# Gradient for linear regression

Pawel Wocjan

February 3, 2020

**Abstract**

We compute the gradient for linear regression.

## 1 Linear regression with single feature

Let $w \in \mathbb{R}$ and $b \in \mathbb{R}$ be the weight and bias for linear regression. Given $x \in \mathbb{R}$, the predicted value is

$$\hat{y} = wx + b. \tag{1}$$

Assume that the correct value for $x$ is $y \in \mathbb{R}$. Then the squared error loss is given by

$$\mathcal{L} = \frac{1}{2}(\hat{y} - y)^2. \tag{2}$$

The gradient of the loss function is

$$\nabla\mathcal{L} = \begin{pmatrix} \dfrac{\partial\mathcal{L}}{\partial b} \\ \\ \dfrac{\partial\mathcal{L}}{\partial w} \end{pmatrix} \in \mathbb{R}^2. \tag{3}$$

Using the chain rule, we compute the bias component of the gradient

$$\frac{\partial\mathcal{L}}{\partial w} = \frac{\partial\mathcal{L}}{\partial\hat{y}} \cdot \frac{\partial\hat{y}}{\partial b} = (\hat{y} - y) \cdot 1 \tag{4}$$

and the weight component of the gradient

$$\frac{\partial\mathcal{L}}{\partial w} = \frac{\partial\mathcal{L}}{\partial\hat{y}} \cdot \frac{\partial\hat{y}}{\partial w} = (\hat{y} - y) \cdot x. \tag{5}$$

Putting these together, we obtain

$$\nabla\mathcal{L} = \begin{pmatrix} 1 \\ x \end{pmatrix} \cdot (\hat{y} - y) \tag{6}$$

## 2 Linear regression for multiple features

### 2.1 Single example

Having dealt with the simple case of linear regression with a single feature in the previous section, we now consider the general case of linear regression with $n$ features.

Let $w = (w_1, \dots, w_n)^T \in \mathbb{R}^n$ and $b \in \mathbb{R}$ be the weight (column) vector and the bias term, respectively. Given the feature (column) vector $x = (x_1, \dots, x_n)^T \in \mathbb{R}^n$, the predicted value by the linear regression model is

$$\hat{y} = \sum_{j=1}^{n} w_j x_j + b. \tag{7}$$

It will be convenient to treat the weights $w_j$ and the bias $b$ in a unified way. To do this, set $w = (w_0, w_1, \dots, w_n)^T \in \mathbb{R}^{n+1}$ with $w_0 = b$ and $x = (1, x_1, \dots, x_n)^T \in \mathbb{R}^{n+1}$. The predicted value is then given by

$$\hat{y} = \sum_{j=0}^{n} w_j x_j. \tag{8}$$

Note that the prediction $\hat{y}$ can also be expressed as the dot product $x^T w$.

The loss is equal to

$$\mathcal{L} = \frac{1}{2}(\hat{y} - y)^2 = \frac{1}{2}(x^T w - y)^2. \tag{9}$$

Using the chain rule to compute the partial derivatives $\partial\mathcal{L}/\partial w_j$ for $j = 0, \dots, n$, we obtain the expression for the gradiet

$$\nabla\mathcal{L} = x(\hat{y} - y) = x(x^T w - y) \in \mathbb{R}^{n+1}. \tag{10}$$

### 2.2 Multiple examples

Let $(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)}) \in \mathbb{R}^{n+1} \times \mathbb{R}$ be the training example in a batch. The loss for the $i$th example is

$$\mathcal{L}^{(i)} = \frac{1}{2}(\hat{y}^{(i)} - y) = \frac{1}{2}({x^{(i)}}^T w - y^{(i)}) \tag{11}$$

and the gradient of the loss for the $i$th example is

$$\nabla\mathcal{L}^{(i)} = x^{(i)}(\hat{y}^{(i)} - y) = x^{(i)}({x^{(i)}}^T w - y^{(i)}). \tag{12}$$

The mean squared error loss for the batch is

$$\mathrm{MSE} = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}^{(i)} \tag{13}$$

and the gradient of the MSE for the batch is

$$\begin{aligned} \nabla\text{MSE} &= \frac{1}{m}\sum_{i=1}^{m}\nabla\mathcal{L}^{(i)} && (14)\\ &= \frac{1}{m}\sum_{i=1}^{m}x^{(i)}({x^{(i)}}^{T}w - y^{(i)}). && (15)\end{aligned}$$

The summation corresponds to a for-loop. To write vectorized code making it possible to process the examples in the batch in parallel, we have to avoid the explicit summation. To this end, define the matrix

$$X = (x^{(1)} \dots x^{(m)}) \in \mathbb{R}^{(n+1)\times m} \tag{16}$$

whose columns are the feature (column) vectors $x^{(i)} \in \mathbb{R}^{n+1}$ and the vector

$$y = (y^{(1)} \dots y^{(m)}) \in \mathbb{R}^{1\times m} \tag{17}$$

whose entries are the labels $y^{(i)} \in \mathbb{R}$.

It is a little bit tricky, but it can be shown that

$$\nabla\text{MSE} = X(X^{T}w - y^{T}). \tag{18}$$

This provides the basis for the vectorized implementation of mini-batch gradient descent in the notebook
`https://colab.research.google.com/drive/1qBxfTPoNcSFvpwu1NDl1V6cHEqL3aQl-`
using the command `numpy.dot`
`https://docs.scipy.org/doc/numpy/reference/generated/numpy.dot.html`.
(Note that in the code we use $X^T$ instead of $X$ and $y^T$ instead of $y$. This is because the first dimension is the batch dimension. I will explain this in class.)