MACHINE LEARNING

Cao Văn Chung
cvanchung@hus.edu.vn

Informatics Dept., MIM, HUS, VNU Hanoi

Logistic Regression

- Mô hình hồi quy Logistic
- Regularization
- Phương pháp giải
    - Phương pháp Gradient Descent
    - Phương pháp Newton-Raphson
- Ví dụ

Logistic Regression

- Hồi quy logistic được sử dụng trong phân loại nhị phân, $y \in \{0, 1\}$. Mặc dù tên của phương pháp có từ "Hồi quy" (Regression) nhưng thường áp dụng cho bài toán phân lớp (classification).
- Là mô hình phân biệt: mô hình trực tiếp $P(y|x)$.
- Mô hình hồi quy logistic được sử dụng rộng rãi trong nhiều bài toán thống kê và học máy.

Logistic Regression

- Xét bài toán phân loại nhị phân, mỗi đối tượng $x \in X$ - biến quan sát, cần được phân loại vào một trong hai lớp $y \in \{0, 1\}$.
- Mô hình dự báo $h_\theta(x)$ trường hợp này được chọn như sau

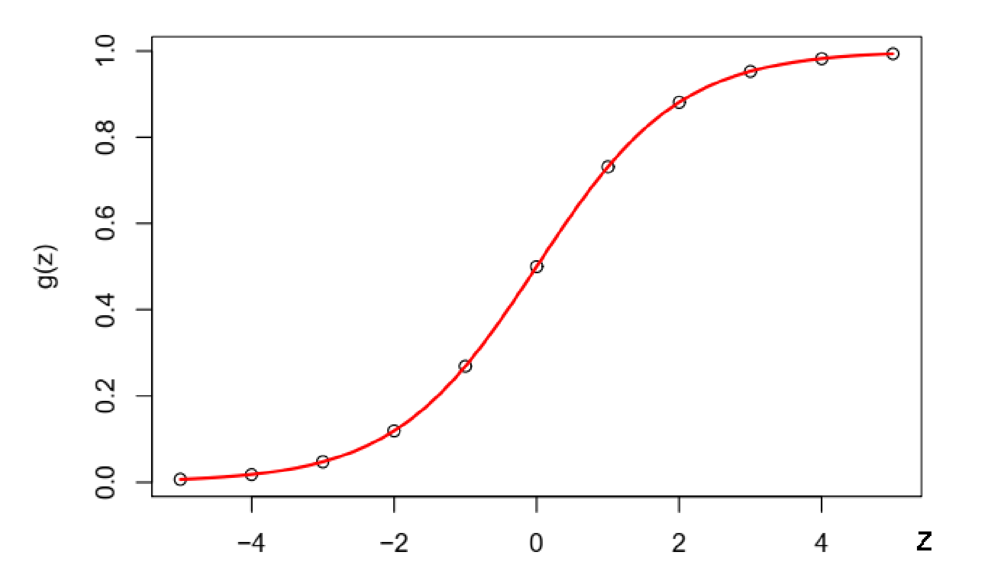
$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + \exp\big(-\theta^T x\big)}, \tag{1}$$

ở đây,

$$g(s) = \frac{1}{1 + \exp\big(-s\big)}$$

được gọi là hàm sigmoid, hoặc nói chung gọi là hàm logistic.

Logistic Regression



Logistic Regression

Một số tính chất của hàm sigmoid

- $g(s) \to 1$ khi $s \to \infty$.
- $g(s) \to 0$ khi $s \to -\infty$.
- $g(s)$ và $h(s)$ nhận giá trị trên $[0, 1]$.
- Đạo hàm của $g(s)$

$$g'(s) = \frac{e^{-s}}{(1+e^{-s})^2} = \frac{1}{1+e^{-s}} \frac{e^{-s}}{1+e^{-s}} = g(s)(1-g(s)).$$

Logistic Regression

- Để ý trong hồi quy tuyến tính, ta sử dụng

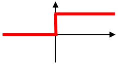
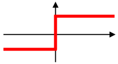
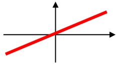
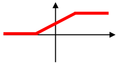
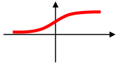
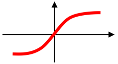
$$h_\theta(x) = \theta^T x = \bar{g}(\theta^T x), \quad \text{với} \quad g(s) \equiv s;$$

  trong hồi quy logistic

$$h_\theta(x) = g(\theta^T x).$$

- Cả hai phương pháp xây dựng trên biểu thức tuyến tính $\theta^T x$, nên cùng được xếp vào nhóm mô hình phân loại tuyến tính.
- Các hàm $g(s)$ để chuyển trạng thái trong các mô hình được gọi là hàm kích hoạt *activation function*. Tùy theo mô hình có thể sử dụng một số dạng hàm activation khác nhau.

Logistic Regression

Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multi-layer Neural Networks	

Logistic Regression

- Do y chỉ nhận giá trị 0 hoặc 1 (xung khắc), ta có mô hình hồi quy logistic:

$$\begin{aligned} P(y=1|x;\theta) &= h_\theta(x) \\ P(y=0|x;\theta) &= 1-h_\theta(x) \end{aligned} \tag{2}$$

  với $\theta \in \mathbb{R}^{d+1}$ là tham số của mô hình.
- Giả sử đã biết vector tham số θ, ta sử dụng mô hình để phân loại như sau:
  - Xếp đối tượng x vào lớp 1 nếu

$$P(y=1|x;\hat{\theta}) > P(y=0|x;\hat{\theta}) \Leftrightarrow h_{\hat{\theta}}(x) > \frac{1}{2} \Leftrightarrow \hat{\theta}^T x > 0.$$

  - Ngược lại xếp x vào lớp 0.

Logistic Regression

- Ta có thể kết hợp hai phương trình của (2) và viết gọn xác suất của lớp y dưới dạng
$$P(y|x;\theta) = \big(h_\theta(x)\big)^y \big(1 - h_\theta(x)\big)^{1-y}. \qquad (2b)$$
Thật vậy, khi $y = 1$, phần thứ hai của vế phải sẽ triệt tiêu, trong khi nếu $y = 0$ phần thứ nhất sẽ bị triệt tiêu.
- Xét bộ training set với $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\} \in \mathbb{R}^{d\times N}$ và $\mathbf{y} = \{y_1, y_2, \dots, y_N\}$, ta áp dụng phương pháp ước lượng hợp lý cực đại, tức là tìm tham số θ để cực đại hóa biểu thức:
$$P(\mathbf{y}|\mathbf{X};\theta) \longrightarrow \max_\theta \quad \text{tức là tìm} \quad \hat{\theta} = \arg\max_\theta \mathrm{P}(\mathbf{y}|\mathbf{X};\theta).$$

Logistic Regression

- Giả sử training dataset được sinh độc lập lẫn nhau, khi đó hàm hợp lí của dữ liệu với tham số θ là

$$L(\theta) = P(\mathbf{y}|\mathbf{X};\theta) = \prod_{i=1}^{N} P(y_i|\mathbf{x}_i;\theta) = \prod_{i=1}^{N} h_\theta(x_i)^{y_i}(1 - h_\theta(x_i))^{1-y_i}.$$

- Hàm log-hợp lí (log-likelihood function) là

$$\begin{aligned} \ell(\theta) = \log L(\theta) &= \log\left(\prod_{i=1}^{N} h_\theta(x_i)^{y_i}(1 - h_\theta(x_i))^{1-y_i}\right) \\ &= \sum_{i=1}^{N} \left[y_i \log h_\theta(x_i) + (1 - y_i)\log(1 - h_\theta(x_i))\right]. \end{aligned} \tag{3}$$

Logistic Regression

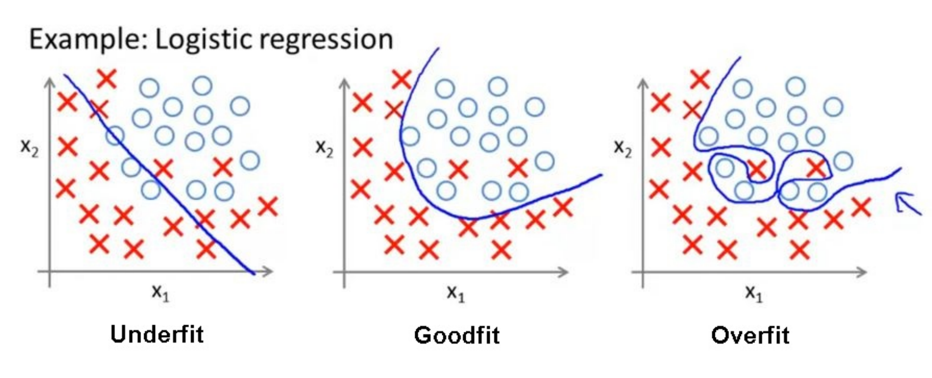
- Dễ dàng kiểm tra giá trị hàm log-likelihood $\ell(\theta)$ luôn âm. Do đó thay cho cực đại hóa $\ell(\theta) \longrightarrow \max_\theta$, ta sẽ giải bài toán

$$-\ell(\theta) \longrightarrow \min_\theta.$$

- Tuy nhiên tương tự hồi quy tuyến tính, hồi quy logistic dựa trên tổ hợp $\theta^T x$. Do đó nếu giải trực tiếp $-\ell(\theta) \longrightarrow \min_\theta$, sẽ có chung một số đặc điểm
  - Không ổn định (nhạy cảm với nhiễu dữ liệu) - Tham khảo trong phần bài giảng hồi quy tuyến tính.
  - Hệ quả là dễ xảy ra Overfit (quá khớp dữ liệu) - Tham khảo phần đọc thêm.

Logistic Regression

Hình: Các trạng thái *underfit*, *goodfit* và *overfit* của mô hình



Regularization

- Để tăng tính ổn định khi giải bài toán cực trị $-\ell(\theta) \longrightarrow \min_\theta$ và tránh khả năng overfit, chúng ta bổ sung phần hiệu chỉnh. Cụ thể ta giải bài toán cực tiểu

$$\begin{aligned} J(\theta) &= (-\ell(\theta) + \alpha R(\theta)) \longrightarrow \min_\theta \\ \text{hay tìm} \quad \hat{\theta} &= \arg\min_\theta \{-\ell(\theta) + \lambda R(\theta)\} \end{aligned} \tag{4}$$

trong đó $\lambda \geq 0$ - tham số hiệu chỉnh; $R(z) : \mathbb{R}^{d+1} \mapsto \mathbb{R}^+$ là phiếm hàm hiệu chỉnh.

Regularization

Một số dạng hàm hiệu chỉnh thường gặp

- $R(\theta) \equiv 0$, chúng ta có bài toán cực tiểu nguyên bản trong hồi quy Logistic.
- $R(\theta) = \|\theta\|_1 = \sum_{i=0}^{d} |\theta_i|$ - ta có mô hình hồi quy logistic hiệu chỉnh dạng L_1.
- $R(\theta) = \|\theta\|_2^2 = \sum_{i=0}^{d} \theta_i^2$ - ta có mô hình hồi quy logistic hiệu chỉnh dạng L_2.
- $R(\theta) = \sum_{j=0}^{d} \log\left(\frac{e^{\theta_j} + e^{-\theta_j}}{2}\right)$ thì ta có mô hình hồi quy logistic hiệu chỉnh dạng hyperbolic-L_1.

Giải bài toán tối ưu

- Chúng ta cần tìm

$$
\begin{aligned}
\hat{\theta} &= \arg\min_{\theta} \{-\ell(\theta) + \lambda R(\theta)\} \\
&= \arg\min_{\theta} \left\{ \sum_{i=1}^{N} [y_i \log h_\theta(x_i) + (1-y_i)\log(1-h_\theta(x_i))] + \lambda R(\theta) \right\}.
\end{aligned}
$$

- Để giải các bài toán cực trị như trên, một số phương pháp lặp thường được dùng
  - Phương pháp *Đối Gradient Ngẫu nhiên - Stochastic Gradient Descent (SGD)* và.
  - Phương pháp *Newton - Newton-Raphson*
- Các phương pháp này giải trực tiếp bài toán tối ưu nên đôi khi được gọi là các phương pháp tối ưu, để phân biệt với các phương pháp đưa về giải phương trình $\nabla J(\theta) = 0$. Tuy nhiên đây cũng là các phương pháp lặp.

Gradient Descent

- Giả sử từ khởi đầu θ^0 bất kỳ, hiện tại ta có xấp xỉ θ. Khi đó theo phương pháp dạng Gradient Descent, θ ở bước tiếp theo sẽ được cập nhật qua công thức

$$\theta := \theta - \alpha \nabla J(\theta) \tag{5}$$

- với $\alpha > 0$ - hệ số học và gradient $\nabla J(\theta)$ của $J(\theta)$

$$\nabla J(\theta) = \left(\frac{\partial J}{\partial \theta_0}, \frac{\partial J}{\partial \theta_1}, \ldots, \frac{\partial J}{\partial \theta_d} \right).$$

  Ở đây dùng $-\alpha \nabla J(\theta)$ vì trong phương pháp GD, giá trị θ được cập nhật theo hướng ngược với vector Gradient (*gradient descent*).

- Mỗi thành phần của tham số $\theta = (\theta_j)_{j=0}^{d}$ được cập nhật theo công thức

$$\theta_j = \theta_j - \alpha \frac{\partial J}{\partial \theta_j} \quad j = 0, 1, \ldots, d.$$

Gradient Descent

- Trước khi đi vào chi tiết việc giải (5) bằng SGD, chúng ta phân tích biểu thức của $\nabla J(\theta)$ để hiểu rõ hơn tại sao chọn hàm kích hoạt Sigmoid cho hồi quy logistic.
- Xét trường hợp chỉ có một cặp dữ liệu (x, y) và không có hiệu chỉnh, đặt $z = h_\theta(x) = g(\theta^T x)$, ta có

$$J(\theta; \mathbf{x}, y) = -\left(y \log z + (1-y)\log(1-z)\right)$$

và gradient (đạo hàm theo θ) của nó là

$$\frac{\partial J(\theta; \mathbf{x}, y)}{\partial \theta} = \frac{\partial J}{\partial z}\frac{\partial z}{\partial \theta} = -\left(\frac{y}{z} - \frac{1-y}{1-z}\right)\frac{\partial z}{\partial \theta} = \frac{z-y}{z(1-z)}\frac{\partial z}{\partial \theta} \tag{6}$$

Gradient Descent

- Để dễ xử lý hơn, ta cần dạng hàm $z = g(\theta^T \mathbf{x})$ sao cho mẫu số bị triệt tiêu. Nếu đặt $s = \theta^T \mathbf{x}$ chúng ta sẽ có:

$$\frac{\partial z}{\partial \theta} = \frac{\partial z}{\partial s}\frac{\partial s}{\partial \theta} = \frac{\partial z}{\partial s}\frac{\partial(\theta^T \mathbf{x})}{\partial \theta} = \frac{\partial z}{\partial s}\mathbf{x}$$

- Thay vào (6) ta có

$$\frac{\partial J(\theta; \mathbf{x}, y)}{\partial \theta} = \frac{z - y}{z(1-z)}\frac{\partial z}{\partial s}\mathbf{x}. \tag{6b}$$

- Vậy để khử mẫu thức, ta cần $z = z(s)$ sao cho $\frac{\partial z}{\partial s} = z(1-z)$. Giải phương trình vi phân này ta sẽ thu được $z = g(s)$ là hàm dạng sigmoid.

Gradient Descent

- Thay $z = g(\theta^T x)$ vào công thức (6b), ta có

$$\frac{\partial J(\theta; \mathbf{x}, y)}{\partial \theta} = (z - y)\mathbf{x} = (h_\theta(\mathbf{x}) - y)\,\mathbf{x}$$

- Do đó, với mỗi thành phần thứ $j = 0, 1, \ldots, d$ của vector $\theta = (\theta_j)$ ta có

$$\left(\frac{\partial J(\theta; \mathbf{x}, y)}{\partial \theta}\right)_j = -\frac{\partial \ell(\theta)}{\partial \theta_j} = (h_\theta(\mathbf{x}) - y)\,x_j$$

- Vậy quá trình cập nhật $\theta = (\theta_j)$ theo (5) được thực hiện như sau

$$\theta_j := \theta_j - \alpha\,(h_\theta(\mathbf{x}) - y)\,x_j, \qquad j = 0, 1, \ldots d. \tag{5b}$$

Gradient Descent

Giả sử tập training data có N cặp $\{(x_i, y_i) | i = 1, 2, \dots N\}$, ta có hai cách tiếp cận để thực hiện cập nhật θ_j theo (5b)

(1) Tại mỗi bước, sử dụng *toàn bộ* dữ liệu $\{(x_i, y_i) | i = 1, 2, \dots N\}$ và áp dụng công thức của $\ell(\theta)$ trong (3), ta có công thức cập nhật θ

$$\theta_j := \theta_j - \alpha \sum_{i=0}^{N} (h_\theta(x_i) - y_i)\, x_{ij}, \qquad j = 0, 1, \dots d. \tag{5c}$$

Theo cách này, ta có phương pháp đối gradient theo loạt - *Batch Gradient Descent*.

Gradient Descent

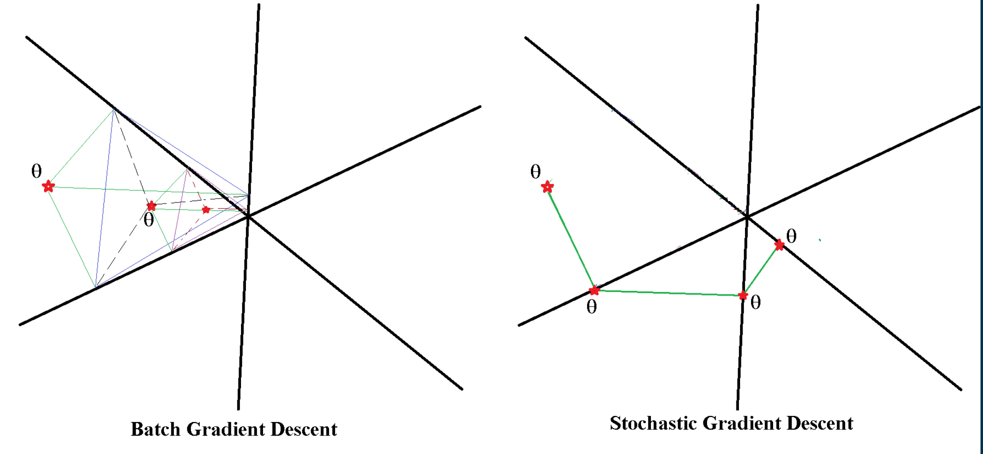
(2) Sử dụng *lần lượt* dữ liệu (x_i, y_i) để cập nhật θ tại mỗi bước

$$\theta_j := \theta_j - \alpha\,(h_\theta(x_i) - y_i)\,x_{ij}, \qquad j = 0, 1, \ldots d. \tag{5c}$$

- Lần lượt thực hiện với $i = 1, 2, \ldots, N$, tức là tất cả các cặp dữ liệu đều tham gia cập nhật θ đơn lẻ;
- Sau khi sử dụng hết N cặp dữ liệu, cần xáo trộn lại thứ tự các cặp (x_i, y_i) và thực hiện lại từ đầu.

Theo cách này, ta có phương pháp đối gradient ngẫu nhiên - *Stochastic Gradient Descent - SGD*.
Hình ở trang sau minh họa ý tưởng tiếp cận của *Batch Gradient Descent* và *Stochastic Gradient Descent*.

Gradient Descent



Batch Gradient Descent

Thuật toán Batch Gradient Descent cho Hồi quy Logistic:

```
function [θ] = BatchGradientDescent_LR(X, Y, α)
# Tìm hệ số hồi quy Logistic θ bằng phương pháp Batch Gradient Descent
    [N, d] = size(X);
    θ = zeros(d, 1);
    repeat
        for j = 1 to d
            θ_j = θ_j − α ∑_{i=0}^{N} (h_θ(x_i) − y_i) x_ij;
        endfor
    until (STOP-condition true)
    return θ;
end
```

$$\theta_j = \theta_j - \alpha \sum_{i=0}^{N} \left(h_\theta(x_i) - y_i\right) x_{ij};$$

Stochastic Gradient Descent

Thuật toán Stochastic Gradient Descent cho Hồi quy Logistic:

```
function [θ] = StochasticGradientDescent_LR(X, Y, α)
# Tìm hệ số hồi quy Logistic θ bằng phương pháp Stochastic Gradient Descent
    [N, d] = size(X);
    θ = zeros(d, 1);
    repeat
        {k} = Permutation(N);
        for i_k = 1 to N
            for j = 1 to d
                θ_j = θ_j − α (h_θ(x_{i_k}) − y_{i_k}) x_{i_k j};
            endfor j
        endfor i_k
    until (STOP-condition true)
    return θ;
end
```

$$\theta_j = \theta_j - \alpha\left(h_\theta(x_{i_k}) - y_{i_k}\right) x_{i_k j};$$

Hiệu chỉnh L_2

- Áp dụng hiệu chỉnh L_2 cho hồi quy Logistic

$$J(\theta) = -\ell(\theta) + \frac{1}{2}\lambda \sum_{j=1}^{d} \theta_j^2 \tag{7}$$

- Áp dụng phương pháp Batch Gradient Descent

$$\theta_0 = \theta_0 - \alpha \sum_{i=0}^{N} [h_\theta(x_i) - y_i]$$

$$\theta_j = \theta_j - \alpha \sum_{i=0}^{N} [h_\theta(x_i) - y_i] \, x_{ij} - \lambda\theta_j, \quad j = 1, 2, \ldots d.$$

Hiệu chỉnh L_2

- Áp dụng phương pháp Stochastic Gradient Descent: với $i = 1, 2 \dots N$

$$\theta_0 = \theta_0 - \alpha \left[h_\theta(x_i) - y_i\right]$$
$$\theta_j = \theta_j - \alpha \left[h_\theta(x_i) - y_i\right] x_{ij} - \lambda\theta_j, \quad j = 1, 2, \dots d.$$

Lưu ý: Ta có thể thay $\ell(\theta)$ bởi $\frac{1}{N}\ell(\theta)$ (trung bình hợp lý) để giảm tích lũy sai số. Khi đó

$$\sum_{i=0}^{N} \left[h_\theta(x_i) - y_i\right] x_{ij} \qquad \text{được thay thế bởi} \qquad \frac{1}{N}\sum_{i=0}^{N} \left[h_\theta(x_i) - y_i\right] x_{ij}.$$

Phương pháp Newton-Raphson

- Người ta cũng thường dùng các thuật toán dạng Newton để giải bài toán cực tiểu hoá phiếm hàm $J(\theta)$ trong trường hợp phiếm hàm này khả vi cấp hai, hoặc có đạo hàm cấp một thỏa mãn một số điều kiện đặc biệt.
- Ý tưởng của các phương pháp dạng Newton:
  - Đưa bài toán tối ưu $J(\theta) \to \min_\theta$ về phương trình $f(\theta) = J'(\theta) = 0$;
  - Giả sử f khả vi (tức J khả vi cấp 2), từ ước lượng xấp xỉ

$$\left\| f'(\theta)\left[(\theta + \delta_\theta) - \theta\right] \right\| = |f(\theta + \delta_\theta) - f(\theta)| + o(\|\delta_\theta\|^2).$$

  - Giả sử $f(\theta^*) = 0$ (θ^* - nghiệm đúng), xấp xỉ hiện tại là θ, suy ra

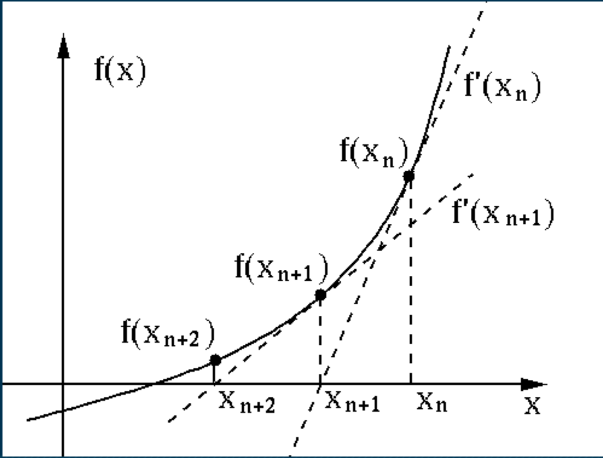
$$\left\| f'(\theta)\left[\theta - \theta^*\right] \right\| = |f(\theta) - f(\theta^*)| + o(\|\theta - \theta^*\|^2) = |f(\theta)| + o(\|\theta - \theta^*\|^2).$$

Phương pháp Newton-Raphson

- ...
  - Vậy giả sử xấp xỉ hiện tại $\theta^{(n)}$ đủ gần nghiệm θ^*, ta sẽ có
  $$f'(\theta^{(n)})\left[\theta^{(n)} - \theta^*\right] \approx f(\theta^{(n)}).$$
  - Từ gợi ý này ta sẽ tìm xấp xỉ tiếp theo $\theta^{(n+1)}$ sao cho
  $$f'(\theta^{(n)})\left[\theta^{(n)} - \theta^{(n+1)}\right] = f(\theta^{(n)}).$$
  - Nếu $f'(\theta^{(n)})$ không suy biến, ta viết lại
  $$\theta^{(n+1)} = \theta^{(n)} - \left[f'(\theta^{(n)})\right]^{-1} f(\theta^{(n)}).$$
  - Đây chính là phương pháp lặp dạng Newton.
- *Lưu ý:* Từ phân tích ta thấy phương pháp Newton hội tụ bậc hai.

Phương pháp Newton-Raphson

Hình: Minh họa phương pháp Newton cho trường hợp hàm 1 biến



Phương pháp Newton-Raphson

- Áp dụng cho bài toán hồi quy Logistic, ta cần giải phương trình

$$\frac{\partial J(\theta; \mathbf{x}, y)}{\partial \theta} = 0 \quad \Leftrightarrow \quad \frac{1}{N}\sum_{i=1}^{N}\left[h_\theta(x_i) - y_i\right]x_i = 0.$$

- Do $\theta = \left(\theta_0, \theta_1, \ldots \theta_d\right)^T$, có thể viết lại phương trình trên

$$f(\theta) = \nabla J(\theta) = \begin{pmatrix} \dfrac{\partial J(\theta; \mathbf{x}, y)}{\partial \theta_\mathbf{0}} \\ \dfrac{\partial J(\theta; \mathbf{x}, y)}{\partial \theta_\mathbf{1}} \\ \vdots \\ \dfrac{\partial J(\theta; \mathbf{x}, y)}{\partial \theta_\mathbf{d}} \end{pmatrix} = \frac{1}{N}\sum_{i=1}^{N}\begin{pmatrix} \left[h_\theta(x_i) - y_i\right]x_{i,0} \\ \left[h_\theta(x_i) - y_i\right]x_{i,1} \\ \vdots \\ \left[h_\theta(x_i) - y_i\right]x_{i,d} \end{pmatrix} = 0.$$

Phương pháp Newton-Raphson

Ta có $f'(\theta) = H(\theta) = (H_{ij}) = \left(\dfrac{\partial^2 J(\theta)}{\partial\theta_i \partial\theta_j}\right)$ - Ma trận Hessian của $J(\theta)$.

$$H_{ij} = \frac{\partial^2 J(\theta)}{\partial\theta_i\partial\theta_j} = \frac{\partial \sum_{k=1}^{N}\left[h_\theta(x_k) - y_k\right]x_{k,i}}{\partial\theta_j} = \sum_{k=1}^{N}\frac{\partial g(\theta^T x_k)}{\partial\theta_j}x_{k,i}$$

$$= \sum_{k=1}^{N}\left\{g(\theta^T x_k)\left[1 - g(\theta^T x_k)\right]x_{k,j}\right\}x_{k,i}$$

$$= \sum_{k=1}^{N} h_\theta(x_k)\left[1 - h_\theta(x_k)\right]x_{k,i}x_{k,j} \quad i,j = 1,\dots,d.$$

Phương pháp Newton-Raphson

Suy ra H là ma trận kích thước $(d+1) \times (d+1)$:

$$H = \sum_{k=1}^{N} h_\theta(x_k)\left[1 - h_\theta(x_k)\right] x_k x_k^T.$$

Với H, $\nabla J(\theta)$ như trên và xấp xỉ hiện tại θ, ta có công thức phương pháp Newton-Raphson

$$\theta = \theta - H^{-1}\left(\nabla J(\theta)\right).$$

Newton-Raphson hiệu chỉnh L_2

- Vector Gradient

$$\nabla J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\begin{pmatrix} \left[h_\theta(x_i) - y_i\right]x_{i,0} \\ \left[h_\theta(x_i) - y_i\right]x_{i,1} + \lambda\theta_1 \\ \vdots \\ \left[h_\theta(x_i) - y_i\right]x_{i,d} + \lambda\theta_d \end{pmatrix} = 0.$$

- Ma trận Hessian

$$H = \sum_{k=1}^{N} h_\theta(x_k)\left[1 - h_\theta(x_k)\right]x_k x_k^T + \lambda\begin{pmatrix} 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & \dots & 0 & 1 \end{pmatrix}.$$

So sánh các phương pháp

- Trong phương pháp đối gradient, ta cần chọn tốc độ học α, còn trong phương pháp Newton, ta không cần chọn tham số này.
- Phương pháp Newton thường hội tụ nhanh hơn phương pháp giảm gradient, chỉ cần một số bước lặp ít hơn để đạt được cực trị.
- Tuy nhiên, mỗi bước lặp của phương pháp Newton lại cần tính toán nhiều hơn nếu số chiều d của ma trận Hessian là lớn.
  - Mỗi bước lặp của phương pháp giảm gradient có độ phức tạp $O(d)$, trong khi mỗi bước lặp của phương pháp Newton có độ phức tạp $O(d^3)$.
  - Khi d lớn (ví dụ, lớn hơn 50,000) thì ta nên dùng phương pháp đối gradient, còn khi d nhỏ (ví dụ, nhỏ hơn 1,000) thì ta có thể tính được nghịch đảo của ma trận Hessian, do đó nên dùng phương pháp Newton.

Ví dụ: Lọc thư rác

- Tập dữ liệu Spambase cung cấp 4601 mẫu thư điện tử tiếng Anh dạng thư rác và không phải thư rác cho trong link: http://archive.ics.uci.edu/ml/datasets/Spambase .
- Tập dữ liệu này thường được dùng để đánh giá hiệu quả của các thuật toán lọc thư rác tự động.
- Một số tính chất của tập dữ liệu này:
  - Tính chất của dữ liệu: đa chiều.
  - Kích thước mẫu: 4601
  - Kiểu của đặc trưng: số nguyên và số thực.
  - Số đặc trưng: 57.

Ví dụ: Lọc thư rác

Mô tả danh sách các đặc trưng

- 48 số thực trong đoạn [0, 100] thuộc kiểu word_freq_WORD là phần trăm từ trong thư là WORD, tức là 100*(số lần từ WORD xuất hiện trong thư) / tổng số từ trong thư
  - Mỗi "từ" là bất kì một chuỗi kí tự nào, có thể là từ theo nghĩa thông thường hoặc một kí hiệu (token).
  - Một số từ thuộc 48 từ được xét: make, address, all, 3d, our, money, technology, conference..
- 6 số thực trong đoạn [0, 100] thuộc kiểu char_freq_WORD là phần trăm kí tự trong thư là CHAR, tức là 100*(số lần kí tự CHAR xuất hiện trong thư) / tổng số kí tự trong thư. Sáu kí tự được xét là ;, (, [, !, $ và #.

Ví dụ: Lọc thư rác

- 1 số thực trong đoạn $[1, \ldots]$ thuộc kiểu *capital_run_length_average* là độ dài trung bình của các chuỗi chứa toàn các kí tự hoa trong thư.
- 1 số nguyên trong đoạn $[1, \ldots]$ thuộc kiểu *capital_run_length_longest* là độ dài của chuỗi dài nhất chứa toàn các kí tự hoa.
- 1 số nguyên trong đoạn $[1, \ldots]$ thuộc kiểu *capital_run_length_total* là tổng độ dài của các chuỗi chứa toàn các kí tự hoa, tức là tổng số kí tự hoa trong thư.

Nếu thư là thư rác thì nó được đánh dấu thuộc lớp 1, không phải thư rác thì thuộc lớp 0

Bài tập

Lập trình cho mô hình Hồi quy Logistic, sau đó tính tham số mô hình cho các trường hợp sau:

- Sử dụng toàn bộ 57 đặc trưng, có tham số tự do và tốc độ học $\alpha = 10^{-6}$.
- Sử dụng toàn bộ 57 đặc trưng, không có tham số tự do và tốc độ học $\alpha = 10^{-6}$.
- Sử dụng 55 đặc trưng (bỏ 2 đặc trưng cuối cùng) và có sử dụng tham số tự do θ_0, tốc độ học $\alpha = 10^{-6}$.
- Sử dụng 55 đặc trưng (bỏ 2 đặc trưng cuối cùng) và có sử dụng tham số tự do θ_0, tốc độ học $\alpha = 10^{-3}$.