Logistic Regression

Tuan Nguyen

Ngày 13 tháng 10 năm 2022

Classification problem
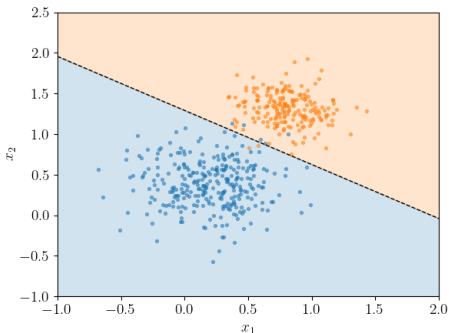
Probabilistic view of classification

Logistic regression

Gradient descent

Classification problem

The goal in classification is to take an input vector x and to assign it to one of K discrete classes C_k where k = 1,...,K.

The input space is thereby divided into decision regions whose boundaries are called decision boundaries or decision surfaces.

Hình 1: Decision boundary

Probabilistic view of classification

Consider first of all the case of two classes. The posterior probability for class C_1 can be written as:

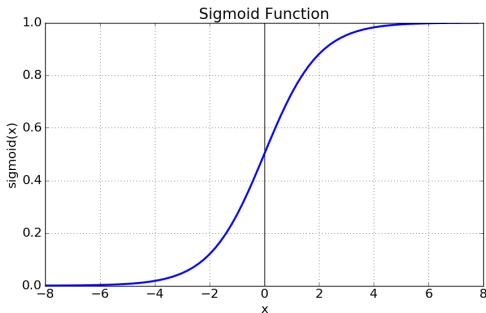$$p(C_1|x) = \frac{p(x|C_1)p(C_1)}{p(x|C_1)p(C_1) + p(x|C_2)p(C_2)} = \frac{1}{1 + e^{-a}} = \sigma(a)$$

where we have defined

$$a = \log\frac{p(x|C_1)p(C_1)}{p(x|C_2)p(C_2)}$$

and $\sigma(a)$ is the logistic sigmoid function defined by

$$\sigma(a) = \frac{1}{1 + e^{-a}}$$

Exercise: Calculate the derivative of sigmoid function.

Hình 2: Sigmoid function

The model logistic regression is defined as:

$$p(C_1|\phi) = y(\phi) = \sigma(w^T\phi)$$
$$p(C_2|\phi) = 1 - p(C_1|\phi)$$

For a data set ϕ_n, t_n, where $t_n \in \{0, 1\}$ and $\phi_n = \phi(x_n)$, with n = 1,...,N, the likelihood function can be written

$$p(t|w) = \prod_{n=1}^{N} y_n^{t_n}(1 - y_n)^{1-t_n}$$

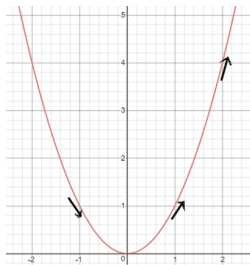where $t = (t_1, ..., t_N)^T$ and $y_n = p(C_1|\phi_n)$

We can define an error function by taking the negative logarithm of the likelihood, which gives the cross_entropy error function in the form

$$L = -\log p(t|w) = -\sum_{n=1}^{N}\{t_n \log y_n + (1 - t_n)\log(1 - y_n)\}$$

where $y_n = \sigma(a_n)$ and $a_n = w^T\phi_n$
Taking the gradient of the error function with respect to w, we obtain
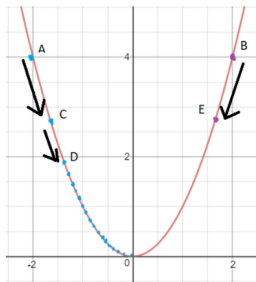
$$\nabla L = \sum_{n=1}^{N}(y_n - t_n)\phi_n$$

Gradient descent

Hình 3: Function $f(x) = x^2$

Function $f(x) = x^2 \rightarrow f'(x) = 2x$. Remarks:

- $f'(1) = 2 * 1 < f'(2) = 2 * 2 =>$ the function at $x = 2$ is steeper than the function at $x = 1 =>$ the higher absolute value of gradient, the steeper function is.

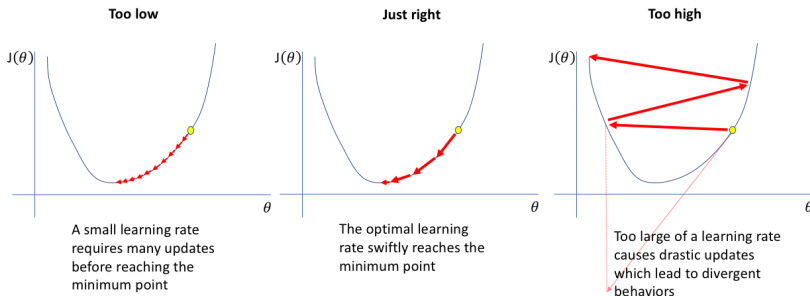- $f'(-1) = 2 * (-1) = -2 < 0 =>$ the function decreases (x increases, y decreases)

Gradient descent (cont.)

Steps to optimize the function $f(x)$, $R \rightarrow R$, $x \rightarrow f(x)$:

1. Initialize randomly $x = x_0$

2. Update $x = x$ - learning_rate $\times$ $f'(x)$, learning_rate is a positive small number.

3. If $f(x)$ is small enough, stop the algorithm. Otherwise, repeat the second step.

Hình 4: Gradient descent update

How to choose learning rate?

Too low

$J(\theta)$

θ

A small learning rate requires many updates before reaching the minimum point

Just right

$J(\theta)$

θ

The optimal learning rate swiftly reaches the minimum point

Too high

$J(\theta)$

θ

Too large of a learning rate causes drastic updates which lead to divergent behaviors

Hình 5: How to choose learning rate
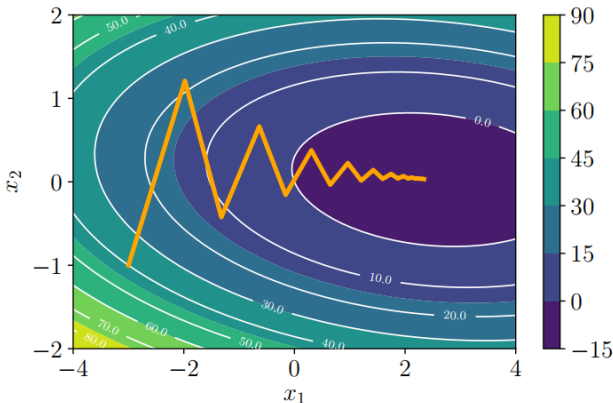
Steps to optimize the function f(x), $R^n \to R$, $x \to f(x)$:

1. Initialize randomly x

2. Update $x = x -$ learning_rate $\times (\dfrac{df}{dx})^T$, learning_rate is a positive small number.

3. If f(x) is small enough, stop the algorithm. Otherwise, repeat the second step.

For example, f(x): $R^2 \to R$

$$f(x) = \frac{1}{2} \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 1 & 20 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} 5 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Initial $x_0 = \begin{bmatrix} -3 \\ -1 \end{bmatrix}$, iterate 5 steps of gradient descent algorithm.

Hình 6: Gradient descent algorithm