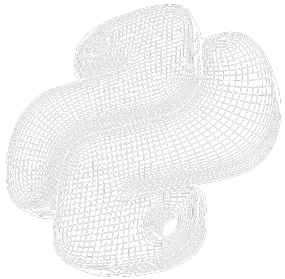
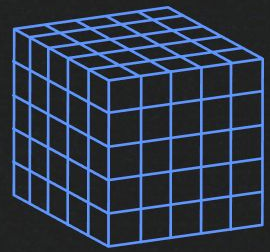


Заняття 7.
Задача прогнозування. Лінійна регресія.
Градієнтний спуск

План заняття

- Модель лінійної регресії: математична основа
- Метод найменших квадратів
- Метод градієнтного спуску
- Оцінка якості регресії
- Знайомство з бібліотекою scikit-learn

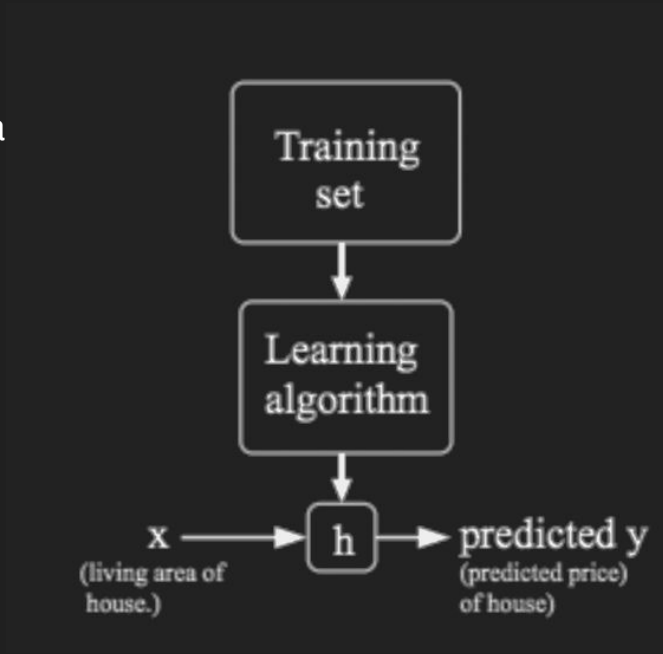


Supervised learning

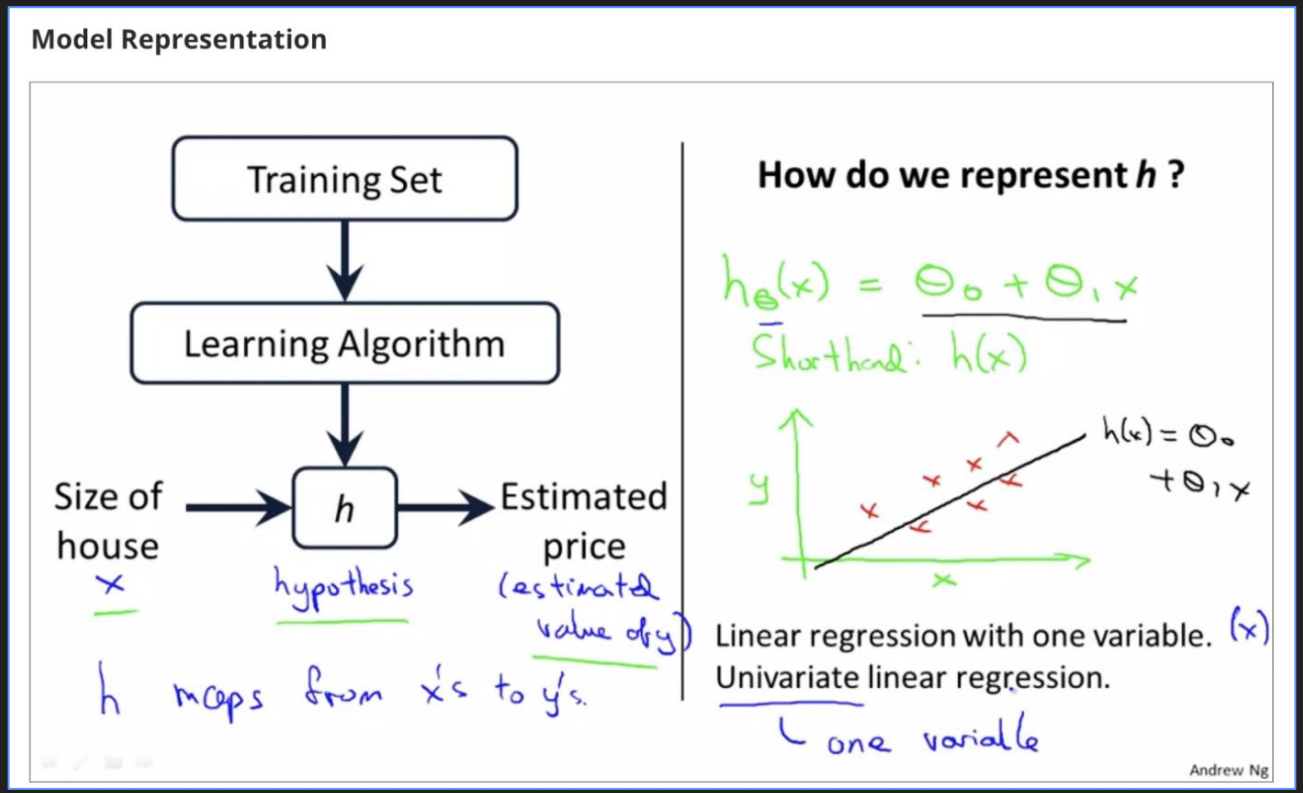
Маючи тренувальну вибірку (training set), що складається з *n* ознак (features):
$X: \{x_1, x_2, \ldots, x_n\}$ і цільової змінної (target variable) *Y*. Наша ціль навчити таку функцію $h : X \rightarrow Y$, що *h(x)* зможе «достатньо добре» оцінити відповідні значення *y*.

Коли величина, яку ми хочемо передбачити (*y*), є неперервною, ми називаємо таку aqпроблему **регресією**.

Коли ж *y* може приймати лише невелику кількість визначених дискретних значень, ми називаємо таку проблему **класифікацією.**



Univariate linear regression



Linear regression

- Univariate linear regression — це коли лише одна ознака для передбачення (feature):

  $h(x) = \theta_0 + \theta_1 * x$

- Multivariate linear regression — коли маємо дві й більше фіч:

  $h(x) = \theta_0 + \theta_1 * x_1 + \theta_2 * x_2$

  $h(x) = \theta_0 + \theta_1 * x_1 + \theta_2 * x_2 + \ldots + \theta_n * x_n$

  $h(x) = \sum_{i=0}^{n} \theta_i * x_i = \theta^T * X$, де X і θ — вектори розмірності n

Cost function

Тепер, знаючи, який вигляд має функція гіпотези h(x), як нам навчити її передбачати ціни оренди квартир?

Ми можемо шляхом «підлаштування» ваг (weights, θ в попередній нотації) довести функцію до стану, коли вона якнайкраще передбачає ті ціни оренди, які ми вже бачили, і після цього сподіватися, що вона так само добре визначатиме їх для нових квартир.

Але для цього нам потрібно визначити, як ми оцінюватимемо, чи добре функція передбачає ціну відомих нам квартир. Іншими словами, наскільки близько передбачення $h(x^{(i)})$ до відомого нам значення $y^{(i)}$.

Ми називатимемо це **функцією втрат (cost function)**.

Cost function

Функції втрат можуть бути різними. Для цього прикладу розглянемо одну з найпопулярніших функцій втрат — функцію квадратичної помилки:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x_i) - y_i\right)^2$$

J(θ) — функція втрат

θ — коефіцієнти функції гіпотези $h_\theta(x)$

$h_\theta(x_i)$ — передбачене значення цільової змінної *y* для і-го екземпляру даних

m — величина набору даних (кількість рядків)

n — кількість ознак

Наша ціль — мінімізувати функцію втрат!

Метод найменших квадратів (least squares)

Маючи нашу навчальну вибірку (training set), ми можемо сформувати матрицю X як m на n матрицю (або m на n+1, оскільки ми додаємо значення x0(i)=1), яка містить навчальні приклади (training examples) як рядки.

Також визначимо y, як вектор розмірністю m, що містить усі цільові змінні (target variables) з навчальної вибірки:

$$\vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \dots \\ y^{(m)} \end{bmatrix}$$

$$X\theta - \vec{y} = \begin{bmatrix} \left(x^{(1)}\right)^T \theta \\ \dots \\ \left(x^{(m)}\right)^T \theta \end{bmatrix} - \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(m)} \end{bmatrix} = \begin{bmatrix} \left(x^{(1)}\right)^T \theta - y^{(1)} \\ \dots \\ \left(x^{(m)}\right)^T \theta - y^{(m)} \end{bmatrix}$$

Метод найменших квадратів (least squares)

Використаємо відомий нам з лінійної алгебри факт, що для вектора z:
$z_T * z = \sum_i z_i^2$, а значить

$$\frac{1}{2m}(X\Theta - y)^T(X\Theta - y) = \frac{1}{2m}\sum_{i=1}^{m}(h_\Theta x^{(i)} - y^{(i)})^2 = J(\Theta)$$

Наше завдання — знайти такі коефіцієнти $\Theta = \Theta_1, \ldots \Theta_n$, за яких значення функції втрат $J(\Theta)$ будуть мінімальними. $J(\Theta)$ досягає мінімуму в точці, в якій похідна за кожним параметром Θ_j (часткова похідна) дорівнює нулю. Обчислюючи ці похідні, одержимо:

$$\Theta = (X^T X)^{-1} X^T y$$

Метод найменших квадратів (least squares)

Рішення, отримане за допомогою МНК, є аналітичним і є найкращою оцінкою параметрів θ, тобто це і будуть ті параметри, які забезпечують найвищу точність моделі.

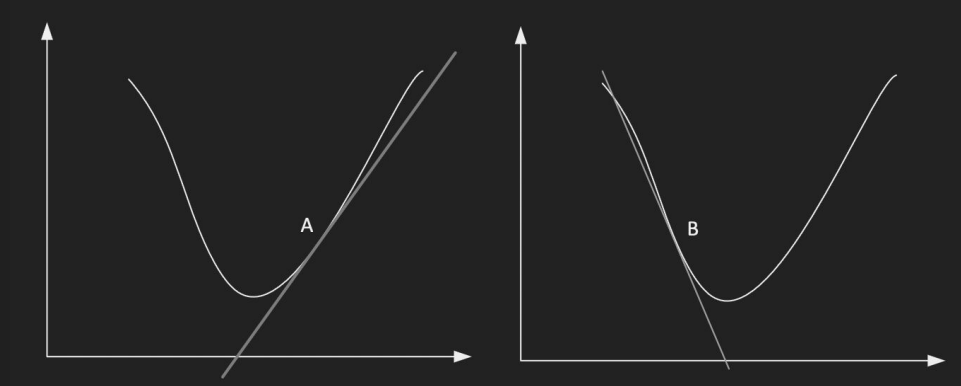
Але:

- пошук оберненої матриці — складна операція. Найкращий з відомих алгоритмів для її розрахунку має асимптотичну складність у $O(n^{2.373})$. Іншими словами, зі збільшенням кількості навчальних прикладів (training examples) у навчальній вибірці (training set) кількість часу, який потрібен для знаходження оберненої матриці, збільшується зі ступенем 2.373. Це призводить до того, що для задач з більшою навчальною вибіркою (training set) ітеративні методи навчання, як-от градієнтний спуск (**gradient descent**), працюють швидше. Під час навчання моделей на великих даних (big data) ітеративні методи навчання — це єдиний вибір, який у нас залишається
- матриця може бути виродженою (тобто не мати оберненої матриці). Таку проблему розв’язують за допомогою регуляризації

Метод найменших квадратів (least squares)

Похідна — «дотична» до функції, її значення в певній точці — кут нахилу цієї дотичної. Залежно від того, який вигляд має графік функції та в якій точці розраховуємо похідну, вона може бути або додатною (нахил «вгору»), або від'ємною (нахил «вниз»).

- Таким чином, орієнтуючись на знак похідної, ми можемо визначити, в який бік треба ступати», щоб наблизитися до мінімуму. Якщо в нашій точці похідна додатна (попереду підйом), потрібно зробити крок назад. Якщо похідна від'ємна (попереду спуск), потрібно зробити крок уперед.

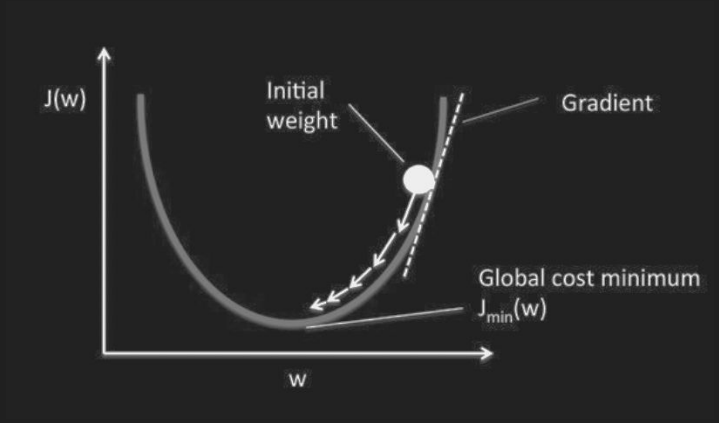


Додатна похідна (у точці A) і від'ємна похідна (у точці B)

Gradient Descent

Алгоритм має такі кроки:

1. Ініціалізуємо ваги (θ) випадковими значеннями.
2. Знайдемо напрямок найшвидшого спуску відносно точки, в якій перебуваємо, тобто напрямок протилежний градієнту в цій точці.
3. Зробимо невеликий крок у напрямку, визначеному в п. 2, опинимося в новій точці.
4. Повторюємо п. 2–3 до збіжності (тобто коли ми «топчемося на місці»).



Gradient Descent

Більш формально:

$$\text{repeat until convergence } \{$$
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad (\text{for } j = 0 \text{ and } j = 1)$$
$$\}$$

$\frac{\partial}{\partial \theta_j} J(\theta)$ — градієнт або часткова похідна функції втрат (cost function). Для спуску нам потрібно кожного разу робити крок у зворотному до градієнта напрямку. До того ж градієнт сам по собі зменшується з наближенням до мінімуму (у мінімумі функції градієнт дорівнює нулю), через це що ближче ми до нього, то менші кроки робитимемо.

α — навчальний темп (learning rate), що визначає, наскільки великі кроки ми робитимемо.

Gradient Descent

Градієнтний спуск для лінійної регресії:

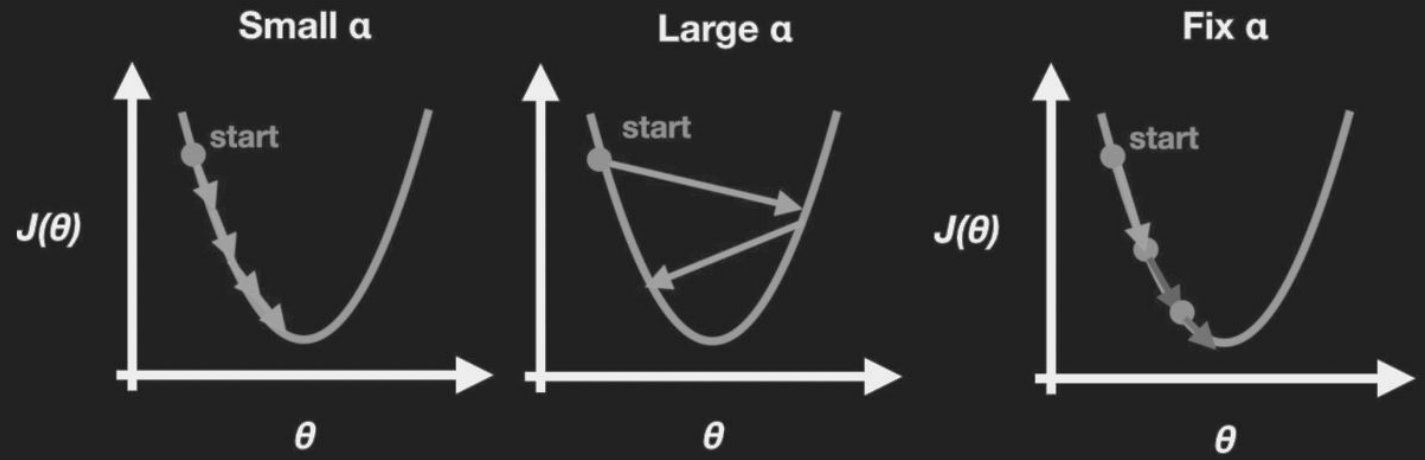
$$\text{Repeat until convergence } \{$$

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)} \qquad (\text{for every } j).$$

$$\}$$

Суть цього методу — подивитися на кожен приклад з навчальної вибірки (training set), визначити градієнт за всією сукупністю навчальних прикладів (training examples) і тільки після цього зробити «крок» градієнтного спуску. Цей спосіб називається **груповий градієнтний спуск (batch gradient descent).**
Також є модифікації, як-от stochastic gradient descent і mini-batch gradient descent.

Learning rate choice



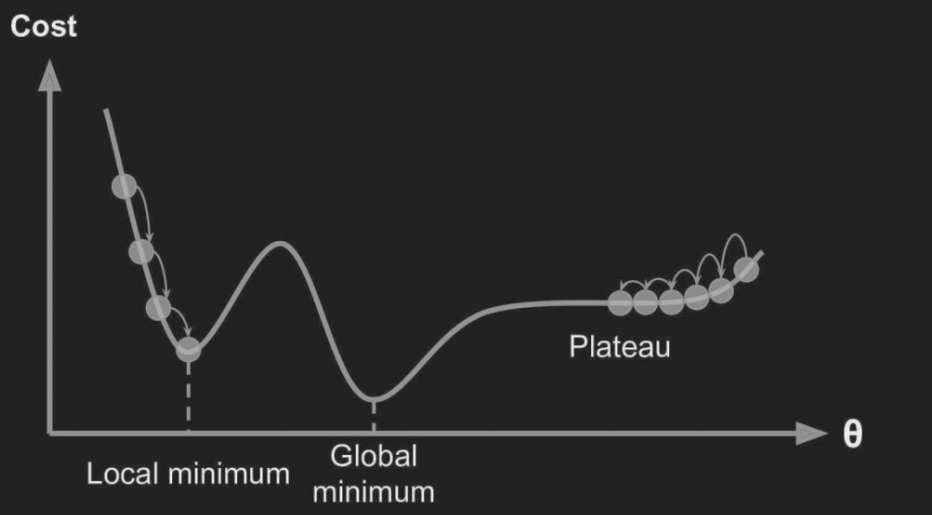
«Спускаємося» дуже повільно (алгоритм повільно збігається)

Алгоритм довго збігається, тому що «перестрибує» зі сторони в сторону, оминаючи мінімум

Фіксований learning rate дозволяє алгоритму робити менші й менші кроки з кожною ітерацією (оскільки помилки стають меншими)

Gradient Descent

Градієнтний спуск — це ітераційний алгоритм оптимізації, в якому для знаходження **локального** мінімуму функції здійснюють кроки, пропорційні протилежному значенню градієнта (або наближеного градієнта) функції в поточній точці.



Оцінка якості регресії

Оцінка якості моделі дуже важлива в Data Science. Це допоможе вам зрозуміти ефективність вашої моделі та полегшить презентацію вашої моделі іншим людям. Існує багато різних метрик якості, але лише деякі з них підходять для оцінки регресії.

Ми розглянемо:

- MSE
- RMSE
- MAE

Mean Squared Error (MSE)

MSE обчислюють як середнє квадратів різниць між прогнозованими та фактичними цільовими значеннями в наборі даних.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Mean Square Error formula

Зведення у квадрат також має ефект «роздування» або збільшення великих помилок. Тобто що більша різниця між прогнозованими та очікуваними значеннями, то більша підсумкова помилка. Це призводить до більшого «покарання» моделей за більші помилки, коли MSE використовують як функцію втрат.

Root Mean Squared Error (RMSE)

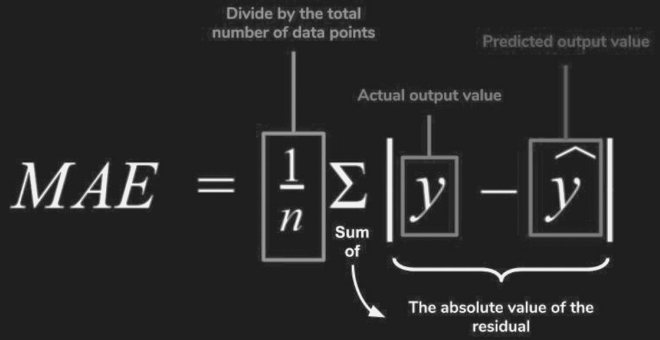
Дорівнює кореню квадратному з MSE. Але на відміну від MSE, має ту саму розмірність, що й цільова змінна.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$$

Так само, як MSE чутлива до великих помилок, а отже, якщо є викиди, ця метрика матиме високе значення.

Mean Absolute Error (MAE)

Середня абсолютна похибка (MAE) подібна до середньоквадратичної похибки (MSE). Однак замість суми квадратів похибок у MSE, MAE бере суму абсолютного значення похибки. MAE — це лінійна оцінка, яка означає, що всі відмінності зважуються однаково незалежно від розміру.



Loss function vs Evaluation metric

Cost function є більш загальним поняттям, ніж loss function. Це може бути сума loss functions у вашому навчальному наборі плюс деякий штраф за складність моделі (регуляризація). Але у випадках розглянутих сьогодні це те саме.

Loss function — частина моделі. Вона впливає на те, як у моделі враховуватимуть помилки.

Метрика оцінки якості може і збігатися, і відрізнятися від Loss function. Метрик може бути кілька: головна та допоміжні. Вони потрібні для того, щоб оцінити якість моделі. Метрики не беруть участі в процесі оптимізації.

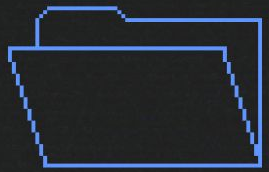
Sklearn

Scikit-learn (Sklearn) — це найкорисніша та надійна бібліотека для машинного навчання на Python. Вона надає вибір ефективних інструментів для машинного навчання та статистичного моделювання, включно із класифікацією, регресією, кластеризацією та зменшенням розмірності через інтерфейс Python. Ця бібліотека, яка здебільшого написана на Python, побудована на NumPy, SciPy і Matplotlib.

https://scikit-learn.org/stable/

Приклад лінійної регресії з sklearn:
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html#sklearn.linear_model.LinearRegression

Live coding / Практика



???

Q&A