Author: Zhengxiang (Jack) Wang
Date: 2021-08-11
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 linear regression model

Table of Contents

1. Linear regression formula

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

$$\hat{y} = \sum_{k=1}^{n} w_k x_k + b \tag{1-1}$$

where:

- $\hat{y}$: the predicted output value, a scalar.
- w_k : the kth weight for the kth input variable x_k , a scalar.
- x_k : the input value for the kth input variable x_k , a scalar.
- b: the bias term, a scalar.
- k: index, $k \in [1, n]$.
- n: the number of input variables, a positive integer.

Usually, when $n = 1$, the model is known as **simple linear regression**; when $n \geq 2$, the model becomes **multivariate linear regression**.

Formula (1-1) can also be written in a vectorized form as follows:

$$\hat{y} = \mathbf{x}\mathbf{w} + b \tag{1-2}$$

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 $\sum_{k=1}^{n} w_k x_k$.

More generally, using vectorized expression, we can generize formula (1-2) for the case of m training examples as follows:

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{w} + \mathbf{b} \tag{1-3}$$

where:

- $\hat{y}$: m predicted output values, a column vector of m rows.
- $\mathbf{X}$: a m by n input variables' matrix. If you make $\mathbf{X}$ (n, m) dimensional, formula (1-3) should be rewritten as: $\hat{\mathbf{y}} = \mathbf{X}^T\mathbf{w} + \mathbf{b}$.
- $\mathbf{w}$: a column vector of n rows. The weights are shared for all the training examples.
- $\mathbf{b}$: a column vector of m rows, each row having identical values as the bias term is also shared for all the training examples.

2. Linear regression loss function

Linear regression model usually uses averaged mean squared error (MSE) as its loss function, which is given as:

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

where:

- $\mathbf{y}$: m true output values, a column vector of m rows.
- $\hat{\mathbf{y}}$: m predicted output values, a column vector of m rows.
- y_i is the ith true output value and $\hat{y}_i$ is the ith predicted output value.
- i: index, $i \in [1, m]$.

Please note that, we use $\frac{1}{2m}$ as the coefficient to cancel out the 2 we will get when deriving $(\hat{y}_i - y_i)^2$ with regard to $\hat{y}_i$ (or the weights and the bias term). This is just a convention followed by many. Using $\frac{1}{m}$, $\frac{1}{2m}$ or even just 1 will do the same job in terms of gradient descent as these coefficients do not affect how the loss function scales when we change the values of $\hat{y}_i$ (or the weights and the bias term). Instead, these coefficients will only result in the final calculated loss being of different magnitudes.

Based on formula (1-1), we can expand formula (2-1) as:

$$L(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{2m} \sum_{i=1}^{m} (\sum_{k=1}^{n} w_k x_{ik} + b - y_i)^2 \tag{2-2}$$

where:

- w_k: the kth weight corresponding to x_{ik}.
- x_{ik}: the input value for the kth variable x_k in the ith training example.

We can also vectorize the formula (2-1) to make it look simpler:

$$L(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{2m} (\hat{\mathbf{y}} - \mathbf{y})^T (\hat{\mathbf{y}} - \mathbf{y}) \tag{2-3}$$

Please note that, as both $\hat{\mathbf{y}}$ and $\mathbf{y}$ are column vectors of m rows, $\hat{\mathbf{y}} - \mathbf{y}$, which is an element-wise subtraction, will still be a column vector of m rows. By transposing $\hat{\mathbf{y}} - \mathbf{y}$, we get a row vector of m columns. The dot product of $(\hat{\mathbf{y}} - \mathbf{y})^T (\hat{\mathbf{y}} - \mathbf{y})$ is equal to $\sum_{i=1}^{m} (\hat{y}_i - y_i)^2$.

We can further expand formula (2-3) by replacing $\hat{\mathbf{y}}$ with $\mathbf{Xw} + \mathbf{b}$ according to formula (1-3):

$$L(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{2m} (\mathbf{Xw} + \mathbf{b} - \mathbf{y})^T (\mathbf{Xw} + \mathbf{b} - \mathbf{y}) \tag{2-4}$$

3. Deriving gradients for the loss function

3.1 With regard to w

To derive the gradients for the loss function of the linear regression model, we get:

$$\frac{\partial L(\mathbf{y},\hat{\mathbf{y}})}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}}[\frac{1}{2m}\sum_{i=1}^{m}(\sum_{k=1}^{n} w_k x_{ik} + b - y_i)^2] \tag{3-1}$$

where:

- $\mathbf{w}$: a vector of n elementes, i.e., $[w_1, w_2, \ldots, w_n]$.

To simplify, we will first derive a single training example for illustration, say, $(\sum_{k=1}^{n} w_k x_{ik} + b - y_i)^2$ or $(\hat{y}_i - y_i)^2$, with regard to $\mathbf{w}$, by which we will get:

$$\frac{\partial}{\partial \mathbf{w}}(\hat{y}_i - y_i)^2 = \begin{bmatrix} \frac{\partial}{\partial w_1}[(\sum_{k=1}^{n} w_k x_{ik} + b - y_i)^2] & \frac{\partial}{\partial w_2}[(\sum_{k=1}^{n} w_k x_{ik} + b - y_i)^2] & \cdots & \frac{\partial}{\partial w_n}[(\sum_{k=1}^{n} w_k x_{ik} + b - y_i)^2] \end{bmatrix} \tag{3-2}$$

Deriving a single training example with regard to a single weight, say, w_1 , is relatively easy and we can then apply the derived result to other weights because the pattens are same. To derive $\frac{\partial}{\partial w_1}[(\sum_{k=1}^{n} w_k x_{ik} + b - y_i)^2]$ or $\frac{\partial}{\partial w_1}(\hat{y}_i - y_i)^2$, we can apply chain rule to get the answer:

$$\frac{\partial}{\partial w_1}(\hat{y}_i - y_i)^2 = \frac{\partial(\hat{y}_i - y_i)^2}{\partial(\hat{y}_i - y_i)}\frac{\partial}{\partial w_1}(\hat{y}_i - y_i) = 2(\hat{y}_i - y_i)\frac{\partial}{\partial w_1}(\sum_{k=1}^{n} w_k x_{ik} + b - y_i) = 2(\hat{y}_i - y_i)x_{i1} \tag{3-3}$$

Please note that, $\frac{\partial}{\partial w_1}(\sum_{k=1}^{n} w_k x_{ik} + b - y_i) = x_{i1}$ because $\frac{\partial}{\partial w_1}(\sum_{k=2}^{n} w_k x_{ik} + b - y_i) = 0$. For w_1 , $(\sum_{k=2}^{n} w_k x_{ik} + b - y_i)$ can be thought of as a constant. More generally, for w_j where $j \in [1, n]$, $\frac{\partial}{\partial w_j}(\sum_{k=1}^{n} w_k x_{ik} + b - y_i) = x_{ij}$ because $\frac{\partial}{\partial w_j}(\sum_{k\neq j} w_k x_{ik} + b - y_i) = 0$.

Therefore, using (3-3), we can rewrite (3-2) as follows:

$$\frac{\partial}{\partial \mathbf{w}}(\hat{y}_i - y_i)^2 = \begin{bmatrix} 2(\hat{y}_i - y_i)x_{i1} & 2(\hat{y}_i - y_i)x_{i2} & \cdots & 2(\hat{y}_i - y_i)x_{in} \end{bmatrix} \tag{3-4}$$

Now, let's put everything back together and derive the gradients for the entire training set with regard to $\mathbf{w}$:

$$\frac{\partial L(\mathbf{y},\hat{\mathbf{y}})}{\partial \mathbf{w}} = \begin{bmatrix} \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)x_{i1} & \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)x_{i2} & \cdots & \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)x_{in} \end{bmatrix} \tag{3-5}$$

which is equal to:

$$\frac{\partial L(\mathbf{y},\hat{\mathbf{y}})}{\partial w_k} = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)x_{ik} \tag{3-6}$$

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

To make (3-5) look even simpler, we can vectorize it as:

$$\frac{\partial L(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{w}} = \frac{1}{m}\mathbf{X}^T(\hat{\mathbf{y}} - \mathbf{y}) \tag{3-7}$$

where:

- $\mathbf{y}$: m true output values, a column vector of m rows.
- $\hat{\mathbf{y}}$: m predicted output values, a column vector of m rows.
- $\mathbf{w}$: a column vector of n rows.
- $\mathbf{X}$: a m by n input variables' matrix and $\mathbf{X}^T$ is its transpose.

Please note that, the result of $\frac{1}{m}\mathbf{X}^T(\hat{\mathbf{y}} - \mathbf{y})$ is a column vector of n rows, same to $\mathbf{w}$. This because in (3-6), $\mathbf{X}$ is (n, m) dimensional, whereas $(\hat{\mathbf{y}} - \mathbf{y})$ is (m, 1) dimensional. The result of the multiplication is thus (n, 1) dimensional, which is a column vector of n rows. The same dimensionality allows element-wise subtraction between $\mathbf{w}$ and $\frac{1}{m}\mathbf{X}^T(\hat{\mathbf{y}} - \mathbf{y})$, which is useful in the later gradient descent.

3.2 With regard to b

First, we have:

$$\frac{\partial L(\mathbf{y}, \hat{\mathbf{y}})}{\partial b} = \frac{\partial}{\partial b}[\frac{1}{2m}\sum_{i=1}^{m}(\sum_{k=1}^{n} w_k x_{ik} + b - y_i)^2] \tag{3-8}$$

where b is a scalar.

Following the same logic of section 3.1, for a single training example, we get:

$$\frac{\partial}{\partial b}(\sum_{k=1}^{n} w_k x_{ik} + b - y_i)^2 = 2(\sum_{k=1}^{n} w_k x_{ik} + b - y_i) = 2(\hat{y}_i - y_i) \tag{3-9}$$

For the entire training set, the result is:

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

4. Gradient descent

4.1 With regard to w

According to the formula (3-6), 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). We can do this based on the formula (3-7), which gives us the following:

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

4.2 With regard to b

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

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