Author: Zhengxiang (Jack) Wang
Date: 2021-08-12
GitHub: https://github.com/jaaack-wang (https://github.com/jaaack-wang)
Contact: jackwang196531@gmail.com (mailto:jackwang196531@gmail.com)
About: Hands-on tutorial for writing out gradient descent for logistic regression model

Table of Contents

1. Logistic regression formula

For a single training example, a logistic regression can be given as:

$$\hat{y} = \frac{1}{e^{-z}} \tag{1-1}$$

where z usually takes the form of linear regression $z = \sum_{k=1}^{n} w_k x_k + b$ or $z = \mathbf{x}\mathbf{w} + b$. We usually use the later vectorized form to plug into the formula (1-1) so that it can look neater:

$$\hat{y} = \frac{1}{e^{-(\mathbf{x}\mathbf{w}+b)}} \tag{1-2}$$

where:

- $\mathbf{x}$: a row vector of n columns, representing $[x_1, x_2, \ldots, x_n]$. If you make $\mathbf{x}$ a column vector instead, formula (1) should be $\hat{y} = \mathbf{x}^\mathbf{T}\mathbf{w} + b$ or $\hat{y} = \mathbf{w}^T\mathbf{x} + b$ (provided that $\mathbf{w}$ is a column vector).
- $\mathbf{w}$: a column vector of n rows, representing $[w_1, w_2, \ldots, w_n]^T$.
- $\mathbf{x}\mathbf{w}$: the dot product of $\mathbf{x}$ and $\mathbf{w}$, equal to $\sum_{k=1}^{n} w_k x_k$.

Therefore, a logistic regression can be thought of as a composite function $\sigma(z)$ where σ is a sigmoid function and z is a linear regression.

More generally, for m training examples, we can write formula (1-1) as follows:

$$\hat{y} = \sum_{i=1} m \frac{1}{e^{-z_i}} \tag{1-3}$$

where $z_i = \sum_{k=1}^{n} w_{ik} x_{ik} + b$. Mathematically, we do not input a sigmoid function with an array of values or a vector, but in some interpreted programming languages, such as Python, R, MATLAB, or Octave, that is made possible with element-wise calculation (in Python and R, element-wise calculation is automatic whereas in MATLAB and Octave, it has to be activated by a dot "." followed by "/" or "*" when doing division and multiplication). That is to say, we can implement $\frac{1}{e^{-(\mathbf{X}\mathbf{w}+\mathbf{b})}}$ (where $\mathbf{X}$ is a m by n matrix) as an equivalent to the formula (1-3) in these interpreted programming languages without explict for loops.

2. Logistic regression loss function

Logistic regression loss function is commonly given as:

$$L(\mathbf{y}, \hat{\mathbf{y}}) = -\frac{1}{m}\sum_{i=1}^{m}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \tag{2-1}$$

where:

- $\mathbf{y}$: m true output values, either 0 or 1, a column vector of m rows.
- $\hat{\mathbf{y}}$: m predicted output values, all $\in (0, 1)$; a column vector of m rows.
- y_i is the i th true output value and $\hat{y}_i$ is the i th predicted output value.

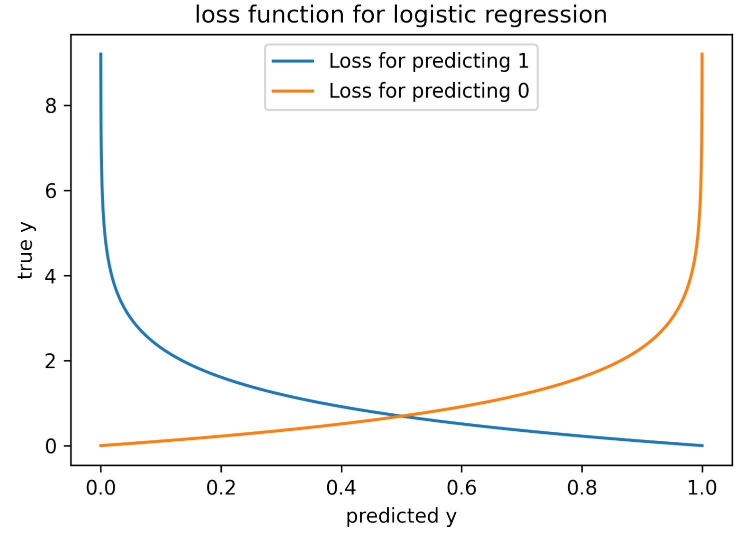
To simplify, for any **single** training example, formula (2-1) can be rewritten as (do not forget the negative sign):

$$L(y_i, \hat{y}_i) = -y_i \log(\hat{y}_i) - (1 - y_i)\log(1 - \hat{y}_i) \tag{2-2}$$

Or:

$$L(y_i, \hat{y}_i) = \begin{cases} -log(\hat{y}_i), & if \quad y = 1 \\ -\log(1 - \hat{y}_i), & if \quad y = 0 \end{cases} \tag{2-3}$$

As for any given $\hat{y}_i$, $\hat{y}_i \in (0, 1)$. Taking a **negative log probability** of $-\log(\hat{y}_i)$ will thus give us a minimal loss when it is approaching 1 and an indefinitely large loss when it is approaching 0 if the true output value is 1. Similarly, if the true output value is 0, $-\log(1 - \hat{y}_i)$ will be minimal when it is approaching 0 and indefinitely large when it is approaching 1.



I will more generally talk about the gradients of the loss function for both the sigmoid function and the softmax function in a separate file within the Backpropagation folder from the perspective of cross-entropy, which makes the derivation easier to compute.

3. Deriving gradients for the loss function

To derive the gradients for the logistic regression loss function, we will first look at a single training example and then generalize the result to m training examples. More concretely, we will first derive the graidents for the formula (2-2) with regard to the weights $\mathbf{w}$ and the bias term b, and the generalize the result for the case of the formula (2-1). We will also apply the **chain rule** to make the derivation process easier.

If you read the file `linear_regression_gradient_descent.ipynb` carefully enough, **we will find that the gradients both with regard to w and b are exactly the same with those derived from the linear regression model!**.

3.1 With regard to w

According to the chain rule, for any **single** training example, $\frac{\partial L(y_i,\hat{y}_i)}{\partial \mathbf{w}} = \frac{\partial L(y_i,\hat{y}_i)}{\partial \hat{y}_i}\frac{\partial \hat{y}_i}{\partial z_i}\frac{\partial \hat{z}_i}{\partial \mathbf{w}}$.

As $z_i = \sum_{k=1}^{n} w_k x_{ik} + b$ and $\mathbf{w} = [w_1 w_2 \cdots w_n]$, it is easy to see that $\frac{\partial z}{\partial \mathbf{w}} = [x_{i1}, x_{i2}, \cdots, x_{in}]$. This is because, to derive $\frac{\partial z_i}{\partial \mathbf{w}}$, we will need to derive $\frac{\partial}{\partial w_k}\sum_{k=1}^{n} w_k x_{ik} + b$ one at a time for $k \in [1, n]$, which will result in a vector of n elements in the end.

Therefore, the key to $\frac{\partial L(y_i,\hat{y}_i)}{\partial \mathbf{w}}$ lies in the first two parts of the derivation chain, i.e., $\frac{\partial L(y_i,\hat{y}_i)}{\partial \hat{y}_i}$ (Part One) and $\frac{\partial \hat{y}_i}{\partial z_i}$ (Part Two).

For Part One:

$$\frac{\partial L(y_i,\hat{y}_i)}{\partial \hat{y}_i} = \frac{\partial}{\partial \hat{y}_i}[-y_i \log(\hat{y}_i) - (1-y_i)\log(1-\hat{y}_i)] = -y_i\frac{1}{\hat{y}_i} + (1-y_i)\frac{1}{1-\hat{y}_i} \tag{3-1}$$

This is because, $\frac{\partial}{\partial \hat{y}_i}[-(1-y_i)\log(1-\hat{y}_i)] = -(1-y_i)\frac{\partial \log(1-\hat{y}_i)}{\partial(1-\hat{y}_i)}\frac{\partial(1-\hat{y}_i)}{\partial \hat{y}_i} = -(1-y_i)(-\frac{1}{1-\hat{y}_i}) = (1-y_i)\frac{1}{1-\hat{y}_i}$.

By plugging $y_i = \frac{1}{1+e^{-z_i}}$ as in the formula (1-1), (3-1) is equal to:

$$\frac{\partial L(y_i,\hat{y}_i)}{\partial \hat{y}_i} = -y_i(1+e^{-z_i}) + (1-y_i)(\frac{1+e^{-z_i}}{e^{-z_i}}) \tag{3-2}$$

For Part Two:

$$\frac{\partial \hat{y}_i}{\partial z_i} = \frac{\partial}{\partial z_i}\frac{1}{1+e^{-z_i}} = \frac{\partial}{\partial z_i}(1+e^{-z_i})^{-1} = (1+e^{-z_i})^{-2}e^{-z_i} \tag{3-3}$$

More details: $\frac{\partial}{\partial z_i}(1+e^{-z_i})^{-1} = \frac{\partial}{\partial(1+e^{-z_i})}(1+e^{-z_i})^{-1}\frac{\partial}{\partial z_i}(1+e^{-z_i}) = -(1+e^{-z_i})^{-2}(-e^{-z_i}) = (1+e^{-z_i})^{-2}e^{-z_i}$. Chain rules again!

Combine the first two parts together:

$$\frac{\partial L(y_i,\hat{y}_i)}{\partial \hat{y}_i}\frac{\partial \hat{y}_i}{\partial z_i} = [-y_i(1+e^{-z_i}) + (1-y_i)(\frac{1+e^{-z_i}}{e^{-z_i}})]\ \ (1+e^{-z_i})^{-2}e^{-z_i} = -y_i\frac{e^{-z_i}}{1+e^{-z_i}} + (1-y_i)\frac{1}{1+e^{-z_i}} \tag{3-4}$$

A final trick! It turns out that $\frac{e^{-z_i}}{1+e^{-z_i}} = 1 - \frac{1}{1+e^{-z_i}} = 1 - \hat{y}_i$, so the formula (3-4) can be rewritten as follows:

$$\frac{\partial L(y_i,\hat{y}_i)}{\partial \hat{y}_i}\frac{\partial \hat{y}_i}{\partial z_i} = -y_i(1-\hat{y}_i) + (1-y_i)\hat{y}_i = \hat{y}_i - y_i \tag{3-5}$$

Therefore, **everything taken together**, for any **single** training example, the gradient for the loss function with regard to $\mathbf{w}$ is the following:

$$\frac{\partial L(y_i, \hat{y}_i)}{\partial \mathbf{w}} = \frac{\partial L(y_i, \hat{y}_i)}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial z_i} \frac{\partial \hat{z}_i}{\partial \mathbf{w}} = \begin{bmatrix} (\hat{y}_i - y_i)x_{i1} & (\hat{y}_i - y_i)x_{i2} & \cdots & (\hat{y}_i - y_i)x_{in} \end{bmatrix} \tag{3-6}$$

Or more comprehensively:

$$\frac{\partial L(y_i, \hat{y}_i)}{\partial w_k} = (\hat{y}_i - y_i)x_{ik} \tag{3-7}$$

where $k \in [1, n]$.

For m training examples, the gradients with regard to the weights thus becomes:

$$\frac{\partial L(\mathbf{y_i}, \hat{\mathbf{y}}_\mathbf{i})}{\partial w_k} = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i)x_{ik} \tag{3-8}$$

where $k \in [1, n]$.

3.2 With regard to b

According to the chain rule, for any **single** training example, $\frac{\partial L(y_i, \hat{y}_i)}{\partial b} = \frac{\partial L(y_i, \hat{y}_i)}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial z_i} \frac{\partial \hat{z}_i}{\partial b}$. Apparently, $\frac{\partial \hat{z}_i}{\partial b} = \frac{\partial}{\partial b} \sum_{k=1}^{n} w_k x_{ik} + b = 1$. Thus:

$$\frac{\partial L(y_i, \hat{y}_i)}{\partial b} = \frac{\partial L(y_i, \hat{y}_i)}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial z_i} = \hat{y}_i - y_i \tag{3-9}$$

which we already proved in the formula (3-5) above.

Therefore, for m training examples, the gradients with regard to the bias term becomes:

$$\frac{\partial L(\mathbf{y_i}, \hat{\mathbf{y}}_\mathbf{i})}{\partial w_k} = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i) \tag{3-10}$$

where $k \in [1, n]$.

4. Gradient descent

4.1 With regard to w

According to the formula (3-8), the gradient descent formula for updating w_k is as follows:

$$w_{k_{new}} = w_k - \frac{\alpha}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)x_{ik} \tag{4-1}$$

where $k \in [1, n]$ and α is the rate of change we want the gradient to decrease (not necessarily always happens), commonly known as the "learning rate". **Please note that**, to implement (4-1), we need to assign $w_{k_{new}}$ as the updated w_k until we have all the weights updated. This is because all the weights are updated based on the old weights and overwriting w_k with the updated w_k before all weights are updated will cause w_2 to w_n to get updated not based on the previous weights, which is problematic.

A more convenient way to update all the weights at once is to vectorize (4-1).

$$\mathbf{w} = \mathbf{w} - \frac{\alpha}{m}\mathbf{X}^T(\hat{\mathbf{y}} - \mathbf{y}) \tag{4-2}$$

where:

- $\mathbf{X}$: a m by n matrix and $\mathbf{X}^T$ is its transpose (n by m dimensional).
- $\hat{\mathbf{y}}$: a column vector of m rows, representing m predicted output values.
- $\mathbf{y}$: a column vector of m rows, representing m true output values.

This works because mathematically, $\mathbf{X}^T(\hat{\mathbf{y}} - \mathbf{y}) = \sum_{i=1}^{m}(\hat{y}_i - y_i)x_{ik}$ for $k \in [1, n]$.

4.2 With regard to b

According to the formula (3-10), the gradient descent formula for updating b is as follows:

$$b = b - \frac{\alpha}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i) \tag{4-3}$$