Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

MATHEMATICS AND STATISTICS FOR DATA SCIENCE

Bài 4: *Calculus*

Phòng LT & Mạng

https://csc.edu.vn/lap-trinh-va-csdl/Mathematics-and-Statistics-for-Data-Science_194

2021

Nội dung

1. **Calculus**
2. **Multivariate Calculus**
3. **Gradient Descent trong Python**

Calculus

❑Calculus (Giải tích)

- *Calculus (from Latin calculus, literally 'small pebble', used for counting and calculations, as on an abacus)[1] is the mathematical study of continuous change, in the same way that geometry is the study of shape and algebra is the study of generalizations of arithmetic operations.~ Wikipedia*
- Tạm dịch: Giải tích (nguồn gốc từ Latinh, nghĩa đen là 'viên sỏi nhỏ', được sử dụng để đếm và tính toán, như trên bàn tính) là nghiên cứu toán học về sự thay đổi liên tục, giống như hình học là nghiên cứu về hình dạng và đại số là nghiên cứu khái quát hóa các tính toán số học.

Calculus

❑Khái niệm cốt lõi của Calculus

- Function (Hàm số)
  - Phương trình (equation) là một hàm số nếu: với bất kỳ x nào trong miền của phương trình, phương trình sẽ mang lại chính xác một giá trị của y khi chúng ta đánh giá phương trình tại một x cụ thể.

$$y = f(x)$$

Calculus

- Derivative (Đạo hàm)
  - Trong Calculus, đạo hàm của một hàm số thực chất là việc mô tả sự biến thiên của hàm số tại một điểm nào đó.
  - Đạo hàm của f(x) đối với x là hàm f'(x) - "f prime of x" - và được định nghĩa:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

http://tutorial.math.lamar.edu/Classes/CalcI/DefnOfDerivative.aspx

Calculus

- Derivative trong Python

```
from scipy.misc import derivative
from math import *
```

```
def f(x):
    return pow(x,2)
```

```
# Solution 1
# scipy.misc.derivative(func, x0, dx=1.0, n=1, args=(), order=3)
# when x = 5, h = 1e-10
f_prime_of_x = derivative(f, 5, dx=1e-10)
f_prime_of_x
```

```
10.00000082740371
```

numerator Δy

denominator Δx

```
# Solution 2
def derive(function, value):
    h = 1e-10
    top = function(value + h) - function(value)
    bottom = h
    slope = top / bottom
    # Returns the slope to the third decimal
    return float("%.3f" % slope)
```

```
# when x = 5, h = 1e-10
f_prime_of_x = derive(f, 5)
f_prime_of_x
```

```
10.0
```

Calculus

❑Ex1: Derivative

- Tính toán “truyền thống”
- Dùng hàm derivative() trong scipy.misc

Calculus

- Product Rule (Quy tắc nhân)
  - Nếu hai hàm số f(x) và g(x) khác nhau (nghĩa là tồn tại đạo hàm) thì product (phép nhân) có thể khác nhau và:

$$(f\,g)' = f'\,g + f\,g'$$

  - Ví dụ: Cho $f(x) = x^3$ và $g(x) = x^6$
    - $(fg)'=(x^3x^6)'=(x^9)'=9x^8$
    - $f'(x)g'(x)=(3x^2)(6x^5)=18x^7$
    - => $(fg)'\neq f'g'$

http://tutorial.math.lamar.edu/Classes/CalcI/ProductQuotientRule.aspx

Calculus

- Product Rule trong Python

```python
def f1(x):
    return pow(x,3)
def g1(x):
    return pow(x,6)
```

```python
# x = 2
x = 2
fg_prime_of_x = (derivative(f1, x, dx=1e-10) * g1(x)) + (f1(x) * derivative(g1, x, dx=1e-10))
fg_prime_of_x
```

```
2304.000190633815
```


Calculus

- Quotient Rule (Quy tắc chia)
  - Nếu hai hàm số f(x) và g(x) khác nhau (nghĩa là tồn tại đạo hàm) thì quotient (phép chia) có thể khác nhau và:

$$\left(\frac{f}{g}\right)' = \frac{f'g - fg'}{g^2}$$

http://tutorial.math.lamar.edu/Classes/CalcI/ProductQuotientRule.aspx

Calculus

- Ví dụ: Cho f(x) = x^3 và g(x) = x^6

$$\left(\frac{f}{g}\right)' = \left(\frac{x^3}{x^6}\right)' = \left(\frac{1}{x^3}\right)' = \left(x^{-3}\right)' = -3x^{-4} = -\frac{3}{x^4}$$

$$\frac{f'(x)}{g'(x)} = \frac{3x^2}{6x^5} = \frac{1}{2x^3}$$

$$\left(\frac{f}{g}\right)' \neq \frac{f'}{g'}$$

Calculus

- Quotient Rule trong Python

```
x = 2
f_prime_of_x = derivative(f1, x, dx=1e-10)
g_prime_of_x = derivative(g1, x, dx=1e-10)
f_div_g_prime_of_x = ((f_prime_of_x * g1(x)) - (f1(x) * g_prime_of_x))/pow(g1(x),2)
f_div_g_prime_of_x
```

```
-0.18750001551381956
```

Calculus

❑ Ex3: Derivative Rules

- Tính toán “truyền thống”
- Dùng hàm derivative() trong scipy.misc

Calculus

- Integrals (Tích phân)
  - Nếu F(x) là nguyên hàm (anti-derivative) bất kỳ của f(x) : F′(x)=f(x)
  - Thì anti-derivative chung nhất của f(x) được gọi là tích phân không xác định (indefinite integral) và được định nghĩa như sau:

$$\int f(x)\, dx = F(x) + c$$

  - Ví dụ:

$$\int x^4 + 3x - 9\, dx = \frac{1}{5}x^5 + \frac{3}{2}x^2 - 9x + c$$

http://tutorial.math.lamar.edu/Classes/CalcI/IndefiniteIntegrals.aspx

Calculus

- Definite Integral (Tích phân xác định)
  - Cho hàm số f(x) liên tục trên khoảng [a, b], chúng ta chia khoảng này cho n khoảng phụ có chiều rộng bằng nhau (Δx) và từ mỗi khoảng, chọn một điểm, x^*_i. Khi đó tích phân xác định của f(x) từ a đến b là:

$$\int_a^b f(x)\,dx = \lim_{n\to\infty}\sum_{i=1}^{n} f(x_i^*)\,\Delta x$$

http://tutorial.math.lamar.edu/Classes/CalcI/DefnofDefiniteIntegral.aspx

Calculus

- Definite Integral trong Python
  - Single Integral: $\int_a^b f(x)dx$

scipy.integrate.quad(function, a, b)

  - Ví dụ: Tính $\int_1^4 x^2\,dx$

```
import scipy.integrate
from numpy import exp
```

```
f= lambda x:x*x
i,error = scipy.integrate.quad(f, 1, 4)
i
```

```
21.000000000000004
```

Calculus

- Definite Integral trong Python
  - Multiple integral: Có thể dùng các hàm dblquad, tplquad và nquad để tính multiple integral. Các hàm này tích hợp 2, 3 và n lớp (biến) tương ứng. Limit của tất cả các tích phân bên trong cần được xác định là các hàm.

Calculus

- Definite Integral trong Python
  - Double integral:

    scipy.integrate.dblquad(func, a, b, gfun, hfun)
  - Ví dụ: Tính $\int_0^{1/2} dy \int_0^{\sqrt{1-4y^2}} 16xy\, dx$

```
import scipy.integrate
from numpy import exp
from math import sqrt
```

```
f = lambda x, y : 16*x*y
g = lambda x : 0
h = lambda y : sqrt(1-4*y**2)
i, error = scipy.integrate.dblquad(f, 0, 0.5, g, h)
print(i)
```

```
0.5
```

Calculus

❑ Ex4: Definite Integral

- Tính toán “truyền thống”
- Dùng hàm quad() trong scipy.integrate

Nội dung

1. Calculus
2. **Multivariate Calculus**
3. Gradient Descent trong Python

Multivariate Calculus

❑ Giới thiệu

- *"Multivariate Calculus (also known as multivariable calculus) is the extension of calculus in one variable to calculus with functions of several variables: the differentiation and integration of functions involving multiple variables, rather than just one." ~Wikipedia*
- Tạm dịch: *"Multivariate Calculus* - Giải tích đa biến *(còn được gọi là multivariable calculus) là phần mở rộng của giải tích một biến để tính toán với các phương thức nhiều biến: phân biệt và tích hợp các phương thức liên quan đến nhiều biến, thay vì chỉ một biến."*

Multivariate Calculus

- *Calculus* (Giải tích) là một bộ công cụ để phân tích mối quan hệ giữa các function và input của chúng. Trong giải tích đa biến, chúng ta có thể lấy một function có nhiều input và xác định ảnh hưởng của từng input riêng biệt.

Multivariate Calculus

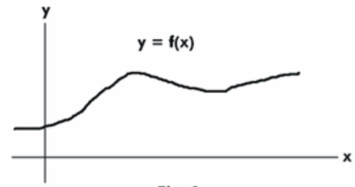


Fig. 1

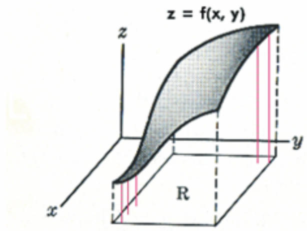
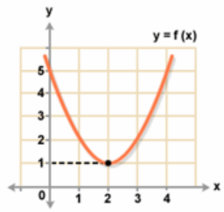
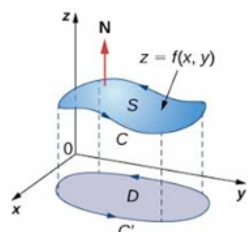


Fig. 2





https://www.toppr.com/bytes/multivariable-calculus/

Multivariate Calculus

- Partial Derivatives (Đạo hàm riêng)
  - Trong toán học, **đạo hàm riêng** của một hàm số đa biến là đạo hàm theo một biến, các biến khác được xem như là hằng số (khác với đạo hàm toàn phần - total derivative, khi tất cả các biến đều biến thiên - vary). Đạo hàm riêng được sử dụng trong giải tích vector và hình học vi phân.
  - Ví dụ: cho $f(x,y)= x^4 + 6\sqrt{y} - 10$
    - Đạo hàm riêng với x : $f_x(x,y)=4x^3$
    - Đạo hàm riêng với y: $f_y(x,y)=3/\sqrt{y}$

https://vi.wikipedia.org/wiki/%C4%90%E1%BA%A1o_h%C3%A0m_ri%C3%AAng

Calculus

- Partial Derivatives trong Python
  - Sử dụng thư viện: sympy (pip install sympy)

```
from sympy import diff, Symbol
```

```
x = Symbol('x')
y = Symbol('y')
f = x**4 + 6*y**0.5 -10
```

```
part_x = diff(f, x)
part_x
```

$4x^3$

```
part_x_value =part_x.subs({x:3})
part_x_value
```

108

```
part_y = diff(f, y)
part_y
```

$\frac{3.0}{y^{0.5}}$

```
part_y_value =part_y.subs({y:4})
part_y_value
```

1.5

Multivariate Calculus

❑ Ex2: Partial Derivatives

- Tính toán "truyền thống"
- Dùng hàm diff() trong sympy

Multivariate Calculus

❑ Tại sao Multivariate Calculus lại quan trọng trong Data Science?

- Trong Data Science, chúng ta cố gắng tìm các inputs có thể cho phép một function phù hợp nhất với dữ liệu. Độ dốc (*slope*/ *descent*) mô tả tốc độ thay đổi ouput đối với input. Xác định ảnh hưởng của từng input tới output là một trong những nhiệm vụ quan trọng. Vì vậy, đòi hỏi chúng ta cần hiểu rõ giải tích đa biến.

Multivariate Calculus

❑ Multivariate Calculus được áp dụng như thế nào trong Data Science?

- Gradient (độ dốc)
  - Gradient hiển thị các thay đổi dọc theo một biến hoặc tập hợp biến cụ thể dưới dạng một hàm "di chuyển" trực tiếp trong không gian và chúng rất dễ tính toán trong các khung tối ưu hóa (optimization frameworks). Các thuật toán tìm cực tiểu / cực đại (minima/maxima) của các hàm này và tối ưu hóa cơ sở trên các ước tính này.

Multivariate Calculus

- Trong toán học, **độ dốc - slope** (Hệ số Góc) hay còn gọi là **gradient** là một đường thẳng biểu diễn độ dốc hay grat. Giá trị của độ dốc càng cao thì độ nghiêng của đường thẳng càng cao. Độ dốc thường được mô tả là tỉ lệ của sự gia tăng giữa hai điểm trên trục *y* của đường thẳng chia cho sự gia tăng giữa hai điểm trên trục *x* của đường thẳng đó. Trong toán học, độ dốc *m* (hoặc *i*) của một đường thẳng chính là:

$$i = m = (y_2 – y_1)/ (x_2 - x_1)$$

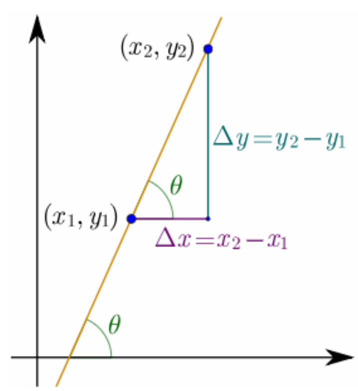
- Khái niệm độ dốc được áp dụng trực tiếp trong gradient trong hình học

https://vi.wikipedia.org/wiki/%C4%90%E1%BB%99_d%E1%BB%91c

Multivariate Calculus

- Độ dốc của một đường thẳng trên một mặt phẳng được định nghĩa là tỉ lệ giữa sự thay đổi ở tọa độ y chia cho sự thay đổi ở tọa độ x:

$$m = \left(\frac{\Delta y}{\Delta x}\right) = \tan(\theta)$$



- Ví du:

• Giả sử một đường thẳng chạy qua hai điểm *P* = (1, 2) và *Q* = (13, 8). Chúng ta có thể tìm độ dốc bằng cách chia sự thay đổi của tọa độ y cho sự thay đổi ở tọa độ x:

$$m = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1} = \frac{8-2}{13-1} = \frac{6}{12} = \frac{1}{2}.$$

• Một ví dụ khác, giả sử một đường thẳng chạy qua các điểm (4, 15) và (3, 21). Độ dốc của đường thẳng sẽ là

$$m = \frac{21-15}{3-4} = \frac{6}{-1} = -6.$$

Multivariate Calculus

- Tại sao gradient thường được tham chiếu trong Machine Learning?
  - Một giảm thiểu (minimize) được gọi là hàm mất mát - “loss” function, đó là thước đo mức độ xa của mô hình trong việc dự đoán chính xác dữ liệu huấn luyện. Ví dụ, lỗi bình phương (squared error) trong trường hợp hồi quy đơn giản.
  - Hàm mất mát được giảm thiểu theo các trọng số hoặc tham số của mô hình để tìm ra mô hình tốt nhất có thể phù hợp với dữ liệu quan sát được. Mô hình tốt nhất (“Best model”) ở đây có nghĩa là đơn giản là bộ giá trị tốt nhất cho các tham số.

Multivariate Calculus

- Ví dụ: Lấy ví dụ lớp mô hình hồi quy tuyến tính (linear regression model): Y = a * X + b. Khi mô hình có tham số a = 0.1, b = 3.0 được coi là một mô hình khác với a = 0.2, b = 2.0. Nhiệm vụ là tìm ra vector hệ số tốt nhất (a, b) để giảm thiểu lỗi dự đoán.

Multivariate Calculus

- Gradient là một vector được tạo thành từ các tham số mô hình (model parameters) chỉ theo hướng đi lên dốc nhất từ một điểm nhất định trên loss function. Thuật toán giảm độ dốc (gradient descent) sau đó sẽ hạ thấp giá trị của loss function bằng cách trượt các tham số theo hướng ngược lại với độ dốc.
- Gradient decent tốt là phương pháp hiệu quả giảm loss trong Machine Learning. Trong lập trình, cần phải tính toán các đạo hàm bằng cách dùng các thư viện như TensorFlow giúp tự động tính toán các vi phân tự động và các biểu thức đại số thông qua quy tắc chuỗi (chain rule).

Nội dung

1. Calculus
2. Multivariate Calculus
3. **Gradient Descent trong Python**

Gradient Descent trong Python

- Khi chúng ta tìm hiểu về Machine Learning, một trong những khía cạnh cơ bản cần biết là tìm hiểu về Gradient Gradient Descent.

- Gradient descent là xương sống của một thuật toán Machine Learning. Khi chúng ta nắm được gradient descent, mọi thứ bắt đầu rõ ràng hơn và dễ hiểu hơn trên các thuật toán khác nhau. Để tự xây dựng gradient descent, chúng ta cần một số package cơ bản như numpy và matplotlib.

https://towardsdatascience.com/gradient-descent-in-python-a0d07285742f

Gradient Descent trong Python

- Gradient descent là một thuật toán tối ưu hóa hoạt động bằng cách tìm kiếm hiệu quả không gian tham số, intercept (θ_0) và slope (θ_1) cho hồi quy tuyến tính (linear regression), theo quy tắc sau:

$$\theta := \theta - \alpha \frac{\delta}{\delta\theta} J(\theta).$$

  - Với J(θ) được gọi là hàm chi phí (cost function) và α là learning rate (tự thiết lập).

Gradient Descent trong Python

- **Trong phạm vi bài này, chúng ta sẽ sử dụng hàm chi phí bình phương tối thiểu (squares cost function) được xác định như sau:**

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

  - Với m là tổng số mẫu huấn luyện và $h\theta(x^{(i)})$ là hàm giả thuyết (hypothesis function) được định nghĩa như sau: $h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$
  - Trong đó, (i) được sử dụng được sử dụng để biểu thị mẫu thứ i

Gradient Descent trong Python

- Chúng ta cần tính đạo hàm cho cả θ_0 and θ_1:

$$\frac{\partial}{\partial\theta_0}J(\theta_0,\theta_1) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})$$

$$= \frac{1}{m}\sum_{i=1}^{m}(\theta_0 + \theta_1 x^{(i)} - y^{(i)})$$

$$\frac{\partial}{\partial\theta_1}J(\theta_0,\theta_1) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x^{(i)}$$

$$= \frac{1}{m}\sum_{i=1}^{m}(\theta_0 + \theta_1 x^{(i)} - y^{(i)})x^{(i)}$$

Gradient Descent trong Python

❑ Thực hiện

- Chuẩn bị dữ liệu
  - Ví dụ: Tạo ra dữ liệu tuyến tính có kèm theo nhiễu

from *sklearn.datasets.samples_generator* **import** make_regression

```
# Dataset
X, y = make_regression(n_samples=100, n_features=1, n_informative=1, random_state=0, noise=35)
```

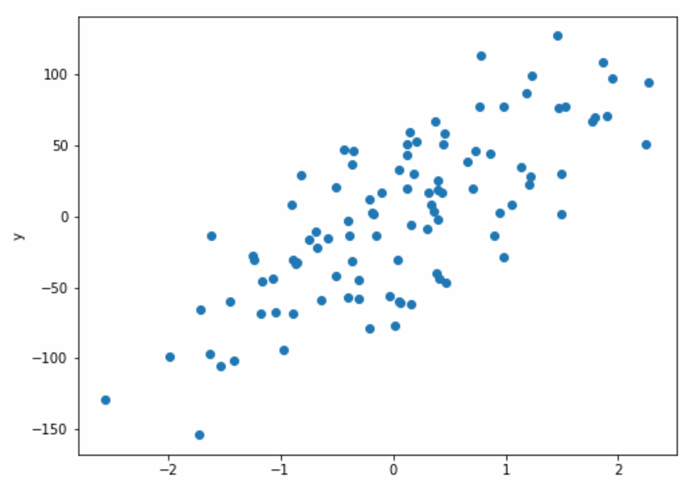
https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_regression.html
https://scikit-learn.org/stable/glossary.html#term-random-state

Gradient Descent trong Python

- Trực quan hóa dữ liệu
  - Ví dụ:

```
plt.figure(figsize=(8,6))
plt.scatter(X, y)
plt.xlabel("X")
plt.ylabel("y")
plt.show()
```

Theo hình ta thấy y có mối quan hệ tuyến tính khá tốt với X. Dữ liệu này đơn giản và chỉ có một biến X độc lập.



Gradient Descent trong Python

- Viết hàm tìm m, b (theta)

```
def gradient_descent_2(alpha, x, y, numIterations):
    # x = [[1 x0],[1 x1], [1  x2]...]
    m = x.shape[0] # number of samples
    theta = np.ones(2)
    for iter in range(0, numIterations):
        # hypothesis = theta0 + theta1.x ~ x.dot(theta) ~ (1 x).dot (theta0 theta1)
        hypothesis = np.dot(x, theta)
        loss = hypothesis - y
        J = np.sum(loss ** 2) / (2 * m)  # cost
        print("iter %s | J: %.3f" % (iter, J))
        theta0_prime = np.sum(loss)/m
        theta1_prime = np.sum(loss * x[:,1])/m

        gradient = np.array([theta0_prime, theta1_prime])
        theta = theta - alpha * gradient  # update
        # lan lap dau tien
        if iter ==0:
            print('hypothesis',hypothesis)
            print('loss', loss)
            print('gradient',gradient)
            print('theta',theta)
    return theta
```


Gradient Descent trong Python

- Gọi hàm tìm m, b (theta)

```
# y = mx + b
m, n = np.shape(X)
print(m,n)
X = np.c_[ np.ones(m), X] # insert column
print(X[0:3])
alpha = 0.01 # learning rate
```

```
100 1
[[ 1.         -0.35955316]
 [ 1.          0.97663904]
 [ 1.          0.40234164]]
```

```
theta = gradient_descent_2(alpha, X, y, 1000)
```

```
print("m = ", theta[1], "b = ", theta[0])
```

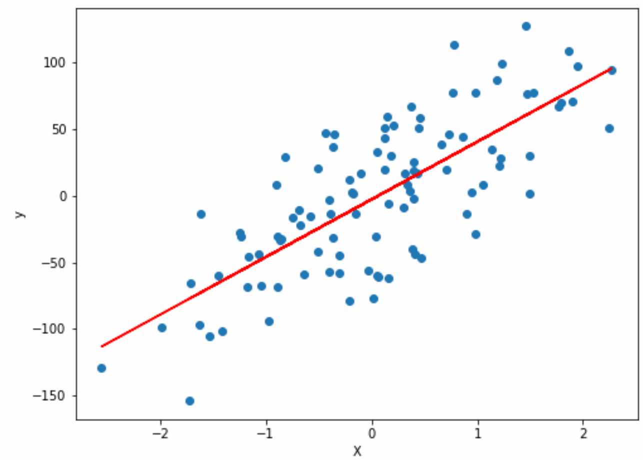
```
m =  43.20234846939813 b =  -2.8483795729664574
```

Gradient Descent trong Python

- Tìm regression line từ m, b (theta), trực quan hóa kết quả

```python
# plot
for i in range(X.shape[1]):
    y_predict = theta[0] + theta[1]*X
```

```python
plt.figure(figsize=(8,6))
plt.scatter(X[:,1],y, marker='o')
plt.plot(X,y_predict, c='r')
plt.xlabel("X")
plt.ylabel("y")
plt.show()
```



Gradient Descent trong Python

- Dự đoán khi có giá trị X_new

```python
# dự đoán giá trị mới
Xnew = np.array([-2,-1,0,1,2])
for i in range(Xnew.size):
    y_predict_new = theta[0] + theta[1]*Xnew
```

```python
y_predict_new
```

```
array([-89.25307651, -46.05072804,  -2.84837957,  40.3539689 ,
        83.55631737])
```

Gradient Descent trong Python

- Ex5: Gradient Descent

- Ex6: Gradient Descent – SAT

B4. Calculus

Bổ sung cho bài giảng

2019

Nội dung bổ sung

1. **Đạo hàm và tích phân**
2. **Gradient Descent**

1. Đạo hàm và tích phân

❑ Dãy số (*sequence*)

$$f : N \to R$$
$$x_n = f(n)$$
$$\{x_n\}_n \equiv \{x_n\} \equiv x_1, x_2, x_3, \cdots, x_n, \cdots$$

❑ Giới hạn của dãy số (hội tụ)

$$\lim_{n\to\infty} x_n = a \qquad x_n \to a, n \to \infty$$
$$\forall(\varepsilon > 0), \exists n_0 : \forall n \geq n_0 : |x_n - a| < \varepsilon$$

1. Đạo hàm và tích phân (tt.)

❑ Một số tính chất cơ bản của giới hạn

- Mọi dãy hội tụ đều có giới hạn duy nhất
- Mọi dãy hội tụ đều bị chặn
- Giả sử: $\lim_{n\to\infty} x_n = a, \lim_{n\to\infty} y_n = b$. Ta có:

$$(i)\ \lim_{n\to\infty}(x_n + y_n) = a + b$$
$$(ii)\ \lim_{n\to\infty}(c + x_n) = c + a$$
$$(iii)\ \lim_{n\to\infty}(c.x_n) = c.a$$
$$(iv)\ \lim_{n\to\infty}(x_n.y_n) = a.b$$
$$(v)\ \lim_{n\to\infty}\left(\frac{x_n}{y_n}\right) = \frac{a}{b}, y_n \neq 0, b \neq 0$$

1. Đạo hàm và tích phân (tt.)

- **Giới hạn của hàm số**
  - A là *lân cận* của $x_0 \in$ } : $\exists(\delta > 0)$: $(x_0 - \delta, x_0 + \delta) \subset A$
  - Hàm số y = f(x) xác định trên lân cận A của x_0

$$\lim_{x \to x_0} f(x) = L \qquad f(x) \to L, x \to x_0$$

$$\forall(\varepsilon > 0), \exists(\delta > 0):$$
$$\forall |x - x_0| < \delta \Rightarrow |f(x) - L| < \varepsilon$$

- **Giả sử f(x) xác định trên A. Hàm f(x) liên tục tại $x_0 \in$ A:**

$$\lim_{x \to x_0} f(x) = f(x_0)$$

$$\forall(\varepsilon > 0), \exists(\delta > 0):$$
$$\forall |x - x_0| < \delta \Rightarrow |f(x) - f(x_0)| < \varepsilon$$

  - Hàm f(x) liên tục trên A nếu f(x) liên tục tại mọi x $\in$ A

1. Đạo hàm và tích phân (tt.)

- **Một số tính chất cơ bản của đạo hàm**
  - Nếu f có đạo hàm tại x_0 thì f liên tục tại x_0
  - Giả sử $f(x), g(x)$ có đạo hàm tại x. Ta có:

$$(i) \quad (a.f + b.g)'(x) = a.f'(x) + b.g'(x)$$

$$(ii) \quad (f.g)'(x) = f'(x).g(x) + g'(x).f(x)$$

$$(iii) \quad \left(\frac{f}{g}\right)'(x) = \frac{f'(x).g(x) - g'(x).f(x)}{g^2(x)}$$

$$(iv) \quad (f \circ g)'(x) = f'(g(x)).g'(x)$$

1. Đạo hàm và tích phân (tt.)

❑ Đạo hàm của một số hàm sơ cấp

1) $f(x) = a \Rightarrow f'(x) = 0$

2) $f(x) = x \Rightarrow f'(x) = 1$

3) $f(x) = x^{\alpha}, \alpha \in R \setminus \{-1\} \Rightarrow f'(x) = \alpha.x^{(\alpha-1)}$

4) $f(x) = \frac{1}{x} \Rightarrow f'(x) = \frac{-1}{x^2}$

5) $f(x) = a^x, a > 0 \Rightarrow f'(x) = a^x.\ln(a)$

6) $f(x) = e^x \Rightarrow f'(x) = e^x$

7) $f(x) = \log_a(x), a > 0 \Rightarrow f'(x) = \frac{1}{x.\ln(a)}$

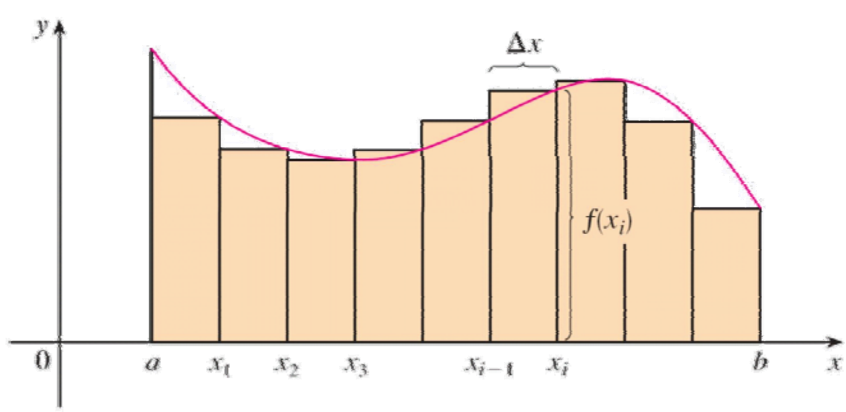
8) $f(x) = \ln(x) \Rightarrow f'(x) = \frac{1}{x}$

9) $f(x) = \sin(x) \Rightarrow f'(x) = \cos(x)$

10) $f(x) = \cos(x) \Rightarrow f'(x) = -\sin(x)$

1. Đạo hàm và tích phân (tt.)

❑ Tích phân xác định



Nội dung bổ sung

1. Đạo hàm và tích phân

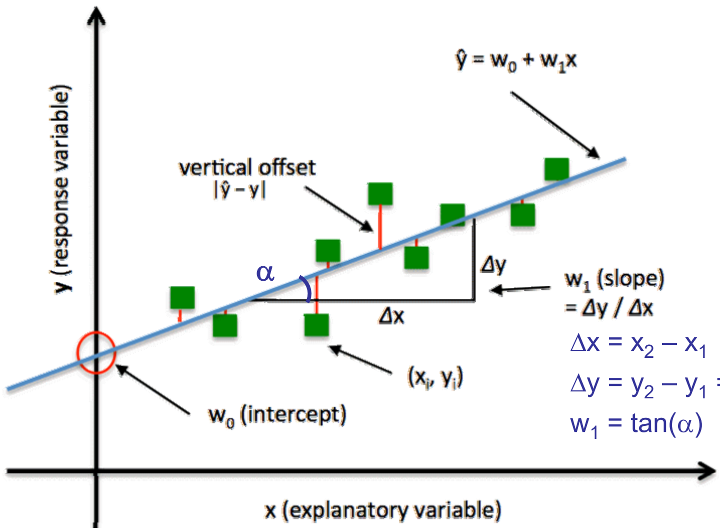
2. **Gradient Descent**

2. Gradient Descent

- **Hồi quy tuyến tính** (*linear regression*)
  - tên khác: *linear fitting*, *linear least square*

2. Gradient Descent (tt.)

❑ **Hồi quy tuyến tính** (*linear regression*)



$\Delta x = x_2 - x_1$

$\Delta y = y_2 - y_1 = f(x_2) - f(x_1)$

$w_1 = \tan(\alpha)$

2. Gradient Descent (tt.)

❑ **Thuật toán Gradient Descent**

- local/global minimum (maximum)
- vòng lặp tìm optimal point x* tiến gần đến x_0 (local minimum)

$f'(x^{(t)}) > 0$: $x^{(t)}$ ở bên PHẢI của $x_0 \Rightarrow$ cần lùi sang TRÁI (A)

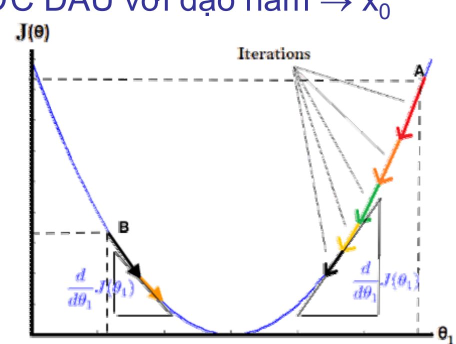
$f'(x^{(t)}) < 0$: $x^{(t)}$ ở bên TRÁI của $x_0 \Rightarrow$ cần tiến sang PHẢI (B)

<u>Tóm lại</u>: $x^{(t)}$ cần di chuyển NGƯỢC DẤU với đạo hàm $\rightarrow x_0$

$$x^{(t+1)} = x^{(t)} - \rho . f'(x^{(t)})$$

$$\theta^{(t+1)} = \theta^{(t)} - \rho . \frac{\partial f(\theta^{(t)})}{\partial \theta^{(t)}}$$

$\rho > 0$: *learning rate* (tốc độ học)



2. Gradient Descent (tt.)

❑ Một số hàm mất mát (loss function)

- *Regression loss*

Mean square error/Quadratic loss/L2 loss: $MSE = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$

Mean absolute error/L1 loss: $MAE = \frac{1}{m}\sum_{i=1}^{m}|y_i - \hat{y}_i|$

Mean bias error: $MBE = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)$

- *Classification loss*

Hinge loss/Multi class SVM loss, Cross entropy loss, …

2. Gradient Descent (tt.)

❑ Sử dụng ma trận giả nghịch đảo

Training set: $T = \{t^{(i)}\}_{i=1}^{m}, t^{(i)} =< x^{(i)}, y^{(i)} >$

input $x^{(i)} \in R^n$ output $y^{(i)} = y_i \in R$

$$X = \begin{pmatrix} x^{(1)} \\ x^{(2)} \\ \vdots \\ x^{(m)} \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{pmatrix} \qquad Y = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

bias

Đặt: $x = (x_1, \cdots, x_n) \in R^n, \hat{y} = f(x) = \sum_{j=1}^{n} w_j x_j + w_0$

$$w^T = (w_0, w_1, \cdots, w_n) \in R^{(n+1)}$$

$$\hat{x} = (\boxed{x_0}, x_1, \cdots, x_n) \in R^{(n+1)}, x_0 = 1, \hat{y} = \hat{x}.w$$

$$\hat{x}_i = (1, x^{(i)}) \in R^{(n+1)}, y_i = y^{(i)}, \forall i = 1, m$$

2. Gradient Descent (tt.)

❑ Sử dụng ma trận giả nghịch đảo

Training set:

$$\hat{X} = \begin{pmatrix} \hat{x}_1 \\ \vdots \\ \hat{x}_m \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m1} & \cdots & x_{mn} \end{pmatrix} \qquad Y = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}$$

Tìm vectơ cột: $w = (w_0, w_1, \cdots, w_n)^T$ sao cho $\hat{y} = \hat{x}.w \approx y$ tốt nhất

Hàm mất mát (*loss function*): $L(w) = \frac{1}{2m}\sum_{i=1}^{m}(y_i - \hat{x}_i w)^2$

Tìm optimal point: $w^* = \arg\min_w L(w)$

2. Gradient Descent (tt.)

❑ Sử dụng ma trận giả nghịch đảo

$$\frac{\partial L(w)}{\partial w} = \frac{1}{m}\hat{X}^T(\hat{X}.w - Y) = 0$$

Giải hệ phương trình, tìm w:

$$\boxed{\underbrace{\hat{X}^T.\hat{X}}_{\mathbf{A}}.w = \underbrace{\hat{X}^T.Y}_{\mathbf{B}}}$$

- Nếu $\hat{X}^T.\hat{X}$ khả nghịch: $w = (\hat{X}^T.\hat{X})^{-1}.\hat{X}^T.Y$

- Nếu $\hat{X}^T.\hat{X}$ KHÔNG khả nghịch: $w = (\hat{X}^T.\hat{X})^{\dagger}.\hat{X}^T.Y$

  với $(\hat{X}^T.\hat{X})^{\dagger}$ là ma trận *giả nghịch đảo* của $\hat{X}^T.\hat{X}$

2. Gradient Descent (tt.)

- **Sử dụng ma trận giả nghịch đảo**

Hàm mất mát (*loss function*): $L(w) = \frac{1}{2m}\sum_{i=1}^{m}(y_i - \hat{x}_i w)^2$

Tìm optimal point: $w^* = \arg\min_w L(w)$

Xét:

$$\hat{X} = \begin{pmatrix} \hat{x}_1 \\ \vdots \\ \hat{x}_m \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & \dots & x_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m1} & \dots & x_{mn} \end{pmatrix} \qquad Y = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}$$

2. Gradient Descent (tt.)

- **Sử dụng ma trận giả nghịch đảo**

$$L(w) = \frac{1}{2m}\sum_{i=1}^{m}(y_i - \hat{x}_i w)^2 \qquad m = |T|$$

- Đạo hàm riêng của L theo w_i

$$L(w_i) = (y_i - \hat{x}_i w)^2 = (y_i - \sum_{j=0}^{n}\hat{x}_j w_j)^2 = (y_i - (\hat{x}_i w_i + \underbrace{\sum_{j \neq i}\hat{x}_j w_j}_{C_i}))^2 =$$

$$= (y_i - (\hat{x}_i w_i + C_i))^2 = y_i^2 - 2y_i(\hat{x}_i w_i + C_i) + (\hat{x}_i w_i + C_i)^2 =$$

$$= y_i^2 - 2y_i\hat{x}_i w_i - 2y_i C_i + \hat{x}_i^2 w_i^2 + 2\hat{x}_i w_i C_i + C_i^2 =$$

$$= -2y_i\hat{x}_i w_i + \hat{x}_i^2 w_i^2 + 2\hat{x}_i w_i C_i + D_i$$

2. Gradient Descent (tt.)

- **Sử dụng ma trận giả nghịch đảo**
  - Đạo hàm riêng của L theo w_i

$$L(w_i) = -2y_i\hat{x}_i w_i + \hat{x}_i^2 w_i^2 + 2\hat{x}_i w_i C_i + D_i$$

$$\frac{\partial L(w_i)}{\partial w_i} = -2y_i\hat{x}_i + 2\hat{x}_i^2 w_i + 2\hat{x}_i C_i = 2\hat{x}_i.(\sum_{j=0}^{n} \hat{x}_j w_j - y_i)$$

$$\frac{\partial L(w)}{\partial w} = \begin{pmatrix} \frac{\partial L(w)}{\partial w_1} \\ \frac{\partial L(w)}{\partial w_2} \\ \vdots \\ \frac{\partial L(w)}{\partial w_n} \end{pmatrix} = \frac{1}{m}\hat{X}^T(\hat{X}.w - Y)$$

Tài liệu tham khảo

Vũ Hữu Tiệp, *Machine Learning cơ bản*, 2018