Gradient Descent

Gradient Descent are frequently used in regression optimization by minimizing the loss to get optimal parameters and model fitting, pretty similar with least square method.

Firstly, let's talk about Gradient. It is like the partial derivative in calculus, where we take partial derivative to each entry so that we can get tangent slopes in different directions. Here, in terms of gradient, we just don't call them partial derivatives any more and then put these values into vectors or matrixes and that is what we need.

Next, how about descent? Dating back to calculus, let's think about how to get our needed extreme values. Here, the most general method is to find the slope of the tangent line, plan or something like that in each direction, where by setting these values into zeros and we can finally get the extreme values. Now, just transform our thoughts into gradient so that we firstly start with the gradient for each entry, then try to find values meet all entries with zeros. However, since we know it is not so easily to find the point to meet all the restrictions, thus we require that the machine to approach for that point little by little, also closer and closer to zero, which actually need the movement to be less and less, by such a tendency of descent, we would be more likely to find that extreme point.

For gradient descent, what we need to consider are as follows:

Features or say independent variables, they decide what the predicted model going to be, assuming we have k features

Predicted values or the predicted model,

(i.e. the example for linear regression: $yhat = \beta_0 + \beta_1 x_1 + \cdots + \beta_j x_j$, $1 \le j \le k$

Loss function, that is the difference between predicted values and true values,

usually, we use mean squared errors, that is $loss = h_\beta(x_i) = \frac{1}{n}\Sigma_1^n(yhat - y)^2$, assuming we have n samples

Learning rate, which lead to the descent process, where by decreasing the movement length and keeping a certain direction approach to zero, the optimization process carries on smoothly

What we need is to minimize the loss function by finding the zero gradients respect to estimated parameters, that is to find $(x_1, x_2, \dots, x_k)$ to meet the zero gradient $\beta_1, \beta_2, \dots, \beta_k$

In algebra, we write our loss function as this:

$$loss = \frac{1}{n}\Sigma_{i=1}^n(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_{ik} x_k - y_i)^2$$

To make the function simpler to express, we add a new feature x_0, for each x_{i0}, we set it into 1, that is $x_{i0} = 1$, thus the loss function becomes:

$$loss = \frac{1}{n}\Sigma_{i=1}^n(\beta_0 x_{i0} + \beta_1 x_{i1} + \cdots + \beta_{ik} x_k - y_i)^2$$

$$= \frac{1}{n}\Sigma_{i=1}^{n}(\Sigma_{j=0}^{k}\beta_j x_{ij} - y_i)^2 = \frac{1}{n}\Sigma_{i=0}^{n}\left(h_{\boldsymbol{\beta}}(x_{ij}) - y_i\right)^2 = \frac{1}{n}\Sigma_{i=0}^{n}\left(h_{\boldsymbol{\beta}}(\boldsymbol{x}_i) - y_i\right)^2 = l(\boldsymbol{\beta}),$$

β means vector $(\beta_0, \beta_1, \dots, \beta_k)$, taking the gradient for **β**, we get:

$$\frac{\partial}{\partial \beta_j} l(\beta_0, \beta_1, \dots, \beta_k) = \frac{1}{2n}\frac{\partial}{\partial \beta}\, \mathrm{h}_{\beta}\left(x_j\right)$$

Updating the gradient descent by minus a learning rate of gradient from the original gradient, that is:

$$\beta_j = \beta_j - \alpha\frac{\partial}{\partial \beta_j}(\beta_0, \dots, \beta_k) = \beta_j - \alpha\frac{2}{n}\Sigma_{j=0}^{k}\left(h_{\boldsymbol{\beta}}(x_j) - y_i\right)x_j$$

In matrix,

$[x_{ij}]$ become matrix X, $[y_i]$ becomes matrix **y**, $[\beta_j]$ becomes matrix **β**,

then the loss matrix can be written as $\frac{1}{n}(X\boldsymbol{\beta} - \boldsymbol{y})^T(X\boldsymbol{\beta} - \boldsymbol{y})$,

gradient becomes, $X^T(X\boldsymbol{\beta} - \boldsymbol{y})$,

updated gradient becomes $\boldsymbol{\beta} = \boldsymbol{\beta} - \boldsymbol{\alpha}X^T(X\boldsymbol{\beta} - \boldsymbol{y})$