

[37.随机梯度下降法.note](#) **随机梯度下降法**：对普通梯度下降法的改进，使算法能应用于大数据集中。

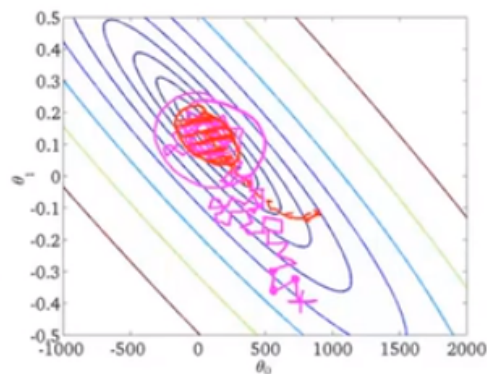
普通梯度下降法，也就是批量梯度下降法中，每次参数迭代，会计算m次加和，当数量级很大的时候，每次计算需要耗费很长时间。

使用随机梯度下降法，不同的样本数据之间就可以改变参数，不用遍历所有数据就可以找到参数sita的收敛值。

| Batch gradient descent | Stochastic gradient descent |
|--|---|
| $J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$ | $cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2} (h_{\theta}(x^{(i)}) - y^{(i)})^2$ |
| $J_{train}(\theta) = \frac{1}{m} \sum_{i=1}^m cost(\theta, (x^{(i)}, y^{(i)}))$ | $J_{train}(\theta) = \frac{1}{m} \sum_{i=1}^m cost(\theta, (x^{(i)}, y^{(i)}))$ |
| Repeat { $\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$ $\frac{\partial}{\partial \theta_j} J_{train}(\theta)$ (for every $j = 0, \dots, n$) } $m = 300,000,000$ | 1. Randomly shuffle dataset. ← 2. Repeat { for $i = 1, \dots, m$ { $\theta_j := \theta_j - \alpha (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$ (for every $j = 0, \dots, n$) } } $\frac{\partial}{\partial \theta_j} cost(\theta, (x^{(i)}, y^{(i)}))$ $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots$ |

随机梯度下降步骤：

- Stochastic gradient descent**
- 1. Randomly shuffle (reorder) training examples
 - 2. Repeat { 1-10x
 - for $i := 1, \dots, m$ {
 - $$\theta_j := \theta_j - \alpha (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$
 (for every $j = 0, \dots, n$)
 - }
- $m = 300,000,000$



尽管可能每一步不一定向着全局最小值前进，但总体趋势是走向全局最小值，且不用每一步梯度下降都计算全局数据。