LOGISTIC REGRESSION
Classification and Prediction

Lê Hồng Phương

<*phuonglh@gmail.com*>
Vietnam National University of Hanoi
Hanoi University of Science

March 2015

Nội dung

Mô hình hồi quy logistic

- Mô hình hồi quy logistic được sử dụng rộng rãi trong nhiều bài toán thống kê và học máy.
- Là mô hình phân biệt: mô hình trực tiếp $P(y|\,\mathbf{x})$.
- Hồi quy logistic được sử dụng trong phân loại nhị phân, $y \in \{0, 1\}$.

Mô hình hồi quy logistic

- Xét bài toán phân loại nhị phân, mỗi đối tượng $\mathbf{x}$ cần được phân vào một trong hai lớp $y \in \{0, 1\}$.
- Ta chọn hàm dự báo $h_\theta(\mathbf{x})$ như sau:

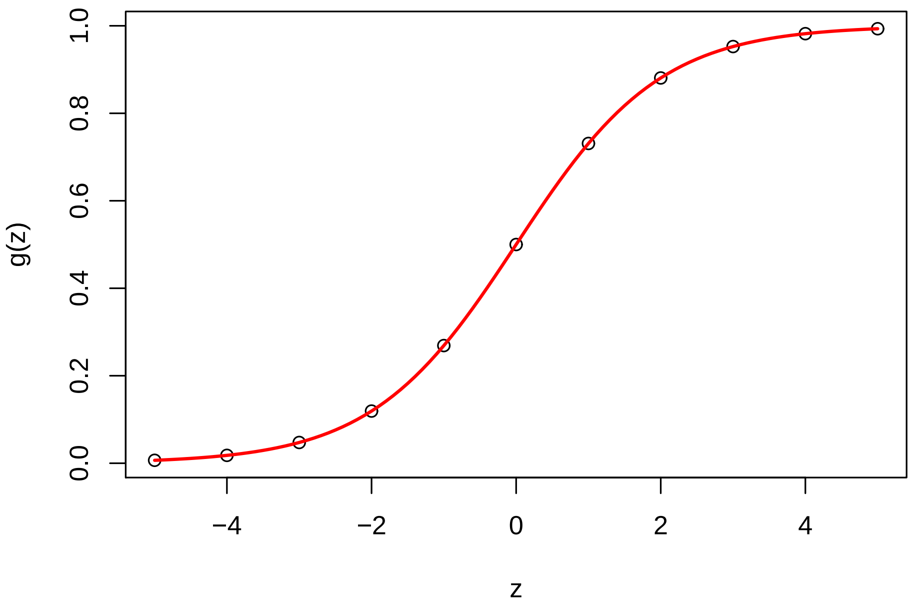
$$h_\theta(\mathbf{x}) = g(\theta^T\,\mathbf{x}) = \frac{1}{1 + \exp(-\theta^T\,\mathbf{x})}, \tag{1}$$

trong đó

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

được gọi là *hàm logistic* hoặc *hàm sigmoid*.

Hàm sigmoid $g(z) = \frac{1}{1+\exp(-z)}$



Hàm sigmoid $g(z) = \frac{1}{1+\exp(-z)}$

Nhận xét:

- $g(z) \to 1$ khi $z \to \infty$
- $g(z) \to 0$ khi $z \to -\infty$.
- $g(z)$ và $h_\theta(\mathbf{x})$ luôn nằm trong đoạn $[0, 1]$.

Đạo hàm của hàm logistic:

$$g'(z) = g(z)(1 - g(z)).$$

Mô hình hồi quy logistic

Mô hình hồi quy logistic:

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

trong đó $\theta \in \mathbb{R}^{D+1}$ là véc-tơ tham số của mô hình.

Mô hình hồi quy logistic

Giả sử đã biết véc-tơ tham số θ, ta sử dụng mô hình để phân loại như sau:

- Xếp đối tượng $\mathbf{x}$ vào lớp 1 nếu

$$P(y = 1|\,\mathbf{x}; \widehat{\theta}) > P(y = 0|\,\mathbf{x}; \widehat{\theta}) \Leftrightarrow h_{\widehat{\theta}}(\mathbf{x}) > 1/2 \Leftrightarrow \boxed{\widehat{\theta}^T \mathbf{x} > 0}.$$

- Ngược lại thì $\mathbf{x}$ được xếp vào lớp 0.

Quy tắc phân loại dựa vào một tổ hợp tuyến tính của x_j và θ_j nên mô hình hồi quy logistic thuộc dạng mô hình phân loại tuyến tính.

Huấn luyện mô hình

Ta có thể viết gọn xác suất của lớp y dưới dạng

$$P(y|\,\mathbf{x};\theta) = (h_\theta(\mathbf{x}))^y(1-h_\theta(\mathbf{x}))^{1-y}.$$

Giả sử rằng tập dữ liệu huấn luyện được sinh độc lập nhau, khi đó hợp lí của dữ liệu với tham số θ là

$$\begin{aligned} L(\theta) &= \prod_{i=1}^N P(y_i|\,\mathbf{x}_i;\theta) \\ &= \prod_{i=1}^N (h_\theta(\mathbf{x}_i))^{y_i}(1-h_\theta(\mathbf{x}_i))^{1-y_i}. \end{aligned}$$

Huấn luyện mô hình

- Log của hợp lí

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

- Sử dụng phương pháp hợp lí cực đại để ước lượng θ, ta cần giải bài toán tối ưu:

$$\widehat{\theta} = \arg\min_\theta[-\ell(\theta) + \lambda R(\theta)], \tag{4}$$

  trong đó $R(\theta)$ là hàm hiệu chỉnh.
- Tham số $\lambda \geq 0$ dùng để điều khiển tính cân bằng của mô hình trong việc phù hợp với dữ liệu quan sát và việc hiệu chỉnh tham số.

Các dạng hiệu chỉnh tham số

Có ba dạng hiệu chỉnh thường gặp:

- Nếu $R(\theta) = 0$ thì đây là mô hình hồi quy logistic thường, không có hiệu chỉnh.
- Nếu $R(\theta) = \|\theta\|_1 = \sum_{j=1}^D |\theta_j|$ thì ta có mô hình hồi quy logistic hiệu chỉnh dạng L_1.
- Nếu $R(\theta) = \|\theta\|_2 = \sum_{j=1}^D \theta_j^2$ thì ta có mô hình hồi quy logistic hiệu chỉnh dạng L_2.
- Nếu $R(\theta) = \sum_{j=1}^D \log\left(\frac{e^{\theta_j}+e^{-\theta_j}}{2}\right)$ thì ta có mô hình hồi quy logistic hiệu chỉnh dạng hyperbolic-L_1.

Các thuật toán lặp giải bài toán tối ưu

Ta cần chọn θ làm cực tiểu hoá hàm mục tiêu

$$J(\theta) = -\ell(\theta) + \lambda R(\theta).$$

Hai thuật toán lặp để tìm θ:

- Thuật toán giảm gradient (ngẫu nhiên)
- Thuật toán Newton-Raphson

Thuật toán giảm gradient

Ta xuất phát từ một giá trị khởi đầu nào đó của θ và lặp để cập nhật θ theo công thức

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

trong đó $\nabla J(\theta)$ là gradient của $J(\theta)$:

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

Mỗi tham số θ_j được cập nhật bởi quy tắc:

$$\theta_j := \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j}, \quad \forall j = 0, 1, \dots, D.$$

Thuật toán giảm gradient

Giả sử $N = 1$, tức tập huấn luyện chỉ có một mẫu $(\mathbf{x}, y)$ và không sử dụng hiệu chỉnh. Ta có

$$\frac{\partial \ell(\theta)}{\partial \theta_j} = \left(y \frac{1}{h_\theta(\mathbf{x})} - (1-y) \frac{1}{1 - h_\theta(\mathbf{x})} \right) \frac{\partial h_\theta(\mathbf{x})}{\partial \theta_j}.$$

Vì

$$\begin{aligned} \frac{\partial h_\theta(\mathbf{x})}{\partial \theta_j} &= \frac{\partial g(\theta^T \mathbf{x})}{\partial \theta_j} \\ &= g(\theta^T \mathbf{x})(1 - g(\theta^T \mathbf{x})) \frac{\partial (\theta^T \mathbf{x})}{\partial \theta_j} \\ &= g(\theta^T \mathbf{x})(1 - g(\theta^T \mathbf{x})) x_j, \end{aligned}$$

Thuật toán giảm gradient

nên

$$\begin{aligned} \frac{\partial \ell(\theta)}{\partial \theta_j} &= [y(1 - g(\theta^T \mathbf{x})) - (1-y) g(\theta^T \mathbf{x})] x_j \\ &= [y - g(\theta^T \mathbf{x})] x_j \\ &= [y - h_\theta(\mathbf{x})] x_j. \end{aligned}$$

Từ đó

$$\begin{aligned} \frac{\partial J(\theta)}{\partial \theta_j} &= - \frac{\partial \ell(\theta)}{\partial \theta_j} \\ &= [h_\theta(\mathbf{x}) - y] x_j. \end{aligned}$$

Thuật toán giảm gradient

Do đó, nếu chỉ có một mẫu huấn luyện $(\mathbf{x}_i, y_i)$ thì ta có quy tắc giảm gradient sau:

$$\theta_j := \theta_j - \alpha [h_\theta(\mathbf{x}_i) - y_i] x_{ij}, \quad \forall j = 0, 1, \dots, D. \tag{6}$$

Về mặt trực quan, ta thấy θ_j được cập nhật tỉ lệ với độ lớn của sai số $(h_\theta(\mathbf{x}_i) - y_i)$

- Nếu sai số dự báo càng lớn thì trọng số tương ứng càng cần thay đổi nhiều
- Nếu không có sai số thì không cần cập nhật trọng số.

Thuật toán giảm gradient theo loạt

Nhắc lại: Thuật toán giảm gradient theo loạt tổng quát:

Algorithm 1: Batch Gradient Descent

Data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)$
Result: θ
$\theta \leftarrow \vec{0}$;
repeat
  $\theta \leftarrow \theta - \alpha \sum_{i=1}^{N} \nabla L_i(\theta)$;
until *converged*;

Thuật toán giảm gradient theo loạt

Thuật toán giảm gradient theo loạt cho hồi quy logistic:

Algorithm 2: Batch Gradient Descent for Logistic Regression

Data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)$
Result: θ
$\theta \leftarrow \vec{0}$;
repeat
  $\theta \leftarrow \theta - \alpha \sum_{i=1}^{N} [h_\theta(\mathbf{x}_i) - y_i] x_{ij}$;
until *converged*;

Thuật toán giảm gradient ngẫu nhiên

Nhắc lại: Thuật toán giảm gradient ngẫu nhiên tổng quát:

Algorithm 3: Stochastic Gradient Descent

Data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)$
Result: θ
$\theta \leftarrow \vec{0}$;
repeat
  Shuffle the training set randomly;
  for $i = 1$ *to* N **do**
    $\theta \leftarrow \theta - \alpha \nabla L_i(\theta)$;
until *converged*;

Thuật toán giảm gradient ngẫu nhiên

Thuật toán giảm gradient ngẫu nhiên cho hồi quy logistic:

Algorithm 4: Stochastic Gradient Descent for Logistic Regression

Data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)$
Result: θ
$\theta \leftarrow \vec{0}$;
repeat
  Shuffle the training set randomly;
  for $i = 1$ *to* N **do**
    $\theta \leftarrow \theta - \alpha [h_\theta(\mathbf{x}_i) - y_i] x_{ij}$;
until *converged*;

Hiệu chỉnh dạng L_2

- Nếu ta dùng dạng hiệu chỉnh L_2:
$$J(\theta) = -\ell(\theta) + \frac{1}{2}\lambda \sum_{j=1}^{D} \theta_j^2. \tag{7}$$
- Giảm gradient theo loạt:
$$\theta_0 := \theta_0 - \alpha \sum_{i=1}^{N} [h_\theta(\mathbf{x}_i) - y_i] x_{i0}$$
$$\theta_j := \theta_j - \alpha \sum_{i=1}^{N} [h_\theta(\mathbf{x}_i) - y_i] x_{ij} - \lambda\theta_j, \quad \forall j = 1, \dots, D.$$
- Giảm gradient ngẫu nhiên:
$$\theta_0 := \theta_0 - \alpha [h_\theta(\mathbf{x}_i) - y_i] x_{i0}$$
$$\theta_j := \theta_j - \alpha [h_\theta(\mathbf{x}_i) - y_i] x_{ij} - \lambda\theta_j, \quad \forall j = 1, \dots, D.$$

Trung bình của hợp lí

Lưu ý: Ta có thể thay $\ell(\theta)$ bởi $\frac{1}{N}\ell(\theta)$ để tránh tính toán với các số lớn. Khi đó
$$\sum_{i=1}^{N} [h_\theta(\mathbf{x}_i) - y_i] x_{ij}$$
sẽ được thay bằng
$$\frac{1}{N} \sum_{i=1}^{N} [h_\theta(\mathbf{x}_i) - y_i] x_{ij}.$$

Thuật toán Newton-Raphson

- Một phương pháp khác hay dùng khác để cực tiểu hoá $J(\theta)$ là sử dụng thuật toán Newton.
- Các phương pháp Newton và giả-Newton là các phương pháp thường tối ưu dùng trong giải tích số.
- Cơ sở của phương pháp này như sau:
  - Giả sử ta có hàm thực khả vi $f : \mathbb{R} \to \mathbb{R}$ và ta cần tìm $\theta \in \mathbb{R}$ sao cho $f(\theta) = 0$.
  - Phương pháp Newton cập nhật dần θ theo công thức:
$$\theta := \theta - \frac{f(\theta)}{f'(\theta)}.$$

Thuật toán Newton-Raphson

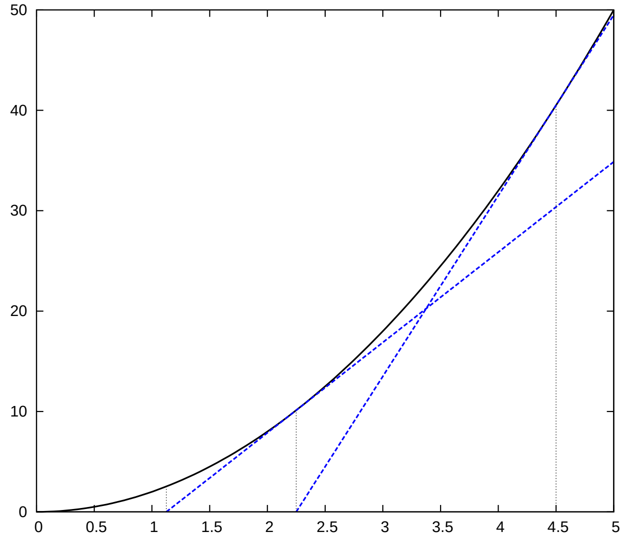
Về mặt trực quan:

- Ta xấp xỉ hàm f bởi một hàm tuyến tính là tiếp tuyến của f tại giá trị θ hiện tại.
- Cập nhật giá trị mới của θ là hoành độ giao điểm giữa tiếp tuyến này với trục hoành.
- Lặp lại quá trình này cho tới khi hội tụ (sai số cập nhật đủ bé).

Chú ý rằng
$$f'(\theta^{(n)}) = \frac{\triangle f}{\triangle \theta} = \frac{f(\theta^{(n)}) - 0}{\theta^{(n)} - \theta^{(n+1)}}.$$
Từ đó ta có
$$\theta^{(n+1)} = \theta^{(n)} - \frac{f(\theta^{(n)})}{f'(\theta^{(n)})}.$$

Thuật toán Newton-Raphson

Phương pháp Newton tìm nghiệm của hàm $f(\theta) = 0$.



Thuật toán Newton-Raphson

- Để cực tiểu hoá $J(\theta)$, ta cần tìm θ sao cho $J'(\theta) = 0$.
- Như vậy, nếu J là hàm khả vi cấp hai thì ta có thể sử dụng phương pháp Newton để tìm θ theo công thức sau

$$\theta := \theta - \frac{J'(\theta)}{J''(\theta)}. \tag{8}$$

Thuật toán Newton-Raphson

Trong không gian nhiều chiều, θ là một véc-tơ, phương pháp Newton có công thức tổng quát như sau

$$\theta := \theta - H^{-1}(\nabla J(\theta)), \tag{9}$$

trong đó H là Hessian của J được xác định bởi

$$H_{ij} = \frac{\partial^2 J(\theta)}{\partial\theta_i \partial\theta_j}, \forall i, j = 0, 1, \ldots, D. \tag{10}$$

Thuật toán Newton-Raphson

Cụ thể là:

- Véc-tơ gradient:

$$\nabla J(\theta) = \begin{pmatrix} \frac{1}{N}\sum_{i=1}^N (h(\mathbf{x}_i) - y_i)x_{i0} \\ \frac{1}{N}\sum_{i=1}^N (h(\mathbf{x}_i) - y_i)x_{i1} \\ \vdots \\ \frac{1}{N}\sum_{i=1}^N (h(\mathbf{x}_i) - y_i)x_{iD} \end{pmatrix}$$

- Ma trận Hessian với kích thước $(D+1) \times (D+1)$:

$$H = \frac{1}{N}\sum_{i=1}^N h(\mathbf{x}_i)(1 - h(\mathbf{x}_i))\,\mathbf{x}_i\,\mathbf{x}_i^T\,.$$

Thuật toán Newton-Raphson

Nếu ta dùng dạng hiệu chỉnh L_2:

- Véc-tơ gradient:

$$\nabla J(\theta) = \begin{pmatrix} \frac{1}{N}\sum_{i=1}^{N}(h(\mathbf{x}_i)-y_i)x_{i0} & \\ \frac{1}{N}\sum_{i=1}^{N}(h(\mathbf{x}_i)-y_i)x_{i1} & -\lambda\theta_1 \\ \vdots & \\ \frac{1}{N}\sum_{i=1}^{N}(h(\mathbf{x}_i)-y_i)x_{iD} & -\lambda\theta_D \end{pmatrix}$$

- Ma trận Hessian với kích thước $(D+1)\times(D+1)$:

$$H = \frac{1}{N}\sum_{i=1}^{N} h(\mathbf{x}_i)(1-h(\mathbf{x}_i))\,\mathbf{x}_i\,\mathbf{x}_i^T + \lambda \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \vdots & 0 & 1 & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \cdots & 1 \end{pmatrix}.$$

Giảm gradient hay Newton-Raphson?

- Trong phương pháp giảm 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.
  - Thông thường, nếu số đặc trưng là nhỏ hơn 100 thì phương pháp Newton hội tụ sau khoảng 15 bước lặp.
- 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 giảm 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.

Giảm gradient hay Newton-Raphson?

- Thuật toán giảm gradient còn có một nhược điểm là khó song song hoá vì bản chất của nó là tính toán tuần tự.
- Chính vì vậy ta khó có thể triển khai thuật toán này trên các máy tính bó hoặc tính toán hiệu năng cao.

GDA và hồi quy logistic

- Trong mô hình GDA, nếu ta coi đại lượng $P(y=1|\,\mathbf{x};\alpha,\mu_0,\mu_1,\Sigma)$ là một hàm của $\mathbf{x}$ thì nó có thể được biểu diễn dưới dạng

$$P(y=1|\,\mathbf{x};\alpha,\mu_0,\mu_1,\Sigma) = \frac{1}{1+\exp(-\theta^T\,\mathbf{x})},$$

  trong đó θ là một hàm thích hợp của $\alpha,\mu_0,\mu_1,\Sigma$.
- Đây chính là hàm logistic để mô hình xác suất $P(y=1|\,\mathbf{x})$.

GDA và hồi quy logistic

- Như vậy, ta thấy nếu $P(\mathbf{x}|y)$ có phân phối chuẩn nhiều chiều thì $P(y|\mathbf{x})$ tuân theo hàm logistic.
- Tuy nhiên điều ngược lại không đúng, tức là nếu $P(y|\mathbf{x})$ có dạng hàm logistic thì không kéo theo được là $P(\mathbf{x}|y)$ có phân phối chuẩn.
- Từ đó ta thấy mô hình GDA đặt giả thiết về dữ liệu *mạnh hơn* mô hình hồi quy logistic.

GDA và hồi quy logistic

- Nếu dữ liệu tuân theo phân phối chuẩn thì phương pháp GDA mô hình tốt và hiệu quả hơn phương pháp hồi quy logistic.
- Ngược lại, vì hồi quy logistic đặt giả thiết yếu hơn về phân phối của dữ liệu nên nó vững hơn và ít bị ảnh hưởng bởi giả thiết mô hình hoá hơn.
- Có nhiều giả thiết khác với phân phối chuẩn cũng dẫn tới phân phối logistic; ví dụ, nếu $\mathbf{x}|y = 0 \sim \texttt{Poisson}(\lambda_0)$ và $\mathbf{x}|y = 1 \sim \texttt{Poisson}(\lambda_1)$ thì $P(y|\mathbf{x})$ cũng có dạng logistic.
- Do đó, khi dữ liệu không có phân phối chuẩn thì hồi quy logistic mô hình tốt hơn GDA. Chính vì vậy, hồi quy logistic được dùng rộng rãi hơn GDA.

Chẩn đoán ung thư

Sử dụng bộ dữ liệu ung thư vú đã giới thiệu trong các bài giảng trước. Ta xây dựng các mô hình hồi quy logistic với các thông số khác nhau như sau:

- Sử dụng thuật toán Newton hoặc giảm gradient để ước lượng tham số;
- Chỉ sử dụng 10 đặc trưng đầu tiên hoặc sử dụng toàn bộ 30 đặc trưng;
- Sử dụng kĩ thuật hiệu chỉnh tham số dạng L_2 hoặc không.

Sử dụng thuật toán Newton-Raphson

Nếu dùng 10 đặc trưng, đánh số các đặc trưng từ 1 tới 10, và không hiệu chỉnh thì các giá trị tham số là

j	θ_j	j	θ_j
1.	2.9479	6.	−2.4326
2.	0.3777	7.	−7.4069
3.	0.0457	8.	−70.1621
4.	−0.0475	9.	−15.1245
5.	−74.4356	10.	96.4245

- Thuật toán cần 8 bước lặp.
- Độ chính xác của mô hình trên tập dữ liệu huấn luyện là 94.72%.
- Nếu sử dụng hiệu chỉnh L_2 với các tham số hiệu chỉnh $\lambda \in \{10^{-6}, 10^{-3}, 10^{-1}\}$ thì mô hình cũng cho độ chính xác tương tự.

Sử dụng thuật toán Newton-Raphson

Nếu dùng 30 đặc trưng thì độ chính xác là 100%, không phụ thuộc vào sử dụng hiệu chỉnh hay không. Các tham số của mô hình là

j	θ_j	j	θ_j	j	θ_j
1.	605.1425	11.	−296.5463	21.	−6.9645
2.	−12.0425	12.	27.1708	22.	−160.7775
3.	−16.4656	13.	118.2308	23.	1.121
4.	−4.6211	14.	−10.9851	24.	−6.9656
5.	−5028.0381	15.	8792.1322	25.	768.7252
6.	5545.4834	16.	−9211.308	26.	965.8915
7.	−2733.9444	17.	−31633.8908	27.	−475.9354
8.	−3316.2043	18.	7888.106	28.	−724.1528
9.	2026.5113	19.	75268.2862	29.	−5942.4999
10.	−6346.2382	20.	9147.0358	30.	−2070.1442

Sử dụng thuật toán giảm gradient

Nếu sử dụng 10 đặc trưng đầu tiên và không sử dụng hiệu chỉnh thì độ chính xác của mô hình trên tập huấn luyện với các giá trị α khác nhau như sau:

α	Độ chính xác	Số bước lặp
10^{-1}	0.62741	2
10^{-2}	0.62741	2
10^{-3}	0.62741	2
10^{-4}	0.37258	3
10^{-5}	0.62741	100
10^{-6}	0.37258	100

Nhận xét:

- Ta thấy độ chính xác của mô hình phụ thuộc vào tốc độ học α.
- Độ chính xác tốt nhất đạt được là 62.74%, nhỏ hơn đáng kể so với phương pháp huấn luyện mô hình bằng thuật toán Newton.

Sử dụng thuật toán giảm gradient

- Véc-tơ tham số θ cũng phụ thuộc vào α. Với $\alpha = 10^{-2}$ thì ta có

$$\theta = (46.534, 73.0323, 295.8065, 1441.1625, 0.3861, \\ 0.275, 0.0762, 0.0444, 0.7283, 0.2702).$$

- Khi sử dụng mô hình có hiệu chỉnh dạng L_2 với các tham số hiệu chỉnh λ lấy các giá trị khác nhau trong tập $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ thì độ chính xác của mô hình không tăng lên, giá trị tốt nhất đạt được cũng là 62.74%.
- Khi sử dụng toàn bộ 30 đặc trưng thì độ chính xác của mô hình cũng không tăng lên. Như vậy thuật toán giảm gradient không tốt bằng thuật toán Newton đối với bài toán này.

So sánh độ chính xác

Độ chính xác của một số mô hình phân loại trên tập dữ liệu ung thư:

Mô hình	Độ chính xác	
	10 đặc trưng	30 đặc trưng
Chuẩn một chiều	92.79%	94.02%
Chuẩn nhiều chiều (GDA)	93.84%	96.48%
Hồi quy tuyến tính	93.84%	96.48%
Hồi quy logistic	94.72%	100%

Lọc thư rác

- Tập dữ liệu Spambase[1] cung cấp 4601 mẫu thư điện tử tiếng Anh dạng thư rác và không phải thư rác.
- 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

[1] `http://archive.ics.uci.edu/ml/datasets/Spambase`

Lọc thư rác

Danh sách các đặc trưng của tập dữ liệu này như sau:

- 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 #.

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.

Lọc thư rác

Log của các xác suất tiên nghiệm $\widehat{\theta}_k, k = 1, 2$ của các lớp:

Lớp	$\log(\widehat{\theta})$
`spam`	-0.931
`non-spam`	-0.500

- Ta thấy hai đặc trưng cuối cùng là hai số nguyên lớn, lớn hơn rất nhiều giá trị của các đặc trưng thực của mô hình.
- Do đó khi xây dựng mô hình, ta thử nghiệm với 2 tập đặc trưng: dùng toàn bộ 57 đặc trưng hoặc dùng 55 đặc trưng đầu tiên.

Sử dụng thuật toán Newton-Raphson

Thuật toán hội tụ sau 12 bước lặp với sai số của log-hợp lí $\epsilon = 10^{-6}$.

- Nếu **có** sử dụng tham số tự do (intercept) θ_0 thì độ chính xác của mô hình là 86.11% với 57 đặc trưng và là 86.87% với 55 đặc trưng.
- Nếu **không** sử dụng tham số tự do θ_0 thì độ chính xác của mô hình là 92.26% với 57 đặc trưng và là 91.26% với 55 đặc trưng.

Sử dụng thuật toán giảm gradient ngẫu nhiên

Trước tiên xét mô hình hồi quy logistic với 55 đặc trưng và **không** sử dụng tham số tự do θ_0. Nếu cố định tốc độ học $\alpha = 10^{-6}$ thì ta có kết quả như sau:

Số bước lặp	**Độ chính xác**	$\lvert\triangle L\rvert$
100	74.96%	2.5442
500	83.69%	0.7486
1000	85.59%	0.3661
2000	87.08%	0.1615
3000	87.98%	0.0965
5000	88.82%	0.0486
10000	89.52%	0.0181
15000	89.76%	0.0100
20000	**90.11**%	0.0006

Sử dụng thuật toán giảm gradient ngẫu nhiên

- Sau $10,000$ bước lặp thì sai khác của log hợp lí của dữ liệu đạt tới giá trị 10^{-2} và độ chính xác của mô hình trên tập huấn luyện đạt 89.52%.
- Sau $20,000$ bước lặp thì sai khác của log hợp lí của dữ liệu đạt tới giá trị 10^{-4} và độ chính xác của mô hình trên tập huấn luyện đạt 90.11%.

Sử dụng thuật toán giảm gradient ngẫu nhiên

Nếu cố định tốc độ học $\alpha = 10^{-3}$ thì ta có kết quả như sau:

Số bước lặp	**Độ chính xác**	$\lvert\triangle L\rvert$
100	85.43%	560
500	89.58%	862
1000	83.41%	14836
2000	89.87%	681
3000	89.82%	82
5000	88.39%	1200
10000	88.15%	49
15000	**90.13**%	55
20000	89.82%	152

Ta thấy khi tốc độ học α không đủ bé thì kết quả của mô hình không ổn định.

- Độ chính xác của mô hình cũng như sai số của log-hợp lí không hội tụ theo số bước lặp.

Sử dụng thuật toán giảm gradient ngẫu nhiên

Nếu xét mô hình hồi quy logistic với 55 đặc trưng và **có** sử dụng tham số tự do θ_0, tốc độ học $\alpha = 10^{-6}$ thì ta có kết quả như sau:

Số bước lặp	**Độ chính xác**	$\lvert \triangle L \rvert$
100	73.24%	2.7243
500	80.41%	0.7991
1000	81.41%	0.3857
2000	82.39%	0.1661
3000	82.65%	0.0971
5000	82.87%	0.0473
10000	**83.67%**	0.0167

Sử dụng thuật toán giảm gradient ngẫu nhiên

Nếu sử dụng toàn bộ 57 đặc trưng, **không có** tham số tự do và tốc độ học $\alpha = 10^{-6}$ thì ta có kết quả như sau:

Số bước lặp	**Độ chính xác**	$\lvert \triangle L \rvert$
100	44.18%	160,015.07
500	68.24%	4041.32
1000	70.44%	152,261.55
2000	48.35%	162,640.91
3000	**76.96%**	151,330.38

Sử dụng thuật toán giảm gradient ngẫu nhiên

Nếu sử dụng toàn bộ 57 đặc trưng, **có** tham số tự do và tốc độ học $\alpha = 10^{-6}$ thì ta có kết quả như sau:

Số bước lặp	**Độ chính xác**	$\lvert \triangle L \rvert$
100	41.44%	160,015.07
500	43.70%	158,769.11
1000	45.31%	165,358.61
2000	**74.46%**	150,231.15
3000	64.76%	1,753.31

So sánh độ chính xác

Độ chính xác của một số mô hình phân loại trên tập dữ liệu thư rác:

Mô hình	**57 đặc trưng**	**55 đặc trưng**
Chuẩn một chiều (dùng chung σ_j)	81.96%	81.69%
Chuẩn một chiều (dùng riêng σ_j)	88.78%	88.30%
GDA	**88.87%**	88.58%
Logistic Newton có hệ số chắn	86.11%	86.87%
Logistic Newton không hệ số chắn	**92.26%**	**91.26%**
Logistic SGD có hệ số chắn	74.46%	83.67%
Logistic SGD không hệ số chắn	70.44%	90.11%

So sánh độ chính xác

- Mô hình sinh có độ chính xác cao nhất là 88.87% (GDA).
- Với các mô hình sinh, việc sử dụng 57 đặc trưng cho kết quả cao hơn nếu chỉ sử dụng 55 đặc trưng.
- Mô hình phân biệt hồi quy logistic có độ chính xác cao nhất là 92.26% khi ước lượng tham số bằng thuật toán Newton.
  - Thuật toán Newton cũng hội tụ rất nhanh chỉ sau 12 bước lặp.
  - Thuật toán SGD đòi hỏi số bước lặp lớn (hàng ngàn) để đạt được độ chính xác tương tự.

So sánh độ chính xác

- Tuy nhiên kết quả của thuật toán SGD không ổn định, phụ thuộc vào tốc độ học α và số bước lặp.
  - Để thuật toán ổn định, tốc độ học α cần được chọn tự động thay vì chọn theo kinh nghiệm.
- Với tập dữ liệu này, mô hình hồi quy logistic không sử dụng tham số tự do θ_0 cho kết quả cao hơn là mô hình có sử dụng tham số tự do.

Bài tập

1. Cài đặt các thuật toán ước lượng tham số của mô hình hồi quy logistic:
   - Thuật toán giảm gradient theo loạt
   - Thuật toán giảm gradient ngẫu nhiên
   - Thuật toán Newton-Raphson
2. Chạy các thuật toán trên dữ liệu thử nghiệm Spambase và kiểm tra kết quả.
3. Tự sưu tầm và thử nghiệm các thuật toán trên một số bộ dữ liệu khác ở UCI Machine Learning Repository[2] và thông báo kết quả.

[2] http://archive.ics.uci.edu/ml/