**Лекція 2. Регресія**

1. **Поняття лінійної регресії**
2. **Моделі з багатьма ознаками**
3. **Основні алгоритми машинного навчання для задач регресії**
4. **Лінійна регресія: втрата**
5. **Лінійна регресія: гіперпараметри**
6. **Використання нейронних мереж та бібліотек Python для рішення задачі регресії.**
7. **Поняття лінійної регресії**

**Лінійна регресія** — це статистичний метод, що використовується для знаходження взаємозв'язку між змінними. В контексті машинного навчання лінійна регресія знаходить зв'язок між ознаками та міткою.

Наприклад, припустимо, що ми хочемо спрогнозувати витрати пального автомобіля в милях на галон на основі ваги автомобіля, і у нас є наступний набір даних: Фунти в тисячах (ознака) | Миль на галон

| **Фунти в тисячах (ознака)** | **Міль на галон (мітка)** |
| --- | --- |
| 3,5 | 18 |
| 3,69 | 15 |
| 3.44 | 18 |
| 3.43 | 16 |
| 4.34 | 15 |
| 4.42 | 14 |
| 2.37 | 24 |



Рис. 1 Вага автомобіля (в фунтах) залежно від оцінки в милях на галон. По мірі того, як автомобіль стає важчим, його рейтинг в милях на галон зазвичай зменшується.

Ми могли б створити свою власну модель, провівши лінію найкращого відповідності через точки:



**Рівняння лінійної регресії в алгебраїчних термінах** модель буде визначатися як

, де

y— це милі на галон — значення, яке ми хочемо спрогнозувати.

m– нахил лінії.

x— це фунти — наше вхідне значення.

b— це точка перетину з віссю y.

**У ML ми пишемо рівняння для моделі лінійної регресії** наступним чином:

де:

y’— це передбачена мітка — результат.

b— зсув моделі. Зсув — це те ж поняття, що і точка перетину з віссю y в алгебраїчному рівнянні прямої. В ML зсув іноді називають w0. Зсув є параметром моделі і розраховується під час навчання.

w1— це вага функції. Вага – це те ж поняття, що і нахил в алгебраїчному рівнянні прямої. Вага є параметром моделі і розраховується під час навчання.

x1 — це вхід.

Під час навчання модель обчислює вагу та зсув, які створюють найкращу модель.



В нашому прикладі ми б обчислили вагу та зсув по проведеній лінії. Зсув дорівнює 30 (де лінія перетинає вісь Y), а вага дорівнює -3,6 (нахил лінії). Модель буде визначена як , і ми зможемо використовувати її для прогнозування. Наприклад, використовуючи цю модель, прогнозована економія пального автомобіля вагою 4000 фунтів становитиме 15,6 миль на галон.



1. **Моделі з багатьма ознаками**

Хоча в прикладі в цьому розділі використовується тільки одна ознака — вага автомобіля, складніша модель може базуватися на кількох ознаках, кожна з яких має окрему вагу

Модель, що прогнозує витрату пального, може також використовувати такі ознаки, як:

* Об'єм двигуна
* Прискорення
* Кількість циліндрів
* Кінські сили



Зображаючи деякі з цих додаткових характеристик, ми бачимо, що вони також мають лінійну залежність від мітки (милі на галон):

Об'єм автомобіля в кубічних сантиметрах та його рейтинг в милях на галон. По мірі того, як двигун автомобіля стає більшим, його пробіг в милях на галон зазвичай зменшується:



Прискорення автомобіля та його рейтинг у милях на галон. Оскільки розгін автомобіля займає більше часу, рейтинг миль на галон зазвичай збільшується.



Потужність автомобіля та його пробіг у милях на галон. Зі збільшенням потужності автомобіля рейтинг миль на галон зазвичай зменшується.



1. **Основні алгоритми машинного навчання для задач регресії**

**Градієнтний спуск**

Градієнтний спуск — це математичний метод, який ітеративно знаходить ваги та зсув, які створюють модель з найменшими втратами. Градієнтний спуск знаходить найкращі ваги та зсув, повторюючи наступний процес для ряду визначених користувачем ітерацій.

Модель починає навчання з випадкових ваг та зсувів, близьких до нуля, а потім повторює наступні кроки:

1. Розрахуйте втрати з поточними вагами та зсувом.
2. Визначте напрямок переміщення ваг та зсуву, який зменшить втрати.
3. Трохи перемістіть значення ваг і зсув у напрямку, який зменшує втрати.
4. Поверніться до першого кроку та повторюйте процес до тих пір, поки модель не зможе далі зменшувати втрати.

Можна використовувати криву втрат, яка показує як зменшуються втрати в процесі навчання:



1. **Лінійна регресія: втрата**

Втрати — це числова метрика, яка описує, наскільки помилкові прогнози моделі. Втрати вимірюють відстань між прогнозами моделі та фактичними мітками. Мета навчання моделі — мінімізувати втрати, зменшивши їх до найменшого можливого значення.

На наступному зображенні ви можете візуалізувати втрати у вигляді стрілок, проведених від точок даних до моделі. Стрілки показують, наскільки далеко прогнози моделі від фактичних значень.



Відстань втрати У статистиці та машинному навчанні втрати вимірюють різницю між прогнозованими та фактичними значеннями. Втрата зосереджена на відстані між значеннями, а не на напрямку. Наприклад, якщо модель прогнозує 2, але фактичне значення дорівнює 5, нас не турбує, що втрата буде від'ємною -3=(-3$). Натомість нас турбує те, щоб відстань між значеннями була меньшою.

Два найбільш поширених методи зняття знака:

* Візьміть абсолютне значення різниці між фактичним значенням і прогнозом (L1)
* Возведіть у квадрат різницю між фактичним значенням і прогнозом (L2).

Далі можна спрогнозувати значення виходячі з цих оцінок втрат відповідно:

|  |  |  |
| --- | --- | --- |
| Тип втрати | Визначення | Рівняння |
| L1  | Сума модулів різниць між фактичними та прогнозними значеннями | $$\sum\_{}^{}|y\_{ф}-y\_{п}|$$ |
| Середня абсолютна помилка MАE | Середнє значення L1 | $$\frac{\sum\_{}^{}|y\_{ф}-y\_{п}|}{n}$$ |
| L2 | Сума квадратів різниць між фактичними та прогнозними значеннями | $$\sum\_{}^{}(y\_{ф}-y\_{п})^{2}$$ |
| Середня квадратична помилка MSE | Середнє значення L2 | $$\frac{\sum\_{}^{}(y\_{ф}-y\_{п})^{2}}{n}$$ |

Як правильно обрати метод обчислення втрат?

Вибір втрат

Рішення про те, чи використовувати MAE або MSE, може залежати від набору даних і способу обробки певних прогнозів. Більшість значень об'єктів у наборі даних зазвичай потрапляють у певний діапазон. Наприклад, автомобілі зазвичай важать від 2000 до 5000 фунтів і проїжджають від 8 до 50 миль на галлон. Автомобіль вагою 8000 фунтів або автомобіль, який проїжджає 100 миль на галлон, знаходиться за межами типового діапазону і буде вважатися аномалією.

Аномалія також може вказувати на те, наскільки далеко прогнози моделі від реальних значень. Наприклад, автомобіль вагою 3000 фунтів або автомобіль, який проїжджає 40 миль на галлон, знаходяться в межах типових діапазонів. Однак автомобіль вагою 3000 фунтів, який проїжджає 40 миль на галлон, буде аномалією з точки зору прогнозу моделі, оскільки модель прогнозує, що автомобіль вагою 3000 фунтів проїде від 18 до 20 миль на галлон.

Вибираючи найкращу функцію втрат, подумайте про те, як ви хочете, щоб модель обробляла аномалії. Наприклад, MSE більше впливає на модель в разі аномалій, ніж MAE. Втрата L2 веде до набагато вищого штрафу за аномалію, ніж втрата L1. Наприклад, на наступних зображеннях показані моделі, навчені за допомогою MAE і MSE. Червона лінія представляє повністю навчену модель, яка буде використовуватися для прогнозування. Аномалії ближче до моделі, навченій за допомогою MSE, ніж до моделі, навченій за допомогою MAE.



Модель, навчена за допомогою MSE, наближає модель до аномалій.



Модель, навчена за допомогою MAE, знаходиться далі від аномалій.

1. **Лінійна регресія: гіперпараметри**

Гіперпараметри — це змінні, які контролюють різні аспекти навчання. Три поширені гіперпараметри:

1. Швидкість навчання
2. Розмір партії
3. Епохи

На противагу цьому, параметри — це змінні, такі як ваги і зсув, які є частиною самої моделі. Іншими словами, гіперпараметри — це значення, якими ви управляєте; параметри — це значення, які модель обчислює під час навчання.

**Швидкість навчання**

Швидкість навчання — це задане вами число з плаваючою комою, яке впливає на швидкість збіжності моделі. Якщо швидкість навчання занадто низька, збіжність моделі може зайняти багато часу. Однак якщо швидкість навчання занадто висока, модель ніколи не збіжиться, а натомість "відскакує" від ваг і зсувів, які мінімізують втрати. Мета полягає в тому, щоб вибрати не надто високу і не надто низьку швидкість навчання, щоб модель швидко збігалася.

Швидкість навчання визначає величину змін, які потрібно внести у ваги і зсув на кожному етапі процесу градієнтного спуску. Модель множить градієнт на швидкість навчання, щоб визначити параметри моделі (значення ваги і зсуву) для наступної ітерації. На третьому етапі градієнтного спуску "

Різниця між параметрами старої моделі і параметрами нової моделі пропорційна нахилу функції втрат. Наприклад, якщо нахил великий, модель робить великий крок. Якщо маленький, то потрібно зробити невеликий крок. Наприклад, якщо величина градієнта дорівнює 2,5, а швидкість навчання — 0,01, то модель змінить параметр на 0,025.

Ідеальна швидкість навчання допомагає моделі збігатися за розумну кількість ітерацій. На рисунку крива втрат показує, що модель значно покращується протягом перших 20 ітерацій, перш ніж почати збіжуватися

Далі показані висока та низька швидкість навчання.



Перевірте своє розуміння.

Яка ідеальна швидкість навчання?

1.0

0,01

Ідеальна швидкість навчання залежить від проблеми.

**Розмір партії**

Розмір партії — це гіперпараметр, який відноситься до кількості прикладів, які модель обробляє перед оновленням своїх ваг та зміщення. Ви можете подумати, що модель повинна розрахувати втрати для кожного прикладу в наборі даних, перш ніж оновити ваги та зміщення. Однак, якщо набір даних містить сотні тисяч або навіть мільйони прикладів, використання всього пакету є недоцільним.

Використовуються

*Стохастичний градієнтний спуск (SGD):* стохастичний градієнтний спуск використовує лише один приклад (розмір партії =1) на ітерацію. При достатній кількості ітерацій SGD працює, але дуже шумно. «Шум» відноситься до змін під час навчання, які призводять до збільшення, а не зменшення втрат під час ітерації. Терміни «стохастичний» означає, що один приклад, що включає кожну партію, вибирається випадковим чином.



*Міні-пакетний стохастичний градієнтний спуск (міні-пакетний SGD):* Міні-пакетний стохастичний градієнтний спуск є компромісом між повним пакетом та SGD. Для кількості точок даних розмір пакету може бути будь-яким числом більше 1 і менше N. Модель випадковим чином вибирає приклади, включені в кожну партію, усереднює їх градієнти, а потім оновлює ваги та зміщення один раз за ітерацію.



Визначення кількості прикладів для кожного пакету залежить від набору даних і доступних обчислювальних ресурсів. Загалом, маленькі розміри партій ведуть себе як SGD, а великі розміри партій ведуть себе як повнопакетний градієнтний спуск.

1. **Використання нейронних мереж та бібліотек Python для рішення задачі регресії.**

 Нейронна мережа для прикладу регресійного аналізу може бути побудована за допомогою простого багатошарового персептрона (MLP) із декількома прихованими шарами. Це стандартна архітектура, яка підходить для задач регресії, оскільки може передбачати неперервні значення на виході.

Структура мережі для регресійного аналізу:

Вхідний шар:

Кількість нейронів у вхідному шарі відповідає кількості ознак (параметрів) вхідних даних. Наприклад, якщо для регресії використовуються дві ознаки, то на вхідному шарі буде два нейрони.

Приховані шари:

Нейронна мережа може мати один або кілька прихованих шарів, кожен із яких містить кілька нейронів (зазвичай від 32 до 128 нейронів на шар).

Для прихованих шарів зазвичай використовують функцію активації ReLU (Rectified Linear Unit), оскільки вона сприяє швидкій і стабільній роботі моделі.

Вихідний шар:

Оскільки мета — отримати неперервне значення, вихідний шар має один нейрон, який відображає передбачене значення.

Для регресійних задач функція активації на виході зазвичай відсутня або є лінійною, щоб результати були безпосередньо пропорційні вхідним значенням.

Функція втрат:

Для регресії часто використовується середньоквадратична похибка (MSE, Mean Squared Error), яка вимірює середню відстань між передбаченими та фактичними значеннями.

Оптимізатор:

Для навчання мережі використовують алгоритм оптимізації, такий як Adam або SGD (градієнтний спуск), щоб мінімізувати функцію втрат і поступово покращувати точність прогнозу.

Приклад коду на Python:

import numpy as np

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.optimizers import Adam

# Створення тренувальних даних

X\_train = np.array([[1, 2], [2, 3], [3, 4], [4, 5]]) # Приклад вхідних даних

y\_train = np.array([3, 5, 7, 9]) # Вихідні значення для регресії

# Створення моделі нейронної мережі

model = Sequential()

model.add(Dense(64, input\_dim=2, activation='relu')) # Перший прихований шар із 64 нейронів

model.add(Dense(32, activation='relu')) # Другий прихований шар із 32 нейронами

model.add(Dense(1)) # Вихідний шар з 1 нейроном для регресії

# Компіляція моделі

model.compile(optimizer=Adam(learning\_rate=0.01), loss='mse')

# Навчання моделі

model.fit(X\_train, y\_train, epochs=100, verbose=1)

# Передбачення для нових даних

X\_new = np.array([[5, 6]])

prediction = model.predict(X\_new)

print("Прогнозоване значення:", prediction)

Пояснення:

Dense(64, activation='relu') — перший прихований шар із 64 нейронів і функцією активації ReLU.

Dense(32, activation='relu') — другий прихований шар із 32 нейронів також використовує ReLU.

Dense(1) — вихідний шар без функції активації, оскільки для регресії потрібен неперервний вихід.

loss='mse' — середньоквадратична похибка обрана як функція втрат, яка підходить для задач регресії.

Ця модель намагається знайти залежність між вхідними ознаками та вихідним значенням, щоб передбачати неперервне значення для нових наборів даних.