BÀI 3: HỒI QUY LOGISTIC

Bài toán hồi quy tuyến tính

$$h_w(x): R^n \rightarrow R$$
$$h_w(x) = w_0.1 + w_1.x_1 + \cdots + w_n.x_n$$

Hay $h_w(x) = w^T.x$, với $w = [w_0; w_1; \dots; w_n]$ và $x = [1; x_1; \dots; x_n]$

Bài toán hồi quy Logistic

$$h_w(x): R^n \rightarrow \{0, 1\}$$
$$h_w(x) = g(w^T.x)$$

với $g(z) = \frac{1}{1+ e^{-z}}$ và $z = w^T.x$. Khi đó: $h_w(x) = \frac{1}{1+ e^{-w^T.x}}$

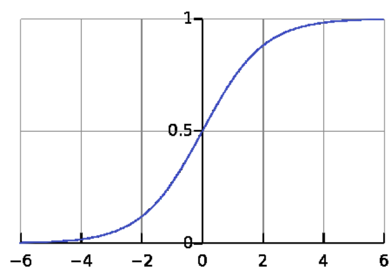
Ghi chú: Trong trường hợp này, bài toán hồi quy Logistic thực chất là bài toán phân lớp nhị phân (binary classification).

Tổng quát hơn, mô hình hồi quy Logistic xây dựng hàm

$$h_w(x): R^n \rightarrow \{0, 1, 2, \dots, k\}, k \in N$$

Ghi chú:

- *Hàm* $g(z) = \frac{1}{1+ e^{-z}}$ *còn gọi là hàm* ***sigmoid****;*
- $0 \le h_w(x) = \frac{1}{1+ e^{-w^T.x}} \le 1.$



Hình 1: Đồ thị hàm Sigmoid trong không gian 2 chiều

Áp dụng hàm sigmoid $h_w(x_i) = \frac{1}{1+ e^{-w^T.x_i}}$ trong dự đoán nhãn lớp y_i như sau

$$\begin{cases} Nếu\ h_w(x_i) \ge 0.5\ thì\ dự\ đoán\ y_i = 1 \\ Nếu\ h_w(x_i) < 0.5\ thì\ dự\ đoán\ y_i = 0 \end{cases}$$

Hàm mất mất – Loss function

Do giá trị của nhãn lớp *y* là rời rạc (hoặc 0 hoặc 1) nên ***độ mất mát thông tin*** giữa dự đoán $h_w(x_i)$ và y_i được tính toán như sau:

$$Cost(h_w(x_i), y_i) = \begin{cases} -log\big(h_w(x_i)\big), & y_i = 1 \\ -log\big(1 - h_w(x_i)\big), & y_i = 0 \end{cases}$$

Khi đó hàm mất mát được tính theo công thức:

$$J(w) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_w(x_i), y_i)$$

Hay

$$J(w) = -\frac{1}{m}\sum_{i=1}^{m}\Big(y_i log\big(h_w(x_i)\big) + (1 - y_i) log\big(1 - h_w(x_i)\big)\Big)$$

Lúc này, bài toán huấn luyện mô hình hồi quy Logistic thực chất là bài toán tối ưu tìm w^* thỏa:

$$w^* = \underset{w}{\operatorname{argmin}}\, J(w)$$

Thuật toán Gradient Descent

Lưu ý: đạo hàm của hàm logarithm $f(x) = \log_a\big(u(x)\big)$ được tính theo công thức sau

$$f'(x) = \frac{u'(x)}{u(x).\ln(a)}$$

Áp dụng công thức tính đạo hàm với hàm logarithm vào hàm mất mát $J(w)$, để tính các đạo hàm riêng, có kết quả như sau

$$\frac{\partial J(w)}{\partial w_i} = \frac{1}{m}\sum_{i=1}^{m}(h_w(x_i) - y_i).x_i$$

Thuật toán Gradient Descent cập nhật giá trị bộ tham số *w* như sau

Repeat until convergence{

$$w_i = w_i - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_w(x_i) - y_i).x_i$$

}

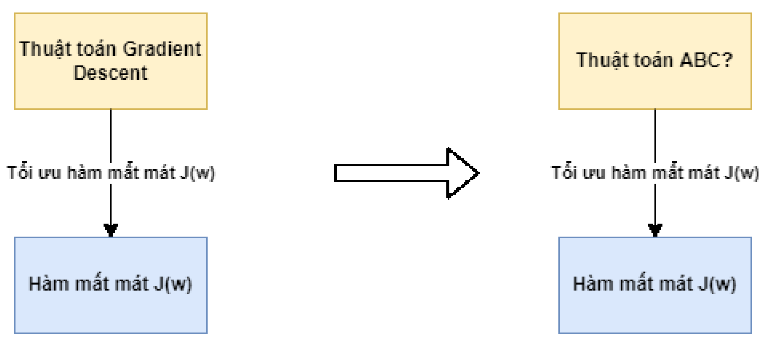
PHẦN PHỤ LỤC

(*Bổ sung cách sử dụng các thuật toán tối ưu*)

Trong thực tế, chúng ta có nhiều thuật toán tối ưu khác ngoài thuật toán Gradient Descent được trình bày trong các Bài giảng 1 – 3. Một số thuật toán tối ưu có thể liệt kê như sau:

- BFGS
- L-BFGS
- TNC
- COBYLA
- .v.v. Bạn có thể tham khảo tại link.

Những thuật toán tối ưu này thường có tốc độ nhanh hơn thuật toán Gradient Descent tuy nhiên lại phức tạp hơn GD nhiều. Do vậy, để tiết kiệm thời gian khi xây dựng thuật toán tối ưu, chúng ta lựa chọn cách sử dụng lại các thuật toán đã được mã hóa. Trong Python, thư viện ***scipy.optimize*** cung cấp các thuật toán tối ưu như vậy.



Hình 2: Hình dung sử dụng thuật toán của SciPy để tối ưu hàm mất mát J(w)

Hướng dẫn sử dụng hàm minimize của thư viện SciPy.optimize

Hướng dẫn lập trình: xem tập tin gợi ý