Gradient descent

JrPhy

Introduction

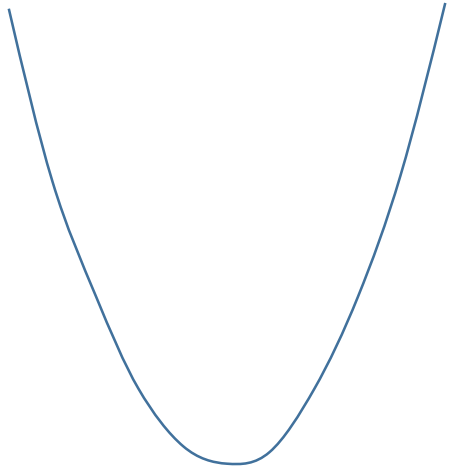
- Artificial Intelligent (AI) is famous today, and its fundamental algorithm is gradient descent, the other algorithms base on this method, like conjugate gradient descent, stochastic gradient descent, ada gradient descent, etc.
- The core idea of gradient descent is finding the minimum, it can be found by solving normal equation then uses QR factorized. Although normal equation gets global minimum definitely, it spends much time, so AI uses gradient descent to modify for the efficiency and accuracy.

Concept

- For a square equation, there is a global maximum or minimum. We can use derivative to get it.
- $y = 2x^2+4x+3$ → $y' = 4x+4 = 0$ minimum is (-1,1)
- Since computer can't compute derivative, so we use finite difference.

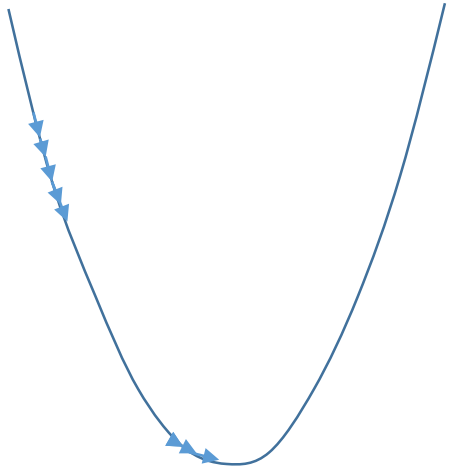
$$y' = \frac{dy}{dx} \approx \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$$

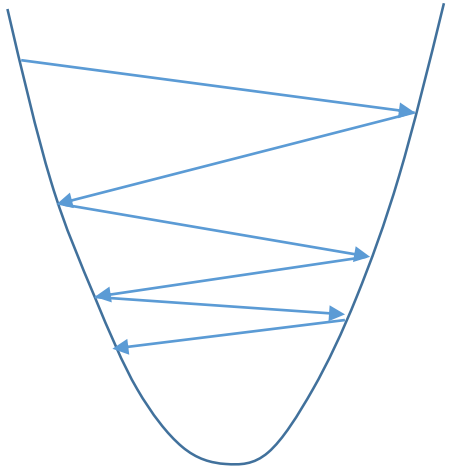
- Its meaning is slope, then choose negative direction to find the minimum.

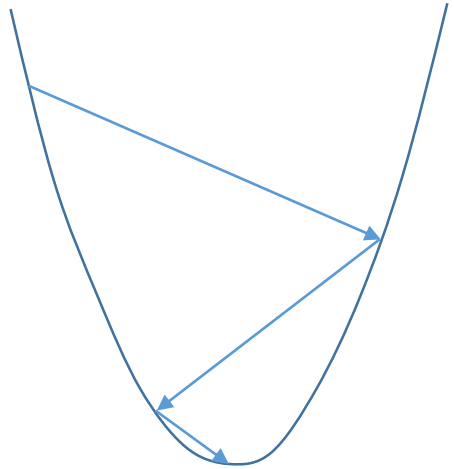


Concept

- Here times a constant λ to try the convergent rate, there is a best constant or function that convergent rate is fastest.
- too slow may not converge the best







Algorithm 1d

- Guess a initial x_0 and λ, then give x_1 to find y'. So

$$x_1' = x_0 - \lambda y'$$

- Because x is a float number, so remember to define error to set the condition.
- The other detail is in the code.

Result

- $y = f(x) = x^2 - 4x + 2$, global minimum at (2, –2)
- $x_0 = 10$, $\lambda = 0.1$ iterators times 57
- The local minimum is: 2.000003, The value is: -2.000000
- You can try my code to change x_0 and λ to see the iterators times

limitation

- For a quadric polynomial, there are 2 local minimum, then how can we know the result is global minimum?
- The answer is no! because 1st derivative can just find the local minimum(maximum), its it limitation.
- And for multi variable, there exists saddle point, this algorithm may stop there, so there are other methods to avoid this issue.

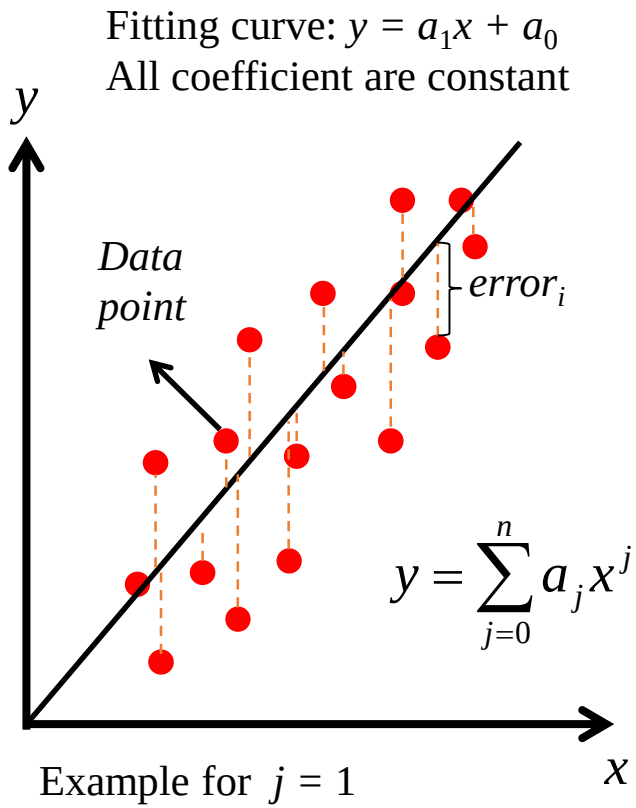
Linear regression by gradient descent

- **Concept**: There are many data (x_i, y_i), after setting on the x-y plane, the distribution may be fitted by some curve $y = f(x)$ or $f(x, y) = k$, k is some constant, $\forall\, x, y, k \in \mathbb{R}$.

- Define $error_i = y_i - y$

- Total error $E = \sum_{i=1}^{n} (y_i - (bx_i + a))^2$

- **Want**: E is minimum value.



Linear regression by gradient descent

- Find the derivative, here are 2 independent variable, so take the partial derivative(the result is the same as the total derivative)

$$\frac{\partial E}{\partial b} = \frac{1}{n}\sum_{i=1}^{n} -2(x_i)(y_i - bx_i - a) \quad \frac{\partial E}{\partial a} = \frac{1}{n}\sum_{i=1}^{n} -2(y_i - bx_i - a)$$

- Divide by n because of n data. Then apply the algorithm

$$b_{i+1} = b_i - \lambda \frac{1}{n}\sum_{i=1}^{n} -2(x_i)(y_i - bx_i - a) \quad a_{i+1} = a_i - \lambda \frac{1}{n}\sum_{i=1}^{n} -2(y_i - bx_i - a)$$

- λ is the step size determined by you.

Polynomial regression by gradient descent

- Most of the procedure are the same as linear regression, only the error is different.
- Suppose we want to regress a polynomial with degree *m*, then

$$y = \sum_{i=0}^{n} a_i x^i \qquad E = \sum_{j=1}^{n} (y_j - (\sum_{i=0}^{m} a_i x^i))^2 \qquad \frac{\partial E}{\partial a_i} = \frac{\lambda}{n} \sum_{j=1}^{n} 2(y_j - (\sum_{i=0}^{m} a_i x^i))$$

- Here I use superscript *k* to represent the next coefficient

$$a_2^{k+1} = a_2^k - \frac{\lambda}{n} \sum_{j=1}^{n} 2x^2 (y_j - (\sum_{i=0}^{2} a_i^k x^i))$$

Polynomial regression by gradient descent

- Here you should be careful about the numerical problem, we take power of 2 as example. Its formula is

$$a_2^{k+1} = a_2^k - \frac{\lambda}{n}\sum_{j=1}^{n} 2x^2(y_j - (\sum_{i=0}^{2} a_i^k x^i))$$

- and its degree is 4, that means if you use a polynomial with its degree bigger than 2, you have to use much smaller λ to ensure it will converge.