ВСТУП

Даний курс присвячено питанням машинного навчання, типовим задачам, які можуть бути вирішені його методами, доступним бібліотекам машинного навчання. Більше того, курс дозволить зрозуміти, як, маючи уявлення про роботу алгоритму, можна використовувати готові реалізації для вирішення прикладних задач.

Навчання – це річ, знайома будь якій людині. Люди навчаються щодня і показують у цьому процесі прекрасні результати. Спостерігаючи закономірності в зміні середовища навколо, вони конструюють певну модель зміни цього середовища і приймають ті чи інші рішення. Середовище певним чином реагує на прийняті рішення і люди знову корегують модель світу.

Машинне навчання – це спрощена версія процесу навчання, яке відбувається з людиною. Як правило, в машинному навчанні наявний певний набір прикладів, спостережень, реакцій до цих спостережень. Задача полягає у тому, щоб сконструювати такі моделі, які будуть максимально ефективно описувати наявні дані і робити достовірні прогнози.

Такими прогнозами можуть бути відповіді на питання: чи є котик на зображенні? Чи є сенс зараз купляти акції певної компанії? Це позитивний чи негативний відгук на товар в Інтернеті?

Існує безліч сфер застосування машинного навчання. Така розповсюдженість стала наслідком розвитку Інтернету і накопичення величезної кількості даних, а також тим, що широкому загалу стали доступні великі обчислювальні потужності.

Взагалі, сфера машинного навчання розвивається понад 60 років і тривалий час вона була сферою досліджень суто науковців. Нині вирішувати задачі за допомогою алгоритмів машинного навчання може будь-яка людина, яка засвоїть логіку роботи алгоритму, ознайомиться із доступними реалізаціями та вивчить їх особливості.

Даний курс для спрощення розуміння не буде заглиблюватись у велику кількість формул. Пріоритетна задача: зрозуміти, що саме вирішують алгоритми та які основні тези їхньої роботи.

В даному курсі буде розглянуто, які основні типи алгоритмів машинного навчання існують, які задачі, що таке регресія, кластеризація, класифікація, чим вони відрізняються. Також буде розказано про основи роботи нейронних

мереж, типи їх архітектури, рекурентні нейронні мережі та нейронні мережі зі згорткою, їх відмінності, реалізації, доступні для швидкого застосування на практиці.

Після цього курсу Ви матимете змогу вирішувати задачу розпізнавання образів на зображенні, задачу сентимент-аналізу (визначення тональності відгуку, наприклад, на фільм).

Даний курс націлений на ознайомлення з алгоритмами і спонукання до більш глибокого ознайомлення із кожною із реалізацій і застосування їх на практиці.

Машинне навчання. Типи навчання. Індуктивне та дедуктивне навчання. Типи машинного навчання: навчання з вчителем, навчання без вчителя, навчання з підкріпленням. Задачі машинного навчання: кластеризації, регресії, зниження розмірності.

Що таке машинне навчання? При всій загадковості, в даній сфері немає нічого складного, оскільки навчання знайоме кожній людині з дитинства.

Машинне навчання відрізняється від звичного нам тим, що ми намагаємося навчити комп’ютер вчитися.

Існує 2 типи навчання:

1. Індуктивне навчання.
2. Дедуктивне навчання.

Індуктивне навчання знайоме кожному, адже воно полягає у спостереженні за світом та побудові певних моделей, які пояснюють причини тих чи інших явищ.

Потім такі моделі неодноразово перевіряються, певні з них «виживають» і використовуються, покращуються. А деякі моделі згодом цілком відкидаються.

Дедуктивне навчання подібне до математики в школі, коли учню дають готові формули і розказують, як застосовувати їх на практиці.

Машинне навчання є індуктивним навчанням, оскільки в основному ми вчимо машину вчитися на прикладах, спостерігати велику кількість прикладів із реального життя, будувати на них моделі, перевіряти їх, застосовувати їх на подальших прикладах.

Машинне навчання (МН)– це підрозділ штучного інтелекту, який розглядає побудову алгоритмів, які можуть навчатися на наявних даних.

Задача МН виглядає так: уявімо собі, що в нас є певний набір об'єктів-прикладів і певний набір міток, тобто, реакцій, відповідей. Між прикладами/спостереженнями і відповідями є певна прихована залежність. **Задача МН – знайти** цю приховану залежність для прогнозування відповідей на основі нових даних.

МН поділяється на **3 типи**:

1) **Навчання з вчителем**

Є набір прикладів, до кожного прикладу є правильна відповідь. Задача системи – навчитися по прикладах надавати правильну відповідь, задану вчителем. Вчителями є ми.

2) **Навчання без вчителя**

Є великий набір даних. В цих даних є приховані закономірності. Задача системи – знайти закономірності, наприклад, розбивши дані на певні групи чи кластери.

3) **Навчання з підкріпленням**

Є певне середовище, в якому є певний агент, що контролюється комп'ютером. Агент може вчиняти певні дії. Певні дії приводять до позитивних відкликів чи негативних відкликів. Задача – максимізувати позитивні і мінімізувати негативні відклики. Приклад: гра, в якій треба максимізувати набрані очки, або виграти усю гру.

Задачі МН класифікуються ще на декілька **типів** по виду вирішуваної проблеми:

- **Задача кластеризації**

Всім відомий приклад, коли комп'ютеру потрібно навчитися розпізнавати котиків і собачок в наборі фотографій. Задача комп'ютера – навчитися розпізнавати по наданій фотографії – це фото котика чи собачки. Задача

кластеризації виглядає досить подібно, але дещо відрізняється. Припустимо, у нас є ті самі фотографії, але ми не кажемо комп'ютеру, які фото є котиків, а які – собачок. Ми потребуємо від алгоритму, щоб він **сам** знайшов закономірності серед фото і розбив їх на декілька кластерів. В залежності від того, наскільки різноманітні дані, кількість кластерів може варіюватись. Ми самі можемо задавати кількість кластерів і наказувати алгоритму розбити дані, наприклад, на 2 кластери. Якщо алгоритм досить ефективний – він знайде дві сукупності фотографій – окремо котиків та окремо собачок.

- **Задача регресії**

Є набір точок, через які треба провести досить рівну лінію, яка б проходила досить близько до цих точок. Для людини це досить проста задача, але коли ми маємо справу із багатовимірними даними, а не двовимірними, то все стає не так очевидно.

- **Задача зниження розмірності**

Є дані, представлені в дуже багатовимірній репрезентації, наприклад, результати соціологічного опитування на тисячу питань для сотень тисяч опитаних. Відповідно, кожна людина у нас репрезентується тисячовимірним вектором. Ми хочемо в ньому знайти приховані закономірності. Застосувавши алгоритм, який вирішує задачу зменшення вимірності, ми можемо знизити вимірність з 1000 вимірів до 3 вимірів. І ми побачимо, що одні резюме знаходяться в одній зоні, а інші – в інших. Надалі можна аналізувати ці дані або сконструювати наступний алгоритм МН, який дістане більш цінні дані із нашої тестувальної вибірки.

Застосовувати МН можна в стількох сферах, що легше перерахувати сфери, в яких МН незастосовне. МН можна застосовувати для розпізнавання об'єктів на зображеннях у кінофільмах, прогнозуванням курсу акцій, знаходження на знімку ракових пухлин ще до того, як лікар проаналізував знімок, знаходження генетичних відхилень по коду ДНК та ін. У багатьох сферах МН застосовують з кращим ефектом, ніж це роблять люди.

Етапи розробки моделі машинного навчання: представлення даних (в т.ч. ознакове описання об’єкту, вибір ключових ознак), конструювання алгоритму (вибір типу задачі), тренування та тестування.

Процес розробки певної моделі МН складається із етапів:

- процес підготовки (представлення) даних
- процес конструювання алгоритму
- процес тренування алгоритму на наявних даних
- процес валідації алгоритму на тестових даних

Якщо модель показала себе досить добре на тестових даних, ми застосовуємо її на практиці.

Першим етапом є процес **представлення даних.**

Якість представлення даних в МН є одним із ключових параметрів ефективної роботи моделі МН.

Залежно від того, яка у нас прикладна задача, процес представлення даних для алгоритму відрізнятиметься. Для вирішення різних задач одні методи представлення даних підходять, інші – ні і ніколи не приведуть до якісної функціонуючої моделі.

Одним із найпоширеніших методів представлення даних у МН є **ознакове описання об’єкту**. Уявімо собі результати соціологічного дослідження: є приклади – люди, і кожна людина відповіла на певний набір питань. Кожне питання – це певна ознака: вік, колір волосся, рівень доходу, рівень освіти. Цих ознак може бути одна-дві, або сотні і тисячі.

Якість роботи алгоритму сильно залежить від того, наскільки ознаки, які ми даємо йому на вхід, справді впливають на мітку, яку ми отримуємо на виході. Якщо у нас є задача прогнозування, чи купить певний клієнт товар Х і ми на вхід даємо дані – колір волосся, шкіри, національність, кількість дітей в сім’ї – цілком можливо, існує якась прихована закономірність між кольором шкіри та волосся і тим, чи купить даний клієнт товар.

Але, очевидно, що модель працювала б краще, якби ми дали на вхід такі дані: рівень доходу, рівень освіти, хобі, професія.

Отож, першим етапом конструювання нашої ефективної моделі МН є **вибір ключових ознак**. І це етап, в якому рішення в основному залежить від

людини/архітектора системи, який, проаналізувавши задачу, робить висновок, що одні ознаки підходять для прогнозування того, що ми хочемо отримати на виході системи, а інші – ні.

Іноді для певних задач дані представляються не у вигляді ознакового описання, а у вигляді т.зв. **«сирих» даних.**

Наприклад, картинка – це ознакове описання, але ознакове описання не того, що зображено, а кожного пікселя окремо. Якщо у нас є кольорова картинка 100 на 100 пікселів RGB, це означає, що у нас є 100x100x3 об'єктів, для кожного з яких є певна ознака – колір. Таке представлення даних виокремлюють в окрему групу і називають «сирими» даними. Методи роботи з ними – відрізняються.

Таким чином, одним із перших етапів роботи в МН для створення певної моделі є представлення даних для комп'ютера в такому вигляді, щоб він їх зрозумів.

Другий етап: визначення типу задачі (конструювання алгоритму). Є декілька типів задач в МН: класифікація, кластеризація, регресія, ранжування або знаходження аномалій.

В залежності від того, яка прикладна задача, потрібно вибрати метод, яким ми будемо її вирішувати.

Наприклад, якщо задача *класифікація*, а ми намагатимемось її вирішити методом *ранжування*, то зрозуміло, що якість нашої роботи низькою.

Третій і четвертий етапи – вибір правильних параметрів і підготування тренувальної вибірки. Ми розділяємо всі наші дані на тренувальну та тестувальну вибірки. На тренувальній ми тренуємо систему. Наприклад, якщо це нейронна мережа, ми ітерація за ітерацією оновлюємо ваги (див. далі) мережі, а коли вона досягнула певного рівня стабільних відповідей, ми її тестуємо на тестувальній вибірці.

Перенавчання. Тренувальна та тестувальна вибірки на прикладі нейронної мережі.

Поговоримо про неприємну річ в машинному навчанні – **перенавчання.**

Одним із критеріїв якості роботи будь-якої системи МН є якість відповіді на тестувальній вибірці.

Чим тестувальна вибірка відрізняється від тренувальної вибірки?

Тренувальна вибірка – це набір прикладів, які ми показуємо системі, щоб вона на їх базі викрила певну приховану закономірність, яка відповідальна за розподіл даних в тренувальній вибірці. Завдяки віднайденою такої закономірності система зможе її використати для ефективного прогнозування відповідей на тестувальній вибірці.

Якісна модель на тестувальній вибірці дає досить близькі результати до міток. Погана модель може показувати дуже хороші результати на тренувальній вибірці (відповіді будуть один в один співпадати з прикладами, які ми їй надавали), але на тестувальній вибірці результати будуть не такими хорошими.

Уявімо типову задачу регресії – є сто тисяч точок у двох вимірах, на площині. Ми даємо системі ці точки і її завдання – побудувати криву, яка б проходила через усі ці точки або максимально наближено до них. Якісна модель спрогнозує відповіді досить близько до тестувальної вибірки. Погана ж видасть результат, що сильно відрізнятиметься. Чому так стається?

Припустимо, що ми розглядаємо такий алгоритм машинного навчання, як нейронні мережі.

Нейронна мережа (більш детально про них поговоримо у наступних лекціях) – це певна комплексна функція, яка має величезну кількість параметрів. Кожен із цих параметрів (т.зв. **ваги**) адаптується для того, щоб наблизити функцію до стану, як розподілені дані у тестувальній вибірці. Дуже часто кількість цих параметрів набагато більша, ніж необхідно опису тестових даних.

Згадаємо графіки функцій – парабола або гіпербола. Що таке парабола? Це крива, яка виникає внаслідок закономірності $y=x^2$.

Припустимо, ми тренуємо мережу, задача якої вивчити, що таке парабола. Ми даємо їй на вхід точки: 2-4, 4-16, -2-4 і т.д. Але, якщо ми тренуємо

нейронну мережу, кількість параметрів буде набагато більша, ніж x чи 2x. Вона може намалювати нам графік набагато складніший, ніж парабола. Якщо кількість наших прикладів не дуже велика, наприклад, ми їй дали лише 100 точок, які лежать на кривій $y=x^2$, можливо, вона побудує криву, яка проходить через всі ці точки, які були в тренувальній вибірці, але це буде не парабола, а, наприклад, крива, що осцилює з великою частотою вгору-вниз, однак проходить через всі точки, що були в тренувальній вибірці.

Отже, якщо ми дамо на вхід точки із тестувальної вибірки, ми побачимо, що значення, які спрогнозує наша система, дуже сильно відрізнятимуться від актуальних даних.

Таке явище називається **перенавчанням.**

Причинами цього явища є **недостатня кількість прикладів**. Система будує дуже узагальнені чи дуже часткові моделі, але не генералізує закономірність, яка лежить за цими прикладами. Якщо прикладів достатньо – сто тисяч, мільйон – імовірність того, що система побудує криву, що пройде через усі ці точки, і ця крива буде дуже сильно відрізнятися від реальної закономірності, що за ними лежить, буде дуже низькою. Це означає, що наша система зменшує долю ймовірності перенавчання.

Одним із ключових параметрів, що впливають на перенавчання системи –є різноманітність та кількість прикладів, а також нюанси архітектури системи.

Також перенавчання системи залежить від того, як ми обрали тренувальну вибірку, які ознаки ми обрали для описання об’єктів у тренувальній вибірці, яка кількість прикладів і наскільки вони різноманітні. Чим більше вони відрізняються один від одного і чим їх більше – тим краще.

Взагалі, в машинному навчанні 80% успіху – це розмір Вашої тренувальної вибірки, її різноманітність і якість параметрів, які Ви обрали для описання кожного із об’єктів вибірки.

Типи ознак (бінарні, номінальні, порядкові, кількісні). Функціонал якості, квадратична та абсолютна помилки

Поговоримо про описання об'єктів та прикладів для нашої системи машинного навчання.

Система машинного навчання вчиться на прикладах для того, щоб їх генералізувати, тобто знайти приховані закономірності між вхідними та вихідними даними так, щоб на тестовій вибірці демонструвати хороші результати. Якість результатів досить сильно залежить від того, які ознаки ми оберемо для описання і яким чином ми їх будемо описувати.

Є декілька способів описання цих ознак.

Наприклад, є **бінарні ознаки**: якщо у нас є певні характеристики, які можуть бути описаними відповіддю на питання так або ні, є певна характеристика або немає, 1 або 0, значить ми можемо їх описувати певним бінарним вектором. Цих ознак може бути декілька, тобто, наприклад, у об'єкта може бути 10 або 100 000 ознак в залежності від прикладної задачі.

Номінальні ознаки: один із прикладів – це координати точки, які можуть бути від нуля до плюс нескінченності в залежності від того, який у нас тип даних. Координати точки (-5, 16) – це будуть дві номінальні ознаки -x та y.

Порядкові ознаки – це ознака позиції, наприклад, в списку задачі ранжування.

Кількісні ознаки – припустимо, описання кількості доходу фізичної особи або зріст, або будь-які інші властивості, які можна описати кількісною характеристикою.

Нейронні мережі, як один із видів машинного навчання, на вхід завжди отримують певний **вектор ознак**, на виході надають **вектор відповідей.** Якщо говорити про нейронні мережі, слід пам'ятати, що все, що робить нейронна мережа, це *деформує багатовимірний простір* ознак таким чином, щоб внаслідок всіх деформацій приклади наблизились до очікуваних зон у вихідному просторі.

Розмірність простору, в якому ми задаємо ознаки, як правило відрізняється від розмірності простору, в якому ми очікуємо відповіді. Припустимо, простір відповідей може бути одновимірний або двовимірний в залежності від задачі.

Окрім описання об'єкту ми також описуємо відповіді, тобто мітки, які ми ставимо до об'єктів, які хочемо отримати внаслідок роботи системи. Мітки також можуть бути бінарні, тобто 1 або 0, наявна або відсутня, це можуть бути індекси класів, до яких належить об'єкт.

Задачі класифікації теж між собою дещо відрізняються, оскільки існують класифікації на класи, які між собою не перетинаються, наприклад, класифікація «котики чи собачки». Або це може бути класифікація на набір класів, які перетинаються, наприклад, класифікація наявних об'єктів на зображенні. Одночасно можуть бути в наявності кілька різних об'єктів і фотографії можуть належати до декількох класів одночасно.

В задачах регресії міткою може виступати будь-який номінальний вектор. Наприклад, ми можемо сконструювати функцію, яка одну двовимірну точку проектує на виході у якусь іншу двовимірну точку.

В задачах ранжування кінцевою метою є впорядкована послідовність результатів, які ми отримаємо.

В процесі навчання будь-якої системи машинного навчання ключовим моментом у досягненні бажаного результату є вибір правильного **функціоналу якості**.

Функціонал якості – це певна функція, яка видає нам рівень помилки, яку робить система. В процесі навчання моделі ми намагаємося мінімізувати помилку, яку видає система. Тобто, у нас є набір прикладів, набір очікуваних результатів. Система спочатку видає результати, які дуже відрізняються від того що ми очікуємо. Відповідно по кожному із прикладів ми можемо вирахувати помилку і скорегувати систему таким чином, щоб помилка була менше. В залежності від того, яку функцію втрат функціоналу якості ми виберемо, буде залежати яким чином буде навчатися наша система.

Функціонал якості в залежності від задачі відрізняється. Наприклад, для задачі регресії досить доцільне використання так званої функції **квадратичної помилки,** коли ми віднімаємо від очікуваного результату результат, який видала система і беремо квадрат цієї різниці. Також можна обрати **абсолютну помилку**, наприклад, в задачі мультикласової класифікації функцією помилки може бути – віднесла система до правильного класу чи не віднесла. Тобто 1 або 0.

Мінімізація помилки. Приклад нейронних мереж. Зміна сили зв’язків.

Яким чином відбувається процес мінімізації помилки?

В машинному навчанні весь процес навчання, як правило, зводиться до мінімізації помилки. Тобто, якщо наша система мінімізувала помилку до досить низької величини і показує такі ж самі хороші результати не тільки на

тренувальній вибірці, але й на тестовій, ми вважаємо, що її можна використовувати.

Одним із прикладів, як відбувається процес мінімізації помилки, є такий вид машинного навчання як **нейронні мережі.**

Нейронні мережі займаються поетапним проектування вхідних векторів таким чином, щоб на виході отримати правильний вектор очікуваної розмірності.

Якщо зрозуміти, як працюють між собою всього-на-всього два нейрони, можна зрозуміти, що відбувається в масштабі всієї нейронної мережі. Кожен нейрон має зв'язки зі своїми сусідами. У процесі навчання, **сила цих зв'язків** може корегуватися – посилюватися або послаблюватися. Якщо сила зв'язку велика, це означає, що стан сусіда сильно залежить від стану нейрона у попередньому шарі. Якщо сила зв'язку мала, це значить, що стан сусіда не сильно залежить від того, наскільки активований поточний нейрон.

На початку тренування мережі, як правило, сили зв'язків між всіма нейронами обираються випадковими. В процесі тренування, кожного разу, коли ми проходимо і отримуємо новий і новий вектор, на виході ми отримуємо певну помилку, яку ми отримали внаслідок застосування функції втрат, наприклад, квадратичної помилки. Далі система проходиться в зворотному напрямку і корегує силу зв'язків між нейронами таким чином, щоб на наступній ітерації помилка була меншою. Якщо в нашій тренувальний вибірці різноманітність прикладів достатньо велика, то нейрони корегують між собою таким чином, щоб на виході ми могли отримати результати максимально наближені до очікуваних.

Лінійна регресія. Градієнтний спуск.

Одним із популярних методів вирішення типових задач в машинному навчанні, що відомий не тільки спеціалістам у машинному навчанні, а й в економіці, є **лінійна регресія.**

Лінійна регресія допомагає вирішити такі типові задачі. Уявімо собі, що ми маємо певний набір спостережень, і маємо координати кожного спостереження, тобто маємо вектор описання кожного із прикладів. Для простоти – нехай наші спостереження – це певні двовимірні точки (з координатами X та Y) і вони певним чином розміщені в просторі.

Припустимо, що ми очікуємо, що точки розміщенні невипадково, а у них є певна закономірність. Припустимо, вони зростають, коли ми рухаємося у

напрямку «плюс нескінченність». Оскільки ми споглядаємо на окремі точки, то чітко сказати, якій закономірності вони слідують –важко. Але відповідь на це питання досить часто можна знайти за допомогою лінійної регресії.

Задача зводиться до знаходження прямої, яка буде мінімально віддалена до кожної точки в нашій тренувальній вибірці. Оскільки ми знаходимо певну пряму – звідси і назва «лінійна регресія».

З шкільного курсу відомо, що пряма на площині задається рівнянням y=kx+b, де k – задає кут нахилу, а b – відповідає за зміщення відносно центру нашої системи координат.

Якщо ми хочемо знайти таку пряму, ми маємо знайти такі коефіцієнти k і b, які дадуть мінімальну відстань до кожної точки. k і b – це конкретні координати точки в просторі, де за вісь ординат відповідає k, а за вісь абсцис – b. Фактично, задача зводиться до того, щоб на площині знайти точку, яка буде відповідати прямій, що має мінімальну відстань до кожної нашої точки.

Яким методом її можна шукати?

Один із методів – **градієнтний спуск**. Оскільки ми розглядаємо цілу площину можливих прямих, кожна із точок набору координат k і b є певною потенційною прямою. Кожна із цих прямих матиме певний набір відстаней до наших точок, а, значить, ми можемо вирахувати, наприклад, середню відстань і отримаємо карту відстаней до точок на площині k і b.

Нам потрібно знайти таку точку, яка буде відповідати мінімуму даної «теплової» карти. Фактично, ми спостерігаємо поверхню, в якій нам потрібно знайти найглибшу точку, тобто глобальний мінімум.

Тобто, в методі градієнтного спуску нам необхідно максимально швидко з випадкових стартових конфігурацій спуститися і знайти координати найглибшої точки на даній площині.

Детально про градієнтний спуск ми поговоримо у одній із наступних лекцій.

Але, по суті, ми шукаємо спосіб, як максимально швидко спуститися з вершини гори до найглибшого підніжжя. Причому, слід обрати такий метод, щоб з великою імовірністю ми спустилися не в локальний мінімум, а в найглибшу точку.

В машинному навчанні – застрягання у локальному мінімумі є серйозною проблемою на рівні проблеми перенавчання. Для вирішення цієї проблеми існують свої методи.

Отже, лінійна регресія, шукаючи координати точки на площині k,b видає відповідну їй пряму, яка мінімально віддалена до точок, які ми надали в тренувальній вибірці. Як правило, дуже рідко буває, щоб точки знаходилися вздовж однієї прямої і відстань від прямої до точок була нульовою.

Якщо ми хочемо писати більш комплексну функцію, є ускладнені методи лінійної регресії або **логістичної регресії**, про що ми поговоримо далі.

Логістична регресія.

В попередні рази ми ознайомились з базовим уявленням про лінійну регресію і в яких задачах її можна використовувати.
Зараз визначимо, що таке логістична регресія.
Нагадаємо, що лінійну регресію можна застосувати у випадках, коли ми хочемо знайти певну лінійну закономірність між вхідними стимулами і ознаками (певними x та y). Гіпотеза полягає в тому, що дана закономірність є неявною.
А як бути в ситуації, коли простір ознак – лінійний, а простір відповідей – не лінійний, тобто може бути набором класів?
У випадку з лінійною регресією все зрозуміло. Припустимо, з тих прикладів, що ми розглядали: є певний рівень доходу на сім'ю і певний рівень витрат. Рівень доходів – конкретне число. Рівень витрат – конкретне число в певному діапазоні.
У випадку логістичної регресії ознаки можуть бути певним звичайним числом, а реакція може бути класом – або наявністю, або відсутністю певної дії.
Припустимо, у нас є набір даних про рівень доходу покупця товару і є дані про те, купила дана людина товар чи ні.
Відповідно, наші ознаки – рівень доходу, конкретне число в діапазоні від 0 до +нескінченності, а відповіді – це два класи, 1 (купила) та 0 (не купила).
Задача – побудувати систему, яка може прогнозувати, з якою імовірністю людина купить чи не купить товар.
Ми не можемо сказати однозначну відповідь, а можемо дати певну статистичну оцінку ймовірності.
Ми беремо нашу навчальну вибірку – кількість рівнів доходу і бінарні ознаки.
Фактично ми хочемо розділити простір прикладів певною площиною так, щоб, знаючи відстань конкретного прикладу до цієї площини з того чи

іншого боку ми могли дати статистичну оцінку, з якою імовірністю до якого із класів належить конкретний приклад.
Що оптимізує логістична регресія?
Формат відповідей у випадку з бінарною логістичною регресією буде 1 або 0. Знайшовши вирішення логістичної регресії, ми будемо мати певну площину, конкретні точки прикладів, на яких ми навчались, залежно від того одиниці це або нулі, мають знаходитись по різні боки від площини.
Потім, тестуючи нашу систему на тестовій вибірці, взявши певний приклад, ми зможемо сказати, з якою імовірністю до якого класу він належить.
Ймовірність буде коливатися в діапазоні від 0 до 1.
Логістична регресія досить подібна до лінійної регресії тим, що це знаходження певного набору параметрів.
Але відрізняється методом видачі результатів і тим, в який простір ми проектуємо наші вхідні точки.
До речі, логістична регресія іноді називається сигмоїдом. Ми знаходимо не пряму, а логістичну криву, яка є максимально гладкою біля 1 або 0 і плавно переходить в районі 0,5.

Навчання логістичної регресії. Софт-макс регресія.

Як буде виглядати процес навчання логістичної регресії? З одного боку у нас є набір прикладів, з іншого – набір правильних класифікацій кожного з прикладів і навчаючи нашу логістичну регресію ми намагаємося мінімізувати помилку в класифікації, тобто максимізувати ймовірність правильної класифікації.

Яким чином?

Один із методів – метод мінімізації емпіричного ризику. Тобто ми намагаємось знайти такий набір коефіцієнтів, яким відповідає певна площина, що розділяє наші приклади, щоб вона була влаштована таким чином, що ймовірність неправильної класифікації була мінімальна. Шукати коефіцієнти вирішення даного рівняння можна, наприклад, методом градієнтного спуску, коли у нас є певний простір коефіцієнтів і ми, рухаючись ітерація за ітерацією в напрямку зменшення помилки (а у випадку логістичної регресії – у напрямку максимізації правильної класифікації), крок за кроком наближаємось до глобального мінімуму функції, якій відповідає дана логістична регресія. В результаті отримуємо певний вектор коефіцієнтів, який при підставлянні в рівняння логістичної регресії максимізує правильність класифікації для випадкової точки.

Також результати логістичної регресії можна інтерпретувати як відстані до межі кластерів. В нас є два класи, будь-яка із точок відповідно належить до одного із двох кластерів. Припустимо, на тестувальній вибірці ми можемо знайти певну точку, яка знаходиться десь посередині певного кластера. Відповідно, наша логістична регресія, якщо вона навчена, повинна видати в вигляді розподілу ймовірності – з якою ймовірністю до якого із класів наш приклад є наближений.

Ми намагаємось уникнути ситуації, коли ситуація видає відповідь 50 на 50. Ми намагаємося максимізувати випадки, коли можна сказати, що даний приклад із ймовірністю, припустимо, 99% належить до одного із класів.

В процесі навчання логістичної регресії ми намагаємося уникнути неоднозначностей, тобто коли розподіл практично однорідний, і прагнемо максимізувати випадки асиметричного розподілу.

У випадку з логістичною регресією ми можемо розширити загальний принцип на більшу кількість класів. Такий вид логістичної регресії називають софт-макс регресією. Принцип в основі – той же самий, єдина різниця – ми робимо прогноз належності даного прикладу не до одного з двох класів, а до більшої кількості.

Наша відповідь буде розподілом ймовірностей по кожному із класів. В сумі вони завжди даватимуть 1.

Але різні класи матимуть різну частину від цієї одиниці розподілу.

Софт-макс регресії і софт-макс активація досить широко застосовується в задачах нейронних мереж і мультикласифікації на багато класів, коли ми знаємо, що кожен із наших прикладів належить до одного із багатьох класів.

Отже, основна відмінність лінійної і логістичної регресії – це різні сфери застосування і вирішувані задачі. Логістична – вирішує задачу класифікації, лінійна – знаходження лінійної кореляції між двома чи більшою кількістю змінних.

Логістична регресія намагається оптимізувати функцію, яка видає нам імовірність того, що ми правильно класифікували приклад.