Optimization

Friday, 30 April 2021 07:42

Gradient Descent

- In general used to find a minimum of any function
- In ML mostly used to find a minimum of the error function (typically MSE) dependent on model parameters

General formulation: $\min_{\vec{x}=(x_1,\dots,x_n)} f(x_1,\dots,x_n)$

ML formulation (special case of the above): $\min_{\vec{\theta}=(\theta_1,\dots,\theta_n)} \sum_{(\vec{x},y)\in D} \left(y - f(\vec{x}\mid\vec{\theta})\right)^2$

training data

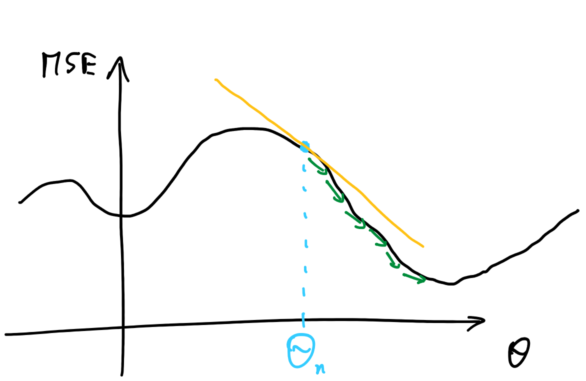
Example

$$\min_{(\theta_0,\theta_1)} \sum_{(x,y)\in D} \left(y - (\theta_1 x + \theta_0)\right)^2$$

$$f(x\mid\theta) = \theta_1 x + \theta_0$$

Idea of GD

1. Start with any $\vec{\theta}$
2. Iteratively move $\vec{\theta}$ in the direction opposite to the derivative



The slope of the tangent line is equal to the derivative of the function with respect to θ

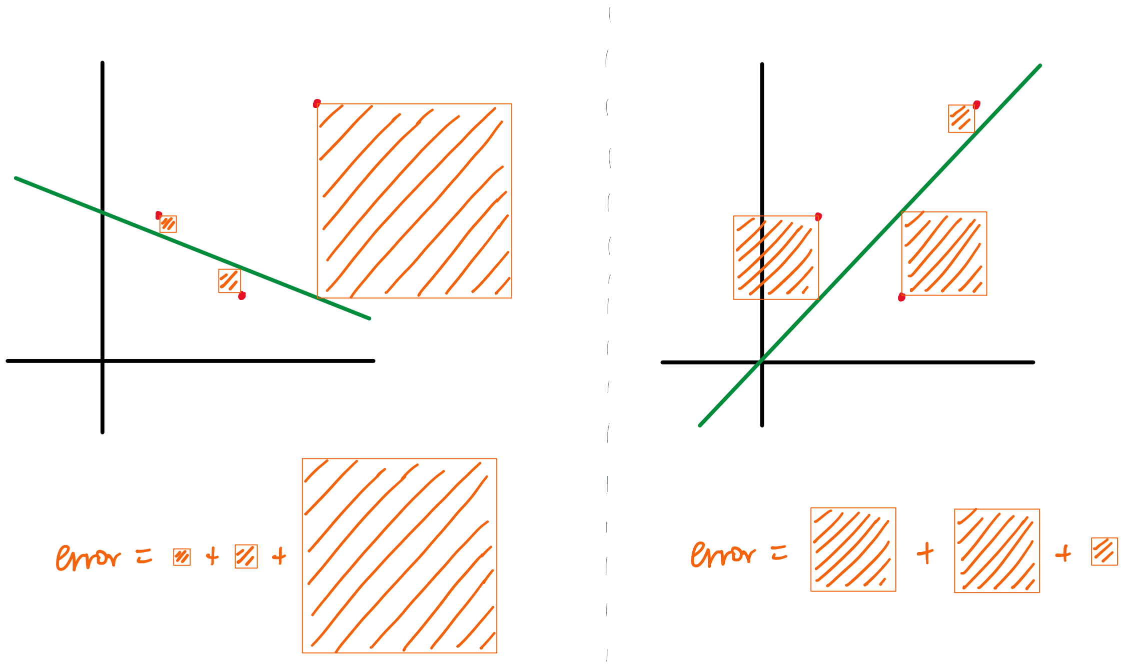
$$\frac{\partial MSE}{\partial \theta}$$

Derivative calculation for MSE

For simplicity we assume a linear model

$$f(x \mid \theta_0, \theta_1) = f(x, \theta_0, \theta_1) = \theta_1 x + \theta_0$$

$$MSE = \frac{1}{|D|} \sum_{(x,y) \in D} (y - (\theta_1 x + \theta_0))^2$$



Since $(f+g)'(x) = f'(x) + g'(x)$ and $(a f(x))' = a f'(x)$
we can make the derivative calculations for a single element in the sum (single datapoint) and then average.

$$\frac{\partial}{\partial \theta_0} (y - (\theta_1 x + \theta_0))^2 = 2(y - (\theta_1 x + \theta_0)) \cdot \frac{\partial}{\partial \theta_0} (y - (\theta_1 x + \theta_0))$$

$$= 2(y - (\theta_1 x + \theta_0))(-1)$$

$$= -2(y - (\theta_1 x + \theta_0))$$

$$\frac{\partial}{\partial \theta_1} (y - (\theta_1 x + \theta_0))^2 = 2(y - (\theta_1 x + \theta_0)) \cdot \frac{\partial}{\partial \theta_1} (y - (\theta_1 x + \theta_0))$$

$$= 2(y - (\theta_1 x + \theta_0))(-x)$$

$$= -2x\,(y - (\theta_1 x + \theta_0))$$

Updating rule

$$\boxed{\vec{\theta}^{\,n} = \vec{\theta}^{\,n-1} - \alpha \frac{\partial}{\partial \theta} MSE(\vec{\theta})}$$

$$\theta_0^n = \theta_0^{n-1} - \alpha \frac{\partial}{\partial \theta_0} MSE(\theta_0^{n-1}, \theta_1^{n-1})$$

$$= \theta_0^{n-1} - \alpha \frac{1}{|D|} \sum_{(x,y)\in D} (-2)(y - (\theta_1 x + \theta_0))$$

$$\theta_1^n = \theta_1^{n-1} - \alpha \frac{\partial}{\partial \theta_1} MSE(\theta_0^{n-1}, \theta_1^{n-1})$$

$$= \theta_1^{n-1} - \alpha \frac{1}{|D|} \sum_{(x,y)\in D} (-2x)(y - (\theta_1 x + \theta_0))$$

Stochastic Gradient Descent (SGD)

1. Start with any $\vec{\theta}$
2. Iteratively:
   a) take a datapoint $(\vec{x}, y) \in D$
   b) calculate the derivative of MSE on this single datapoint with respect to $\vec{\theta}$
   c) shift θ in the direction opposite to the derivative

Problem: SGD can be unstable and diverge

Mini-batch Gradient Descent

Remark: most of the time when people say SGD they mean Mini-batch GD

1. Start with any $\vec{\theta}$

2. Iteratively:
   a) take a mini-batch $\{(\vec{x}, y)\} \subset D$
   b) calculate the derivative of MSE on this mini batch with respect to $\vec{\theta}$
   c) shift θ in the direction opposite to the derivative

There are many other variants of SGD used in practice:

- SGD with momentum
- Rmsprop
- NAG
- Adam (the most popular)
- AdaGrad
- AdaDelta

Stochastic Gradient Descent for matrix factorization

$$\min_{\{p_u, q_i\}} \underbrace{\sum_{(u,i)\in K} \underbrace{(r_{ui} - q_i^T p_u)^2 + \overbrace{\lambda(\|q_i\|^2 + \|p_u\|^2)}^{L_2 \text{ regularization term}}}_{\text{error}(u,i)}}_{\text{error}}$$

For simplicity consider embedding dim = 2

$$p_u = (p_{u1}, p_{u2})$$

$$q_i = (q_{i1}, q_{i2})$$

The stochastic gradient descent step looks as follows

$$p_{u1}^{(n)} = p_{u1}^{(n-1)} - \alpha \frac{\partial}{\partial p_{u1}} \text{error}(u,i)$$

$$p_{u2}^{(n)} = p_{u2}^{(n-1)} - \alpha \frac{\partial}{\partial p_{u2}} \text{error}(u,i)$$

$$q_{i1}^{(n)} = q_{i1}^{(n-1)} - \alpha \frac{\partial}{\partial q_{i1}} \text{error}(u,i)$$

$$q_{i2}^{(n)} = q_{i2}^{(n-1)} - \alpha \frac{\partial}{\partial q_{i2}} \text{error}(u,i)$$

Substituting the formula for the error we get

$$\frac{\partial}{\partial p_{u1}} \text{error} = \frac{\partial}{\partial p_{u1}} \left(\left(r_{ui} - (q_{i1} p_{u1} + q_{i2} p_{u2}) \right)^2 + \lambda \left(q_{i1}^2 + q_{i2}^2 + p_{u1}^2 + p_{u2}^2 \right) \right)$$

$$= 2\left(r_{ui} - (q_{i1} p_{u1} + q_{i2} p_{u2})\right)(-q_{i1}) + 2\lambda p_{u1}$$

Denote

$$e_{ui} = r_{ui} - (q_{i1} p_{u1} + q_{i2} p_{u2})$$

Then

$$\frac{\partial}{\partial p_{u1}} \text{error} = -2 e_{ui} q_{i1} + 2\lambda p_{u1}$$

Analogously for p_{u2}, q_{i1}, q_{i2}

$$\frac{\partial}{\partial p_{u2}} \text{error} = -2 e_{ui} q_{i2} + 2\lambda p_{u2}$$

$$\frac{\partial}{\partial q_{u1}} \text{error} = -2 e_{ui} p_{i1} + 2\lambda q_{u1}$$

$$\frac{\partial}{\partial q_{u2}} \text{error} = -2 e_{ui} p_{i2} + 2\lambda q_{u2}$$

Therefore the updating formula for SGD looks as follows:

$$\left(p_{u1}^{(n)}, p_{u2}^{(n)}\right) = \left(p_{u1}^{(n-1)}, p_{u2}^{(n-1)}\right) - \alpha \left(\frac{\partial}{\partial p_{u1}} \text{error}, \frac{\partial}{\partial p_{u2}} \text{error} \right)$$

$$\left(p_{u1}^{(n)}, p_{u2}^{(n)}\right) = \left(p_{u1}^{(n-1)}, p_{u2}^{(n-1)}\right) - \alpha\left(\frac{\partial}{\partial p_{u1}} \text{error}, \frac{\partial}{\partial p_{u2}} \text{error}\right)$$

$$= \left(p_{u1}^{(n-1)}, p_{u2}^{(n-1)}\right) - \alpha\left(-2e_{ui}^{(n-1)} q_{i1}^{(n-1)} + 2\lambda p_{u1}^{(n-1)}, -2e_{ui}^{(n-1)} q_{i2}^{(n-1)} + 2\lambda p_{u2}^{(n-1)}\right)$$

$$= \left(p_{u1}^{(n-1)}, p_{u2}^{(n-1)}\right) + 2\alpha\left(e_{ui}^{(n-1)} q_{i1}^{(n-1)} - \lambda p_{u1}^{(n-1)}, e_{ui}^{(n-1)} q_{i2}^{(n-1)} - \lambda p_{u2}^{(n-1)}\right)$$

Analogously for q_{i1}, q_{i2}

Rewriting the above equations in vector form gives the following formulation (renaming α to be 2α)

$$p_u^{(n)} = p_u^{(n-1)} + \alpha\left(e_{ui}^{(n-1)} q_i^{(n-1)} - \lambda p_u^{(n-1)}\right)$$

$$q_i^{(n)} = q_i^{(n-1)} + \alpha\left(e_{ui}^{(n-1)} p_u^{(n-1)} - \lambda q_i^{(n-1)}\right)$$