних обчислювальних ресурсів, особливо для генерації даних високої роздільної здатності.

Контрольні запитання

1. Опишіть архітектуру генеративних змагальних мереж (GANs). Які дві основні частини входять до складу GANs і які функції вони виконують?
2. Поясніть механізм змагання генератора та дискримінатора в GANs. Як відбувається навчання GANs?
3. Що таке функція втрат GANs і як вона пов'язана з процесом навчання генератора та дискримінатора?
4. Опишіть архітектуру Deep Convolutional GAN (DCGAN). Як вона використовує згорткові шари для генерації зображень?
5. Наведіть приклади практичного застосування GANs у різних сферах. Які досягнення та обмеження мають GANs у створенні реалістичних даних?

Лекція 12. Поліпшені архітектури GANs: Conditional GANs, CycleGAN, StyleGAN.

Conditional GANs

Уявіть, що ви можете не просто попросити GAN згенерувати зображення кота, а й вказати, якого саме кота ви хочете: його породу, колір шерсті, позу, оточення. Саме таку можливість надають Conditional GANs (cGANs).

Conditional GANs – розширення стандартної архітектури GANs, де як генератор, так і дискримінатор отримують додаткову інформацію – умову (condition), яка описує бажані характеристики згенерованих даних. Ця умова може бути представлена у вигляді мітки класу (наприклад, «кіт», «собака»), текстового опису (наприклад, «рудий кіт, що сидить на траві») або навіть іншого зображення.

Як це працює?

У cGANs умова подається на вхід як генератора, так і дискримінатора. Генератор використовує умову разом із випадковим шумом, щоб створити дані, які відповідають заданій умові. Дискримінатор, своєю чергою, повинен не тільки відрізнити справжні дані від згенерованих, а й перевірити, чи відповідають згенеровані дані заданій умові.

Приклад:

Припустимо, ми хочемо навчити cGAN генерувати зображення рукописних цифр. Як умову ми можемо використовувати мітку класу, яка вказує, яку цифру потрібно згенерувати. У цьому випадку генератор отримує на вході випадковий шум і мітку класу (наприклад, «3»), і намагається створити зображення рукописної цифри 3. Дискримінатор отримує на вході зображення (справжнє або згенероване) і мітку класу, і повинен визначити, чи є це зображення справжньою рукописною цифрою, що відповідає заданій мітці.

Застосування cGANs:

- **Генерація зображень за текстовим описом:** створення зображень на основі текстових описів, наприклад, «чоловік у капелюсі йде по вулиці».
- **Редагування зображень:** зміна окремих атрибутів зображення, наприклад, кольору волосся, виразу обличчя, віку людини на фотографії.
- **Розфарбовування зображень:** перетворення чорно-білих зображень на кольорові.
- **Переклад зображень (Image-to-Image Translation):** перетворення зображень з одного домену в інший, наприклад, перетворення ескізу на фотореалістичне зображення, карти Google на супутникові знімки.

CycleGAN

Уявіть, що ви можете перетворити фотографію коня на зебру, зберігши при цьому його позу та оточення. Або перетворити літній пейзаж на зимовий, не змінюючи композицію. Саме це стає можливим завдяки CycleGAN – архітектурі, яка здійснює дива перенесення стилю між різними доменами.

На відміну від cGANs, CycleGAN не потребує парних прикладів даних для навчання. Тобто, вам не потрібно мати набір даних, де кожному зображенню коня відповідає точно таке ж зображення зебри. CycleGAN може навчитися перетворювати коней на зебр, маючи просто набір зображень коней і набір зображень зебр.

Як це працює?

CycleGAN складається з двох генераторів і двох дискримінаторів, які утворюють два цикли:

1. **Цикл A -> B -> A:**

- Генератор G_A_to_B перетворює зображення з домену A (наприклад, коні) в домен B (наприклад, зебри).
- Дискримінатор D_B намагається відрізнити справжні зображення з домену B від згенерованих G_A_to_B(A).
- Генератор G_B_to_A перетворює згенероване зображення G_A_to_B(A) назад у домен A.
- Втрата циклічності (cycle-consistency loss) гарантує, що G_B_to_A(G_A_to_B(A)) буде близьким до оригінального зображення A.

2. **Цикл B -> A -> B:**

- Генератор G_B_to_A перетворює зображення з домену B (наприклад, зебри) в домен A (наприклад, коні).
- Дискримінатор D_A намагається відрізнити справжні зображення з домену A від згенерованих G_B_to_A(B).
- Генератор G_A_to_B перетворює згенероване зображення G_B_to_A(B) назад у домен B.
- Втрата циклічності гарантує, що G_A_to_B(G_B_to_A(B)) буде близьким до оригінального зображення B.

Така циклічна структура навчання дозволяє CycleGAN вивчити відображення між двома доменами без необхідності мати парні приклади даних.

Застосування CycleGAN:

- **Перенесення стилю:** перетворення фотографій на картини в стилі різних художників, зміна пори року на фотографіях, перетворення коней на зебр, яблук на апельсини тощо.
- **Покращення якості зображень:** видалення шуму, підвищення роздільної здатності, відновлення пошкоджених зображень.
- **Збільшення даних (Data Augmentation):** створення нових навчальних даних шляхом перетворення існуючих, що може бути корисним для навчання моделей машинного навчання в умовах обмеженої кількості даних.

StyleGAN: генерація з високою роздільною здатністю та контролем стилю

StyleGAN – одна з найпередовіших і найвражаючих архітектур GANs на сьогоднішній день. Вона дозволяє генерувати дивовижно реалістичні зображення з високою роздільною здатністю (наприклад, 1024x1024 пікселів і вище) і надає безпрецедентний контроль над різними аспектами стилю згенерованих зображень.

Як це працює?

StyleGAN має ряд ключових відмінностей від попередніх архітектур GANs:

- **Прогресивне нарощування (Progressive Growing):** навчання починається з генерації зображень дуже низької роздільної здатності (наприклад, 4x4 пікселів), а потім поступово збільшує роздільну здатність, додаючи нові шари до генератора і дискримінатора. Такий підхід робить навчання більш стабільним і дозволяє генерувати зображення високої якості.
- **Відображення з латентного простору в проміжний простір W (Mapping Network):** замість того, щоб подавати випадковий шум z безпосередньо в генератор, StyleGAN спочатку перетворює його в проміжний вектор w за допомогою багатошарової нейронної мережі (mapping network). Цей проміжний вектор потім використовується для контролю стилю на різних рівнях генератора.
- **Адаптивна інстанційна нормалізація (Adaptive Instance Normalization, AdaIN):** StyleGAN використовує AdaIN для впровадження стилю, закодованого в проміжному векторі w, у процес генерації зображення. AdaIN дозволяє контролювати такі аспекти стилю, як зачіска, вираз обличчя, освітлення, колірна гама та ін., на різних рівнях деталізації.
- **Додавання шуму (Noise Inputs):** StyleGAN додає шум на різних рівнях генератора, що сприяє генерації дрібних деталей і текстур, а також підвищує різноманітність згенерованих зображень.

Застосування StyleGAN:

- **Генерація реалістичних зображень облич:** StyleGAN особливо відомий своєю здатністю генерувати дивовижно реалістичні зображення людських облич, які важко відрізнити від справжніх фотографій.
- **Створення та редагування персонажів:** StyleGAN дозволяє створювати нових персонажів для ігор, фільмів та анімації, а також редагувати існуючих, змінюючи їхні атрибути.
- **Дизайн:** StyleGAN може використовуватися для генерації нових дизайнів інтер'єрів, одягу, автомобілів та інших об'єктів.
- **Мистецтво:** StyleGAN стає все більш популярним інструментом серед художників, які використовують його для створення нових форм цифрового мистецтва.

Контрольні запитання

1. Що таке Conditional GANs (cGANs) і чим вони відрізняються від звичайних GANs? Як cGANs використовують умови для контролю процесу генерації?
2. Наведіть приклади практичного застосування cGANs. Як можна використовувати cGANs для генерації зображень за текстовим описом?
3. Поясніть принцип роботи CycleGAN. Як ця архітектура дозволяє перетворювати стилі між зображеннями без пар відповідностей?
4. Що таке втрата циклічності (cycle-consistency loss) і яку роль вона відіграє в навчанні CycleGAN? Наведіть приклади застосування CycleGAN.
5. Опишіть основні відмінності архітектури StyleGAN від попередніх GANs. Як StyleGAN використовує прогресивне нарощування, відображення з латентного простору та адаптивну інстанційну нормалізацію для генерації високоякісних зображень?

МОДУЛЬ 5. ПРАКТИЧНІ ЗАСТОСУВАННЯ ГЕНЕРАТИВНОГО ШТУЧНОГО ІНТЕЛЕКТУ

Лекція 13. Генеративні моделі для обробки тексту: GPT, BERT, трансформери.

Від рекурентних мереж до трансформерів

До недавнього часу в NLP панували рекурентні нейронні мережі (RNN), особливо LSTM (Long Short-Term Memory). RNN ідеально підходили для обробки послідовностей, оскільки вони враховують попередній контекст при обробці кожного наступного слова. Однак, RNN мають і свої недоліки: вони навчаються повільно, особливо на довгих послідовностях, і погано розпаралелюються, оскільки обробляють текст послідовно, слово за словом.

У 2017 році група дослідників з Google представила архітектуру трансформер, яка стала справжнім проривом в NLP. Трансформери відмовилися від рекурентності на користь механізму уваги (attention), що дозволило їм обробляти текст паралельно і значно прискорити навчання.

Трансформер

В основі трансформера лежить механізм уваги, який дозволяє моделі фокусуватися на найбільш важливих частинах вхідної послідовності при обробці кожного слова. Уявіть, що ви читаєте речення і намагаєтеся зрозуміти його зміст. Ви не просто читаєте слова одне за одним, а й звертаєте особливу увагу на ключові слова, які несуть основне смислове навантаження. Механізм уваги в трансформерах працює схожим чином.

Архітектура трансформера складається з двох основних частин:

- **Кодувальник (Encoder):** обробляє вхідну послідовність (наприклад, речення) і перетворює її на набір векторних представлень (embeddings), які містять інформацію про кожне слово і його контекст.
- **Декодувальник (Decoder):** генерує вихідну послідовність (наприклад, переклад речення іншою мовою), використовуючи векторні представлення, отримані від кодувальника, і механізм уваги.

Ключові компоненти трансформера:

- **Механізм уваги (Attention):** дозволяє моделі визначати, на які частини вхідної послідовності слід звернути особливу увагу при обробці кожного слова. Існує кілька

різновидів механізму уваги, але найпоширенішим є масштабований точковий добуток уваги (Scaled Dot-Product Attention).

- **Багатоголова увага (Multi-Head Attention):** замість одного механізму уваги, трансформер використовує декілька паралельних механізмів уваги (голів), кожен з яких навчається фокусуватися на різних аспектах вхідної послідовності. Це дозволяє моделі вловлювати складніші залежності між словами.
- **Позиційне кодування (Positional Encoding):** оскільки трансформер не використовує рекурентності, йому потрібно якось передати інформацію про порядок слів у послідовності. Для цього до векторних представлень слів додаються спеціальні вектори позиційного кодування, які несуть інформацію про позицію слова в реченні.
- **Пряме поширення (Feedforward):** кожен шар кодувальника і декодувальника містить повнозв'язну нейронну мережу прямого поширення, яка застосовується до кожного векторного представлення окремо.
- **Нормалізація шару (Layer Normalization):** застосовується після кожного механізму уваги і повнозв'язної мережі для стабілізації навчання.
- **Залишкові з'єднання (Residual Connections):** дозволяють градієнтам легше поширюватися мережею під час навчання, що сприяє більш ефективному навчанню глибоких трансформерів.

GPT: генеративний попередньо навчений трансформер

Однією з найвідоміших моделей, заснованих на архітектурі трансформер, є GPT (Generative Pre-trained Transformer), розроблена компанією OpenAI. GPT – модель, навчена генерувати текст, передбачаючи наступне слово в послідовності.

Як працює GPT?

GPT використовує тільки декодувальну частину трансформера. Модель отримує на вході послідовність слів і намагається передбачити наступне слово. Для цього вона використовує механізм уваги, щоб врахувати контекст попередніх слів.

Навчання GPT проходить у два етапи:

1. **Попереднє навчання (Pre-training):** GPT навчається на величезному корпусі текстових даних (наприклад, на всьому Інтернеті) без розмітки. Модель вчиться передбачати наступне слово в реченні, що дозволяє їй вивчити загальні закономірності мови, граматику, семантику і навіть деякі факти про світ.

2. **Тонке налаштування (Fine-tuning):** попередньо навчена модель донавчається на меншому наборі даних, специфічному для конкретного завдання, наприклад, для генерації новинних статей, відповідей на питання або машинного перекладу.

Застосування GPT:

- **Генерація тексту:** GPT може генерувати зв'язний, осмислений і часто невідрізний від написаного людиною текст на різні теми.
- **Відповіді на питання:** GPT може відповідати на питання, ґрунтуючись на інформації, яку вона вивчила під час попереднього навчання.
- **Машинний переклад:** GPT може перекладати текст з однієї мови на іншу.
- **Створення чат-ботів:** GPT може використовуватися для створення чат-ботів, здатних вести осмислені діалоги з користувачами.
- **Написання коду:** GPT може генерувати код на різних мовах програмування.

BERT: двонаправлений кодувальник на основі трансформера

Інша революційна модель, заснована на трансформерах, – BERT (Bidirectional Encoder Representations from Transformers), розроблена Google. На відміну від GPT, BERT використовує тільки кодувальну частину трансформера і навчається не передбачати наступне слово, а відновлювати пропущені слова в реченні.

Як працює BERT?

BERT навчається за допомогою двох основних завдань:

1. **Маскування мовної моделі (Masked Language Model, MLM):** у вхідній послідовності випадковим чином маскуються деякі слова (замінюються на спеціальний токен [MASK]). Модель повинна передбачити, які слова були замасковані, ґрунтуючись на контексті.
2. **Передбачення наступного речення (Next Sentence Prediction, NSP):** моделі даються два речення, і вона повинна визначити, чи є друге речення логічним продовженням першого.

Завдяки двонаправленій архітектурі і завданню MLM, BERT вчиться вловлювати контекст слова з обох боків, що дозволяє йому отримувати більш якісні векторні представлення слів, ніж однонаправлені моделі, такі як GPT.

Навчання BERT також проходить у два етапи:

1. **Попереднє навчання:** BERT навчається на величезному корпусі текстових даних (наприклад, на Вікіпедії та книгах) без розмітки, вирішуючи завдання MLM і NSP.
2. **Тонке налаштування:** попередньо навчена модель донавчається на меншому наборі даних, специфічному для конкретного завдання, наприклад, для класифікації тексту, розпізнавання іменованих сутностей або відповідей на питання.

Застосування BERT:

- **Класифікація тексту:** BERT може використовуватися для класифікації тексту за різними категоріями, наприклад, для визначення тональності тексту (позитивна, негативна, нейтральна), тематики новин або спам-фільтрації.
- **Розпізнавання іменованих сутностей (Named Entity Recognition, NER):** BERT може виділяти в тексті іменовані сутності, такі як імена людей, назви організацій, географічні назви тощо.
- **Відповіді на питання:** BERT може відповідати на питання, ґрунтуючись на заданому тексті.
- **Машинний переклад:** BERT може використовуватися для покращення якості машинного перекладу.
- **Пошукові системи:** BERT значно покращив якість роботи пошукових систем, дозволяючи їм краще розуміти зміст запитів і видавати більш релевантні результати.

Трансформери в інших областях:

Архітектура трансформер виявилася настільки успішною, що її почали застосовувати і в інших областях, крім NLP:

- **Комп'ютерний зір:** Vision Transformer (ViT) використовує трансформери для обробки зображень, досягаючи результатів, порівнянних з кращими згортковими нейронними мережами.
- **Обробка аудіо:** трансформери застосовуються для розпізнавання мови, генерації музики та інших завдань, пов'язаних з обробкою аудіосигналів.

Контрольні запитання

1. Які основні проблеми виникали при використанні рекурентних нейронних мереж (RNN) для обробки тексту? Як архітектура трансформер вирішує ці проблеми?

2. Опишіть основні компоненти архітектури трансформер: механізм уваги, багатоголова увага, позиційне кодування.
3. Що таке GPT (Generative Pre-trained Transformer)? Як відбувається попереднє навчання та тонке налаштування GPT?
4. Що таке BERT (Bidirectional Encoder Representations from Transformers)? Чим BERT відрізняється від GPT? Як BERT використовує маскування мовної моделі для навчання?
5. Наведіть приклади застосування GPT та BERT у різних задачах обробки природної мови (NLP).

Лекція 14. Генеративні моделі для обробки зображень: DeepDream, нейронні стилізації.

DeepDream

DeepDream – алгоритм, розроблений інженерами Google у 2015 році, який став справжньою сенсацією в інтернеті, породивши безліч химерних і сюрреалістичних зображень. DeepDream, по суті, є інверсією звичайної нейронної мережі, натренованої для розпізнавання об'єктів.

Як це працює?

Зазвичай, коли ми подаємо зображення на вхід нейронної мережі, вона активує певні нейрони, що відповідають за розпізнавання тих чи інших об'єктів. Наприклад, якщо мережа бачить зображення собаки, активуються нейрони, пов'язані з розпізнаванням собак, шерсті, вух, хвоста тощо.

DeepDream працює навпаки. Замість того, щоб розпізнавати об'єкти, ми беремо довільне зображення і змушуємо мережу посилювати ті нейрони, які вже активувалися. У результаті мережа починає «бачити» в зображенні те, чого там насправді немає, або гіперболізувати існуючі деталі, перетворюючи їх на химерні образи.

Процес роботи DeepDream можна описати так:

1. Вибираємо зображення і подаємо його на вхід попередньо навченій нейронній мережі (наприклад, Inception, яка добре вміє розпізнавати об'єкти).
2. Вибираємо шар мережі, активації якого ми хочемо посилити. Зазвичай, вищі шари відповідають за розпізнавання складніших об'єктів, тому їх активація призводить до появи в зображенні впізнаваних образів.
3. Обчислюємо градієнт зображення по відношенню до активацій обраного шару.
4. Модифікуємо зображення в напрямку градієнта, тобто посилюємо ті його частини, які найбільше активують нейрони обраного шару.
5. Повторюємо кроки 3-4 кілька разів, поступово трансформуючи зображення.

Що бачить DeepDream?

Результати роботи DeepDream часто нагадують галюцинації або сновидіння. У звичайних хмарах мережа може побачити химерних тварин, у листі дерев – очі та обличчя, а в текстурі каменю – цілі архітектурні ансамблі.

Інтерпретувати результати DeepDream непросто. Можна сказати, що алгоритм візуалізує те, що «бачить» нейронна мережа, коли дивиться на зображення. Він показує нам, які ознаки та патерни є найбільш значущими для мережі, і як вона інтерпретує світ.

Застосування DeepDream:

- **Мистецтво:** DeepDream став популярним інструментом серед художників, які використовують його для створення сюрреалістичних і психоделічних зображень.
- **Дослідження нейронних мереж:** DeepDream дозволяє дослідникам краще зрозуміти, як працюють нейронні мережі, як вони кодують інформацію і які ознаки є для них найбільш важливими.
- **Візуалізація даних:** DeepDream може використовуватися для візуалізації даних, представлених у вигляді зображень, наприклад, медичних знімків або карт активності мозку.

Нейронні стилізації

Якщо DeepDream дозволяє нам зазирнути в «підсвідомість» нейронної мережі, то нейронні стилізації (Neural Style Transfer) дають можливість перенести стиль одного зображення на інше, створюючи справжні шедеври машинного мистецтва.

Уявіть, що ви можете взяти фотографію вашого улюбленого міста і перетворити її на картину в стилі Ван Гога, зберегти зміст фотографії, але надати їй неповторної естетики великого постімпресіоніста. Саме це і роблять нейронні стилізації.

Як це працює?

В основі нейронних стилізацій лежить ідея розділення змісту і стилю зображення. Зміст – це об’єкти, зображені на картині, їхнє розташування, загальна композиція. Стиль – це колірна гама, текстура, манера мазка, характерні деталі, які роблять стиль художника впізнаваним.

Нейронні стилізації використовують згорткові нейронні мережі (CNN) для виділення ознак, що відповідають за зміст і стиль зображення. Зазвичай, для цього використовуються попередньо навчені мережі, такі як VGG, які добре вміють розпізнавати об’єкти.

Процес роботи алгоритму нейронної стилізації можна описати так:

1. Вибираємо два зображення: зображення-контент (content image), з якого ми хочемо взяти зміст, і зображення-стиль (style image), з якого ми хочемо взяти стиль.
2. Подаємо обидва зображення на вхід попередньо навченій CNN і отримуємо активації різних шарів мережі.

3. Визначаємо, які шари мережі відповідають за зміст, а які – за стиль. Зазвичай, нижчі шари кодують інформацію про деталі і текстури (стиль), а вищі – про загальну композицію і об’єкти (зміст).
4. Визначаємо функцію втрат, яка складається з двох частин: втрати змісту (content loss) і втрати стилю (style loss). Втрата змісту вимірює, наскільки згенероване зображення відрізняється від зображення-контенту за активаціями «змістовних» шарів. Втрата стилю вимірює, наскільки згенероване зображення відрізняється від зображення-стилю за активаціями «стильових» шарів. Для обчислення втрати стилю зазвичай використовується матриця Грама, яка описує кореляції між різними ознаками на одному шарі.
5. Генеруємо нове зображення, яке мінімізує функцію втрат. Зазвичай, цей процес починається з випадкового шуму і ітеративно оновлюється зображення в напрямку, протилежному градієнту функції втрат.

Застосування нейронних стилізацій:

- **Мистецтво:** нейронні стилізації стали популярним інструментом серед художників, які використовують їх для створення нових творів мистецтва, стилізації фотографій і відео, а також для дослідження різних художніх стилів.
- **Дизайн:** нейронні стилізації можуть використовуватися для створення унікальних графічних ефектів, дизайну інтер’єрів, одягу та інших об’єктів.
- **Мобільні додатки:** існує безліч мобільних додатків, які дозволяють користувачам застосовувати нейронні стилізації до своїх фотографій і відео в реальному часі.
- **Кіноіндустрія:** нейронні стилізації можуть використовуватися для створення спецефектів, стилізації сцен, а також для розфарбовування чорно-білих фільмів.

Обмеження та подальший розвиток:

Незважаючи на вражаючі результати, нейронні стилізації все ще мають деякі обмеження:

- **Обчислювальна складність:** процес генерації зображення може бути досить повільним, особливо для зображень високої роздільної здатності.
- **Якість стилізації:** не всі стилі однаково добре переносяться, а іноді згенеровані зображення можуть містити артефакти або виглядати неприродно.
- **Контроль над процесом:** наразі можливості контролю над процесом стилізації обмежені. Користувач може вибирати зображення-контент і зображення-стиль, але не може точно вказати, які саме аспекти стилю потрібно перенести.

Дослідження в галузі нейронних стилізацій активно розвиваються. Вчені працюють над створенням більш швидких і ефективних алгоритмів, покращенням якості стилізації, а також над розширенням можливостей контролю над процесом генерації.

Контрольні запитання

1. Поясніть принцип роботи алгоритму DeepDream. Як він використовує попередньо навчену нейронну мережу для візуалізації її «снів»?
2. Що таке нейронні стилізації? Як вони дозволяють переносити стиль одного зображення на інше?
3. Опишіть процес роботи алгоритму нейронної стилізації. Як визначаються та використовуються функції втрати змісту та стилю?
4. Які практичні застосування мають DeepDream та нейронні стилізації у мистецтві, дизайні та інших сферах?
5. Які обмеження мають алгоритми нейронних стилізацій? Як ведуться роботи з їхнього вдосконалення?

Лекція 15. Гібридні нейронні мережі: інтеграція класичних і сучасних методів.

Гібридні нейронні мережі

Гібридні нейронні мережі – системи, які комбінують різні типи нейронних мереж та/або класичні алгоритми машинного навчання для вирішення складних завдань. Ідея полягає в тому, щоб використовувати сильні сторони кожного компонента і компенсувати їхні слабкі сторони, досягаючи таким чином кращої продуктивності, ніж при використанні кожного компонента окремо.

Уявіть собі оркестр, де різні інструменти, кожен зі своїм унікальним тембром і можливостями, грають разом, створюючи гармонійне і багате звучання. Так само і в гібридних нейронних мережах: різні алгоритми та архітектури взаємодіють один з одним, вирішуючи різні аспекти завдання і досягаючи спільної мети.

Переваги гібридного підходу:

- **Підвищена точність і ефективність:** комбінування різних методів дозволяє досягти кращої точності та ефективності у вирішенні завдань, ніж при використанні одного методу.
- **Гнучкість і адаптивність:** гібридні системи можуть бути адаптовані до різних типів даних і завдань, використовуючи найбільш підходящі компоненти для кожного конкретного випадку.
- **Обробка складних даних:** гібридні мережі можуть ефективно обробляти дані зі складною структурою, які важко піддаються аналізу традиційними методами.
- **Інтерпретованість:** деякі гібридні моделі, що включають класичні алгоритми, можуть бути більш інтерпретованими, ніж «чисті» глибокі нейронні мережі, що полегшує розуміння їхньої роботи і прийнятих ними рішень.

Приклади архітектур гібридних нейронних мереж:

- **Поєднання згорткових і рекурентних мереж:** ця комбінація часто використовується для обробки відео, де згорткові мережі витягують просторові ознаки з кожного кадру, а рекурентні мережі моделюють часову залежність між кадрами. Наприклад, така архітектура може бути використана для розпізнавання дій на відео.
- **Ансамблі дерев рішень і нейронних мереж:** дерева рішень можуть бути використані для попередньої обробки даних і виділення ознак, які потім подаються на вхід нейронної мережі. Такий підхід може підвищити точність і швидкість навчання, особливо на табличних даних.

- **Інтеграція нейронних мереж з алгоритмами кластеризації:** наприклад, K-Means може використовуватися для кластеризації даних, а потім кожен кластер може бути оброблений окремою нейронною мережею.
- **Гібридні GANs:** поєднання різних типів GANs або інтеграція GANs з іншими генеративними моделями, такими як VAE, для поліпшення якості та стабільності генерації.
- **Використання класичних методів для обробки виходів нейронних мереж:** наприклад, застосування методу опорних векторів (SVM) до ознак, витягнутих з останнього шару глибокої нейронної мережі, для вирішення завдання класифікації.

Практичне застосування гібридних нейронних мереж:

1. **Обробка тексту:**

- **Визначення тональності тексту:** гібридна модель може комбінувати рекурентну нейронну мережу (LSTM або GRU) для аналізу послідовності слів у реченні з класичними методами машинного навчання, такими як метод опорних векторів (SVM) або наївний баєсівський класифікатор, для визначення емоційного забарвлення тексту (позитивне, негативне, нейтральне).
- **Машинний переклад:** поєднання трансформерів з іншими методами, такими як статистичний машинний переклад, може покращити якість перекладу, особливо для рідкісних мов або специфічних доменів.
- **Генерація тексту:** гібридні моделі можуть комбінувати мовні моделі на основі трансформерів (наприклад, GPT) з алгоритмами пошуку за променем (beam search) або іншими методами оптимізації для генерації більш зв’язного і релевантного тексту.

2. **Обробка зображень:**

- **Розпізнавання об’єктів:** гібридна модель може використовувати згорткову нейронну мережу для виділення ознак із зображення, а потім застосовувати класичний алгоритм машинного навчання, такий як випадковий ліс, для класифікації об’єктів на основі цих ознак.
- **Сегментація зображень:** поєднання згорткових мереж з умовними випадковими полями (Conditional Random Fields, CRFs) може покращити точність сегментації зображень, враховуючи як локальні ознаки, так і глобальний контекст.
- **Генерація зображень:** гібридні GANs можуть комбінувати різні типи GANs (наприклад, DCGAN і StyleGAN) або інтегрувати GANs з VAE для створення більш реалістичних і різноманітних зображень.

3. **Обробка аудіо:**

- **Розпізнавання мови:** гібридні моделі можуть поєднувати згорткові та рекурентні нейронні мережі для обробки аудіосигналу з методами прихованих марковських моделей (Hidden Markov Models, HMMs) для моделювання послідовності фонем.
- **Визначення емоцій у мовленні:** глибокі нейронні мережі можуть витягувати ознаки з аудіосигналу, а потім класичні алгоритми машинного навчання, такі як SVM, можуть використовуватися для класифікації емоційного стану мовця.
- **Генерація музики:** гібридні моделі можуть комбінувати рекурентні мережі, такі як LSTM, для генерації послідовності нот з GANs для створення більш реалістичного і різноманітного музичного звучання.

4. **Обробка відео:**

- **Розпізнавання дій:** гібридні моделі можуть використовувати згорткові мережі для аналізу кожного кадру відео, а рекурентні мережі – для моделювання часової залежності між кадрами, що дозволяє розпізнавати складні дії, такі як «гра на гітарі» або «приготування їжі».
- **Відстеження об'єктів:** поєднання згорткових мереж для виділення ознак об'єктів з алгоритмами фільтрації Калмана або фільтра частинок для відстеження їхнього руху в часі.
- **Генерація відео:** гібридні GANs можуть використовуватися для створення реалістичних відеороликів, комбінуючи згорткові мережі для обробки просторових ознак з рекурентними мережами для моделювання часової динаміки.

5. **Медицина:**

- **Діагностика захворювань:** гібридні моделі можуть аналізувати медичні зображення (наприклад, рентгенівські знімки, МРТ) за допомогою згорткових мереж, а потім використовувати клінічні дані пацієнта (наприклад, вік, стать, історію хвороби) та класичні алгоритми машинного навчання для прогнозування ймовірності захворювання.
- **Прогнозування результатів лікування:** поєднання глибоких нейронних мереж, що обробляють геномні дані, з деревами рішень, що аналізують клінічні дані, може допомогти передбачити ефективність лікування для конкретного пацієнта.
- **Розробка ліків:** гібридні моделі можуть використовуватися для ідентифікації потенційних цілей для ліків, прогнозування їхньої ефективності та безпеки, а також для оптимізації процесу розробки нових препаратів.

6. **Фінанси:**

- **Виявлення шахрайства:** гібридні моделі можуть комбінувати алгоритми виявлення аномалій, такі як ізолюючий ліс (Isolation Forest), з глибокими нейронними мережами, які аналізують послідовності транзакцій, для виявлення шахрайських операцій.
- **Оцінка кредитного ризику:** поєднання класичних методів скорингу з нейронними мережами, які обробляють текстові дані (наприклад, відгуки клієнтів, новини), може підвищити точність оцінки кредитоспроможності позичальників.
- **Алгоритмічна торгівля:** гібридні моделі можуть використовувати рекурентні нейронні мережі для прогнозування часових рядів фінансових інструментів і класичні методи машинного навчання для розробки торгових стратегій на основі цих прогнозів.

Виклики та перспективи:

Розробка гібридних нейронних мереж – складне завдання, яке вимагає глибокого розуміння як класичних алгоритмів машинного навчання, так і сучасних нейромережевих архітектур. Потрібно ретельно підбирати компоненти гібридної моделі, враховуючи їхні сильні та слабкі сторони, а також особливості конкретного завдання і даних.

Одним із викликів є визначення оптимального способу інтеграції різних компонентів. Існує безліч варіантів: послідовне з'єднання, паралельне з'єднання, використання одного компонента для попередньої обробки даних, а іншого – для основного завдання, і так далі. Вибір оптимальної архітектури часто вимагає експериментування і тонкого налаштування.

Інший виклик – інтерпретованість гібридних моделей. Чим складніша модель, тим важче зрозуміти, як вона приймає рішення. Це може бути проблемою в областях, де важлива прозорість і зрозумілість процесу прийняття рішень, наприклад, у медицині та фінансах.

Незважаючи на ці виклики, гібридні нейронні мережі є дуже перспективним напрямом досліджень. Вони дозволяють поєднувати найкращі риси різних підходів до машинного навчання, досягаючи високої точності, ефективності та адаптивності.

Майбутнє гібридних нейронних мереж:

Можна очікувати, що в майбутньому гібридні нейронні мережі будуть ставати все більш складними та досконалими. З'являться нові способи інтеграції різних алгоритмів і архітектур, будуть розроблені нові методи навчання та оптимізації гібридних моделей.

Одним із перспективних напрямів є розробка гібридних моделей, які поєднують у собі сильні сторони різних типів генеративних моделей, таких як VAE, GANs і трансформери. Це дозволить створювати ще більш реалістичні та різноманітні дані, а також відкриє нові можливості для творчості та інновацій.

Інший напрямок – інтеграція нейронних мереж з іншими підходами до штучного інтелекту, такими як символьні обчислення, логічне програмування та еволюційні алгоритми. Це дозволить створювати ще більш потужні та гнучкі системи штучного інтелекту, здатні вирішувати найскладніші завдання.

Контрольні запитання

1. Що таке гібридні нейронні мережі? У чому полягає основна ідея їхнього створення? Наведіть приклади поєднання різних алгоритмів у рамках гібридної моделі.
2. Які переваги надає використання гібридних нейронних мереж порівняно з використанням «чистих» класичних алгоритмів або «чистих» глибоких нейронних мереж?
3. Опишіть приклади архітектур гібридних нейронних мереж. Як можна комбінувати згорткові та рекурентні мережі для обробки відео? Як можна поєднати дерева рішень і нейронні мережі?
4. Наведіть приклади застосування гібридних нейронних мереж у різних областях: обробка тексту, зображень, аудіо, відео. Які компоненти (алгоритми/мережі) використовуються в цих гібридних моделях?
5. Які основні виклики та проблеми виникають при розробці та навчанні гібридних нейронних мереж? Як можна визначити оптимальний спосіб інтеграції різних компонентів?

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

Основна література

1. Кононова К. Ю. Машинне навчання: методи та моделі : підручник. Харків : ХНУ імені В. Н. Каразіна, 2020. 301 с.
2. Gallatin, K. & Albon, C. Machine Learning with Python Cookbook. Practical Solutions from Preprocessing to Deep Learning. O'Reilly Media, Inc. 2023. 416 p.
3. Howard, J., & Gugger, S. Deep Learning for Coders with fastai & PyTorch. AI Applications Without a PhD. O'Reilly Media, Inc. 2020. 624 p.
4. Huyen, C. Designing Machine Learning Systems. An Iterative Process for Production-Ready Applications. O'Reilly Media, Inc. 2022. 389 p.
5. Lakshmanan, V., Görner, M., & Gillard, R. Practical Machine Learning for Computer Vision. End to-End Machine Learning for images.2021. 481 p.
6. Moroney, L. AI and Machine Learning for Coders: A Programmer's Guide to Artificial Intelligence. O'Reilly Media, Inc. 2021. 392 p.

Додаткова література

1. Булгакова О.С., Зосімов В.В., Поздєєв В.О. Методи та системи штучного інтелекту:теорія та практика. Навчальний посібник. Одеса : ОЛДІ-ПЛЮС, 2020. 356 с.
2. Гавриленко С.Ю., Челак В.В., Горносталь О.А., Зозуля В.Д. Машинне навчання. Лабораторний практикум / С.Ю. Гавриленко, В.В. Челак, О.А. Горносталь, В.Д Зозуля. Х.: НТУ «ХПІ», 2022.86 с.
3. Методи та системи штучного інтелекту: навч. посіб. / укл. Д.В. Лубко, С.В. Шаров. Мелітополь: ФОП Однорог Т.В., 2019. 264 с.
4. Субботін С.О. Нейронні мережі: навч. посібник / С.О. Субботін, А.О. Олійник; за ред. С.О. Субботіна. Запоріжжя : ЗНТУ, 2014. 132 с.
5. Тимощук П.В. Штучні нейронні мережі. Навчальний посібник / Львів: Видавництво Львівської Політехніки, 2011. 444 с.
6. Федорін І. В. Методи та технології обчислювального інтелекту: Практикум : навч. посіб. КПІ ім. Ігоря Сікорського. Київ: КПІ ім. Ігоря Сікорського, 2022. 317 с.
7. Шаповал Н. В. Методи та системи штучного інтелекту. Комп'ютерний практикум : навч. посіб. Київ : КПІ ім. І. Сікорського, 2022. 45 с.
8. Charu C. Aggarwal Neural Networks and Deep Learning. Springer International Publishing. 2023. 529 p.
9. Dulani Meedeniya Deep Learning. CRC Press LLC, 2023. 200 p.
10. Foster, D. Generative Deep Learning. O'Reilly Media, Inc. 2023. 456 p.
11. Hapke, H., & Nerlson, C. Building Machine Learning Pipelines. Automating Model Life Cycles

with TensorFlow. O'Reilly Media, Inc. 2020. 367 p.
12. Huyen, C. Designing Machine Learning Systems. An Iterative Process for Production-Ready Applications. O'Reilly Media, Inc. 2022. 389 p.
13. Machine Learning and Its Application to Reacting Flows : ML and Combustion / N. Swaminathan, A. Parente (eds.). Cham : Springer, 2023. 346 p.
14. Zgurovsky M. Z., Zaychenko Y. P. The Fundamentals of Computational Intelligence: System Approach. Springer International Publishing Switzerland, 2016. 375 p.