ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA

TOÁN ỨNG DỤNG CÔNG NGHỆ THÔNG TIN

Khoa Công Nghệ Thông Tin
TS. Nguyễn Văn Hiệu

Chuyên đề tối ưu hóa

Bài 1: Gradient Descent

Nội dung

1. Giới thiệu
2. Gradient descent cho hàm một biến
3. Gradient descent cho hàm nhiều biến
4. Phương pháp khắc phục hạn chế của Gradient descent
5. Các biến thể của Gradient descent
6. Bài tập

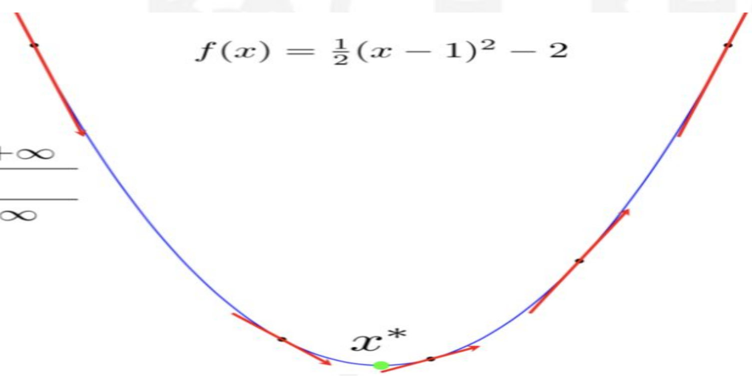
Giới thiệu

Bài toán tối ưu:

- Tìm giá trị lớn nhất hoặc nhỏ nhất của hàm số
- Bài toán có nhiều ứng dụng, đặc biệt ngành khoa học máy tính

Ví dụ

x	$-\infty$		1		$+\infty$
$f'(x)$		$-$	0	$+$	
$f(x)$	$+\infty$	$\searrow$	-2	$\nearrow$	$+\infty$



Một số vấn đề:

- x^* là nghiệm cục bộ thì $f'(x^*) = 0$
- Bên trái x^*: $f'(x^*) < 0$; Bên phải x^*: $f'(x^*) > 0$; tại x^*: $f'(x^*) = 0$

Giới thiệu

- Tìm nghiệm tối ưu toàn cục của hàm số:
  - Thường khó khăn hoặc không thể
  - Tìm nghiệm tối ưu cục bộ
- Tìm nghiệm tối ưu cục bộ của hàm số:
  - Đạo hàm của hàm số đó phải bằng 0.
  - Khó khăn với bài toán với dữ liệu lớn hoặc chiều dữ liệu lớn
- Hướng tiếp cận
  - Xuất phát từ một điểm (xem gần với nghiệm bài toán)
  - Dùng phương pháp lặp để dịch chuyển điểm đó đến nghiệm cần tìm

  ⇒ Phương pháp Gradient descent

Gradient descent cho hàm một biến

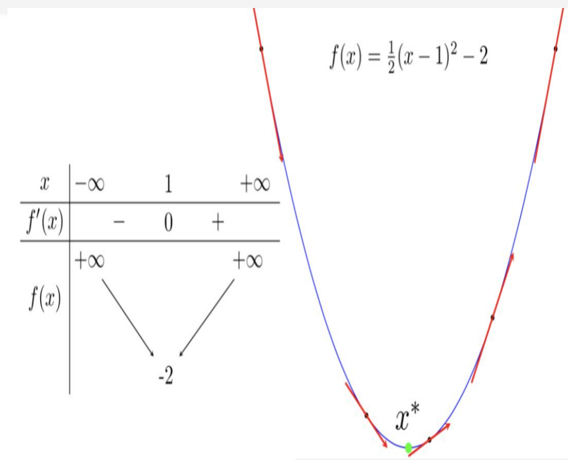
Xem ví dụ trước

- Nếu $f'(x_t) > 0$, thì x_t nằm bên phải x^*. Cần dịch chuyển sang trái

$$x_{t+1} = x_t + \Delta = x_t - \alpha . f'(x_t)$$

- Nếu $f'(x_t) < 0$, thì x_t nằm bên trái x^*. Cần dịch chuyển sang phải

$$x_{t+1} = x_t + \Delta = x_t - \alpha . f'(x_t)$$



- **Thuật toán Gradient descent:**
  - Dự đoán một điểm khởi tạo $x_t = x_0$
  - Cập nhật x_t đến đạt đến kết quả chấp nhận

$$x_t := x_t - \alpha f'(x_t)$$

- Khái niệm “Gradient descent” chính dấu “-” : ngược hướng đàm hàm

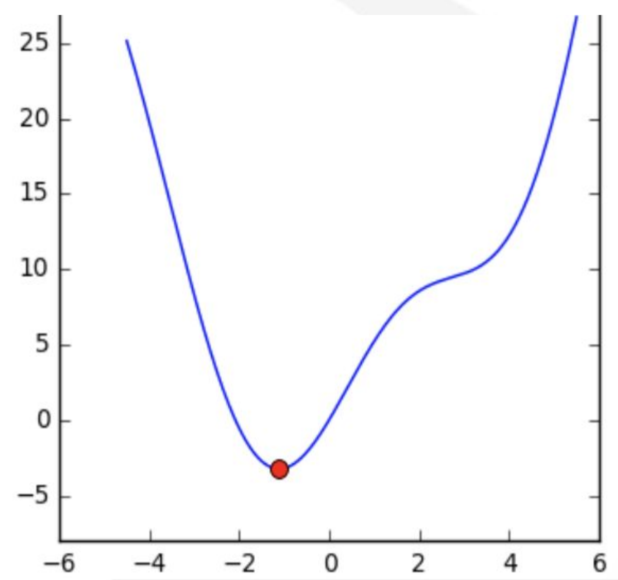
Gradient Descent cho hàm một biến(tiếp)

- Ví dụ tối ưu hàm số $f(x) = x^2 + 5 \,.\sin(x)$
- Khó khăn: $f'(x) = 2.x + 5\,.\,cos(x) = 0$
- Gradient descent: cho điểm $x_t = x_0$ và cập nhất

$$x_t := x_t - \alpha(2.x + 5.\cos(x))$$

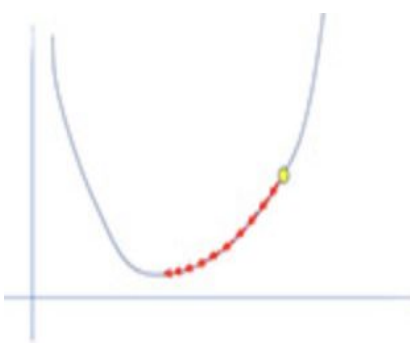
- Demo:

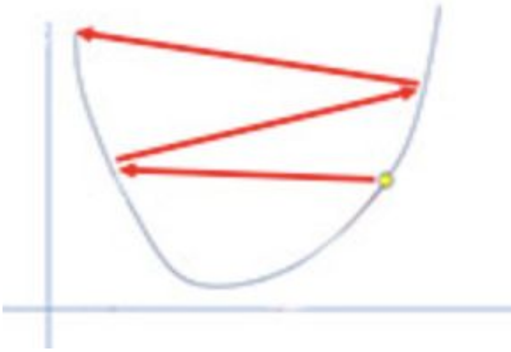
```python
def myGD1(alpha, x0, gra = 1e-3, loop = 1000):
    x = [x0]
    for it in range(loop):
        x_new = x[-1] - alpha*grad(x[-1])
        if abs(grad(x_new)) < gra:
            break
        x.append(x_new)
    return (x, it)
```

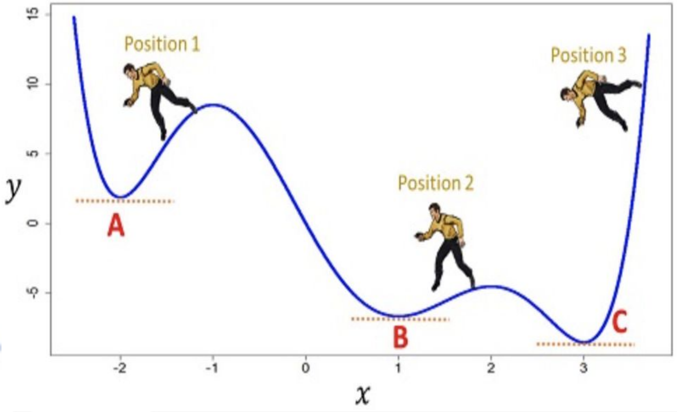


Gradient Descent cho hàm một biến(tiếp)

- Điều kiện dừng của Gradient Descent
  - Giới hạn số bước lặp
  - So sánh giá trị hàm của nghiệm tại 2 lần cấp nhật
  - Kiểm tra giá trị tuyệt đối của Gradient
- Tốc độ hội tụ của Gradient Descent:
  - Phụ thuộc vào điểm khởi tạo
  - Phụ thuộc vào chỉ số Alpha







Gradient Descent cho hàm đa biến

- Bài toán:
  - Tối ưu cho hàm $f(\theta), \theta = (\theta_0, \theta_1, \ldots, \theta_n)^T$
- Đạo hàm tại điểm θ: $\Delta_\theta \mathrm{f}(\theta)$
- **Thuật toán** (tương tự hàm một biến)
  - Dự đoán một điểm khởi tạo θ
  - Cập nhật θ đến khi nhận được kết quả chấp nhận được

$$\theta = \theta - \alpha \cdot \Delta_\theta f(\theta)$$

Gradient Descent cho hàm đa biến(tiếp)

- Bài toán Linear Regression (nhắc lại)
- Hàm mất mát

$$L(w) = \frac{1}{2N}\sum_{i=1}^{N}\left(\bar{X}^{(i)} \cdot w - y^{(i)}\right)^2$$
$$= \frac{1}{2N}\left\|\bar{X}.w - y\right\|_2^2$$

- Các ký hiệu:

$$y = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix}, w = \begin{pmatrix} w_0 \\ \vdots \\ w_n \end{pmatrix}, \quad \bar{X} = \begin{pmatrix} 1 & x_1^{(1)} & \dots & x_n^{(1)} \\ 1 & x_1^{(2)} & \dots & x_n^{(2)} \\ \vdots & \vdots & \dots & \vdots \\ 1 & x_1^{(N)} & \dots & x_n^{(N)} \end{pmatrix}$$

- Đạo hàm:

$$\Delta_w L(w) = \frac{1}{N}\bar{X}^T\left(\bar{X}.w - y\right)$$

Gradient Descent cho hàm đa biến(tiếp)

- Phương pháp truyền thống

$$\Delta_w L(w) = \frac{1}{N}\bar{X}^T(\bar{X}.w - y) = 0$$

$$\implies w = \left(\bar{X}^T\bar{X}\right)^{-1}\bar{X}^T y$$

- Vấn đề khó khăn
  - N và n lớn
  - ...

⇒ Sử dụng Gradient descent

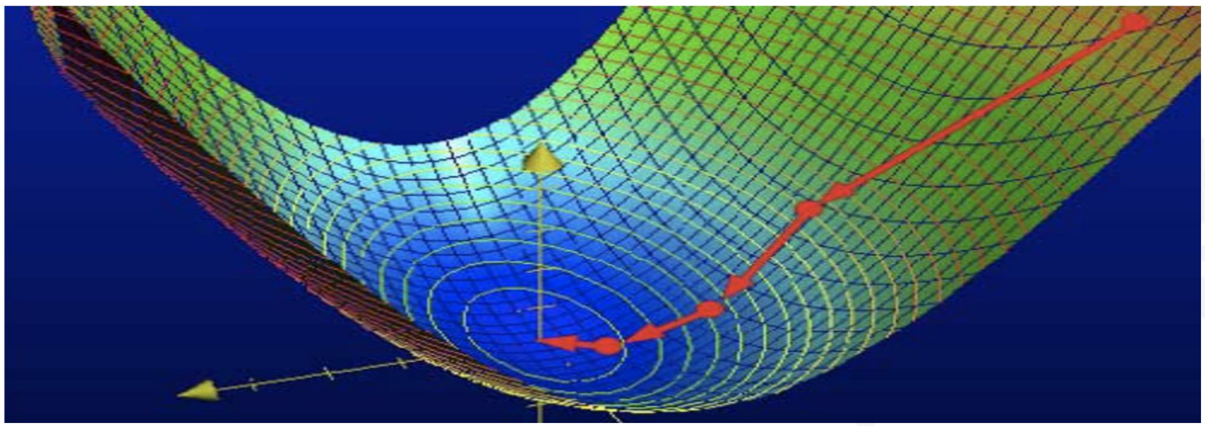
Gradient Descent cho hàm đa biến(tiếp)

- Cài đặt Gradient descent

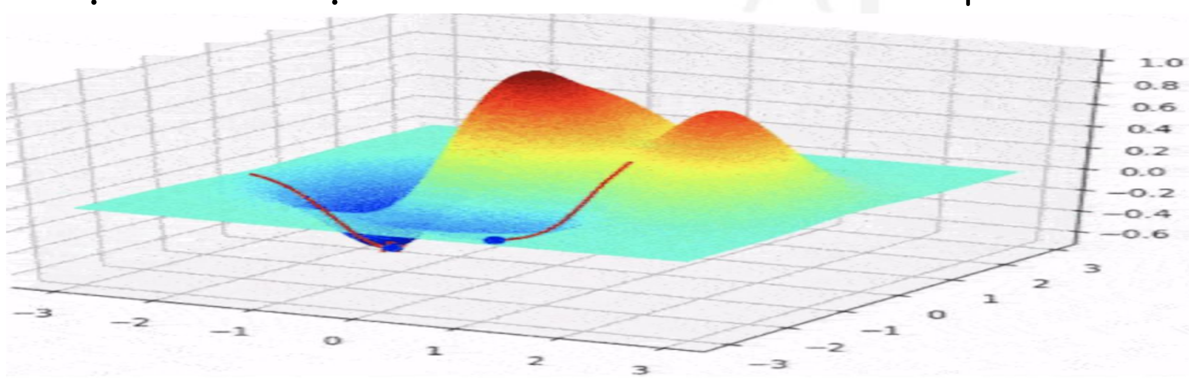
```python
def grad(w):
    N = Xbar.shape[0]
    return 1/N * Xbar.T.dot(Xbar.dot(w) - y)

def l(w):
    N = Xbar.shape[0]
    return .5/N*np.linalg.norm(Xbar.dot(w)-y, 2)**2

def myGradientDescent(w_init, grad, alpha, loop = 1000, esilon = 1e-4):
    w = [w_init]
    for i in range(loop):
        w_new = w[-1] - alpha*grad(w[-1])
        if np.linalg.norm(grad(w_new))/len(w_new) < esilon:
            break
        w.append(w_new)
    return (w, i)
```

Gradient descent với hàm đa biến(tiếp)

- Điểm xét sẽ lăn từ từ xuống hố và dừng lại ở đáy hố



- Tùy thuộc vào chọn điểm ban đầu mà kết quả có thể khác nhau

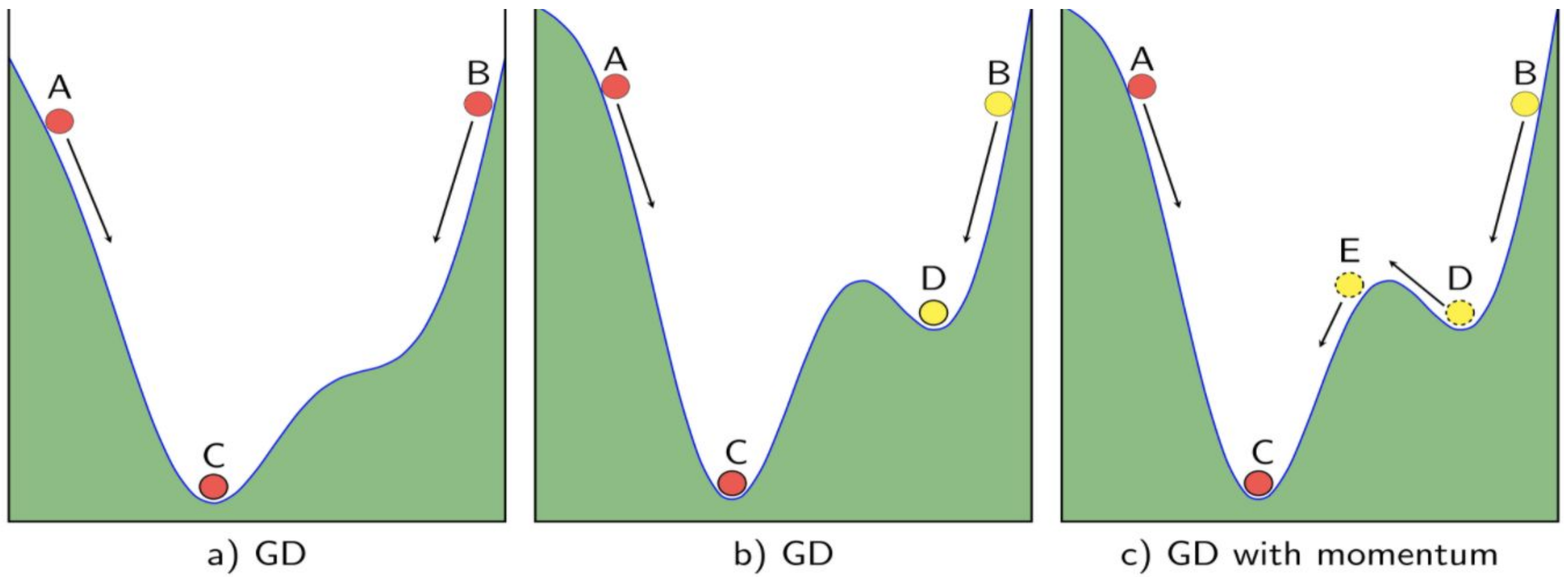


Phương pháp khắc phục Gradient Descent

- Gradient Descent với Momentum
- Nesterov Accelerated Gradient (NAG)

Gradient Descent với Momentum(đà)

- Vấn đề



- B có vận tốc đủ lớn khi di chuyển đến D, và theo đà sẽ di chuyển sang bên trái. Nếu vận tốc lớn hơn nữa thì sẽ di chuyển đến E và dừng ở C

Gradient Descent với Momentum(đà)

- Thuật toán:
  - Dự đoán một điểm khởi tạo và vận tốc ban đầu

$$\theta_0 \; ; \; v_0$$

  - Cập nhật đến đạt kết quả chấp nhận

$$\theta := \theta - v_t$$

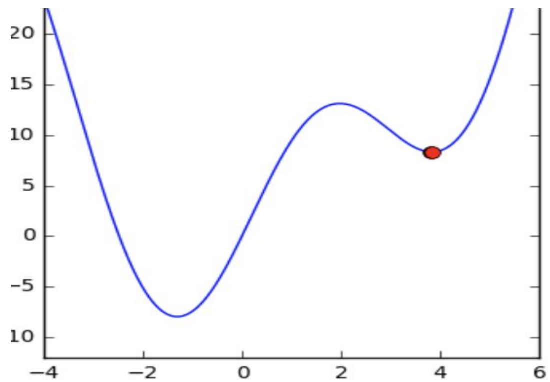
- v_t là vận tốc vật lý:
  - Thông tin về độ dốc (tức đạo hàm - **gradient**)
  - Thông tin về đà (tức vận tốc của v_{t-1} - **momentum**)

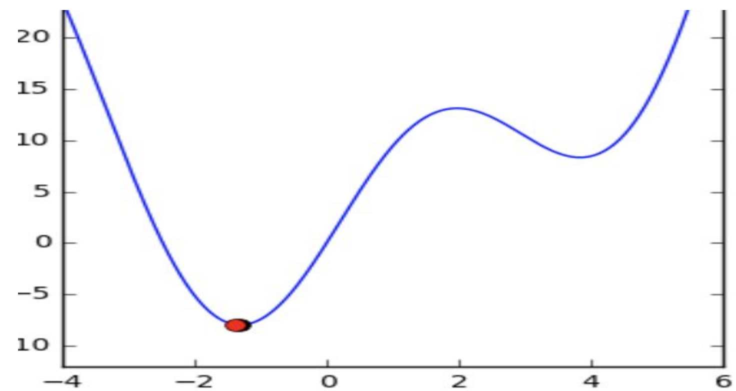
$$v_t = \beta \, . \, v_{t-1} + \alpha . \, \Delta_\theta J(\theta)$$

Gradient Descent với Momentum(đà)

- Ví dụ:

$$f(x) = x^2 + 10.\sin(x)\,,\quad x_0 = 5,\, \alpha = 0.1$$

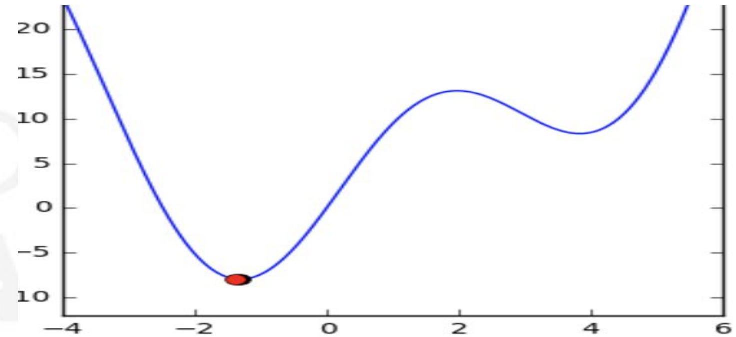




- Demo

```
def GD_momentum(theta_init, alpha=0.1, beta=0.9):
    theta = [theta_init]
    v_old = np.zeros_like(theta_init)
    for it in range(1000):
        v_new = beta*v_old + alpha*grad(theta[-1])
        theta_new = theta[-1] - v_new
        theta.append(theta_new)
        v_old = v_new
    return (theta,it)
```

Nesterov Accelerated Gradient (NAG)



- Gradient Descent với Momentum
  - Giúp trượt quá điểm cục bộ
  - Hội tụ rất chậm khi về đích
- Phương pháp NAG
  - Dự đoán một điểm khởi tạo và vận tốc ban đầu

$$\theta_0 \; ; \; v_0$$

  - Cập nhật đến đạt kết quả chấp nhận

$$\theta := \theta - v_t$$

$$v_t = \beta \, . \, v_{t-1} + \alpha . \, \Delta_\theta J(\theta - \beta \, . \, v_{t-1})$$

Các biến thể Gradient Descent

Batch Gradient descent

- Mỗi vòng lặp, dùng tất cả dữ liệu để tính gradient
- 1 Epoch: mỗi lần duyệt qua tất cả dữ liệu
- Khó khăn:
  - Dữ liệu huấn luyện quá lớn (như facebook)
  - Online Learning: dữ liệu cập nhật liên tục
- Ví dụ

$$L(w) = \frac{1}{2N}\sum_{i=1}^{N}\left(\bar{X}^{(i)} \cdot w - y^{(i)}\right)^2$$
$$= \frac{1}{2N}\left\|\bar{X}.w - y\right\|_2^2$$

$$\Delta_w L(w) = \frac{1}{N}\bar{X}^T\left(\bar{X}.w - y\right)$$

Các biến thể Gradient Descent

Stochastic Gradient descent

- Mỗi vòng lặp, dùng một mẫu dữ liệu để tính gradient
- Stochastic gradient descent có N lần cập nhật dữ liệu là 1 epoch
- Quy tắc cập nhật SGD:

$$\theta = \theta - \alpha \cdot \Delta_\theta J\left(\theta, x^{(i)}, y^{(i)}\right)$$

$\left(x^{(i)}, y^{(i)}\right)$
cặp dữ liệu (input, label)

$$L\left(w, \bar{X}^{(i)}, y^{(i)}\right) = \frac{1}{2}\left(\bar{X}^{(i)} \cdot w - y^{(i)}\right)^2 = \frac{1}{2}\left\|\bar{X}^{(i)} \cdot w - y^{(i)}\right\|_2^2$$

$$\Delta_w L\left(w, \bar{X}^{(i)}, y^{(i)}\right) = \bar{X}^{(i)T}\left(\bar{X}^{(i)} w - y^{(i)}\right)$$

Các biến thể Gradient Descent

Mini_batch Gradient descent

- Mini-batch gradient descent: mỗi vòng lặp, dùng một số lượng nhỏ k mẫu dữ liệu để tính gradient. Con số k được gọi là batch size
- Mini-batch - sử dụng lượng dữ liệu k>1 và nhỏ k < N
- Xáo trộn dữ liệu trước lúc chia Mini-batch
- Quy tắc cập nhật Mini-batch GD:

$$\theta = \theta - \alpha \cdot \Delta_\theta J(\theta, x_{i:i+n}, y_{i:i+n})$$

- Trong thuật ngữ hiện đại, Stochastic gradient descent và Mini-batch gradient descent là như nhau: mỗi vòng lặp dùng k mẫu, k có thể là 1.

Bài tập 1

Cài đặt thuật toán Stochastic Gradient descent cho bài toán Linear Regression

Bài tập 2

Cài đặt thuật toán **Mini Batch Gradient descent cho bài toán Linear Regression**

Bài tập 3

Cài đặt thuật toán **Gradient descent cho bài toán MF**

Tài liệu

1. https://ruder.io/optimizing-gradient-descent/index.html#stochasticgradientdescent
2. Gradient Descen - Andrew Ng

Cám Ơn!

Khoa Công Nghệ Thông Tin
TS. Nguyễn Văn Hiệu

Chuyên đề tối ưu hóa

Bài 2: Phương pháp Lagrange

Phương pháp Lagrange

- Bài toán

$$\begin{cases} f(x) \rightarrow & \min(/max) \\ g_i(x) = & b_i,\ i\ = 1, \ldots, m \\ x \in R^n & \end{cases}$$

- Bước 1: Lập hàm Lagrange

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i [b_i - g_i(x)]$$

- Bước 2: Giải hệ phương trình

$$0 = \frac{\mathrm{d}L}{\mathrm{d}x_j} = \frac{\mathrm{d}f}{\mathrm{d}x_j} - \sum_{i=1}^{m} \lambda_i \frac{\mathrm{d}g_i}{\mathrm{d}x_j},\ j\ = 1, \ldots, n$$

$$0 = \frac{\mathrm{d}L}{\mathrm{d}\lambda_j} = b_j - g_j(x),\ j\ = 1, \ldots, m$$

Phương pháp Lagrange

- Bước 3: từ tập điểm cực trị của L
  - $d^2L > 0$, thì cực trị của L là điểm cực tiêu
  - $d^2L < 0$, thì cực trị của L là điểm cực đại

Cám Ơn!