梯度下降算法

日期： 2019-3-15

作者： lunyang liu

在求解机器学习算法的模型参数，即无约束优化问题时，梯度下降（Gradient Descent）是最常采用的方法之一，另一种常用的方法是最小二乘法。这里就对梯度下降法做一个完整的总结。

1. 梯度

在微积分里面，对多元函数的参数求∂偏导数，把求得的各个参数的偏导数以向量的形式写出来，就是梯度。比如函数$f(x,y)$, 分别对x,y求偏导数，求得的梯度向量就是$(\partial f/\partial x, \partial f/\partial y)^T$,简称$grad f(x,y)$或者$\Delta f(x,y)$。对于在点$(x_0,y_0)$的具体梯度向量就是$(\partial f/\partial x_0, \partial f/\partial y_0)^T$或者$\Delta f(x_0,y_0)$，如果是3个参数的向量梯度，就是$(\partial f/\partial x, \partial f/\partial y, \partial f/\partial z)^T$,以此类推。

那么这个梯度向量求出来有什么意义呢？他的意义从几何意义上讲，就是**函数变化增加最快的地方**。具体来说，对于函数$f(x,y)$,在点(x_0,y_0)，沿着梯度向量的方向就是$(\partial f/\partial x_0, \partial f/\partial y_0)^T$的方向是$f(x,y)$增加最快的地方。或者说，沿着梯度向量的方向，更加容易找到函数的最大值。反过来说，沿着梯度向量相反的方向，也就是$-(\partial f/\partial x_0, \partial f/\partial y_0)^T$的方向，梯度减少最快，也就是更加容易找到函数的最小值。

2. 梯度下降与梯度上升

在机器学习算法中，在最小化损失函数时，可以通过梯度下降法来一步步的迭代求解，得到最小化的损失函数，和模型参数值。反过来，如果我们需要求解损失函数的最大值，这时就需要用梯度上升法来迭代了。

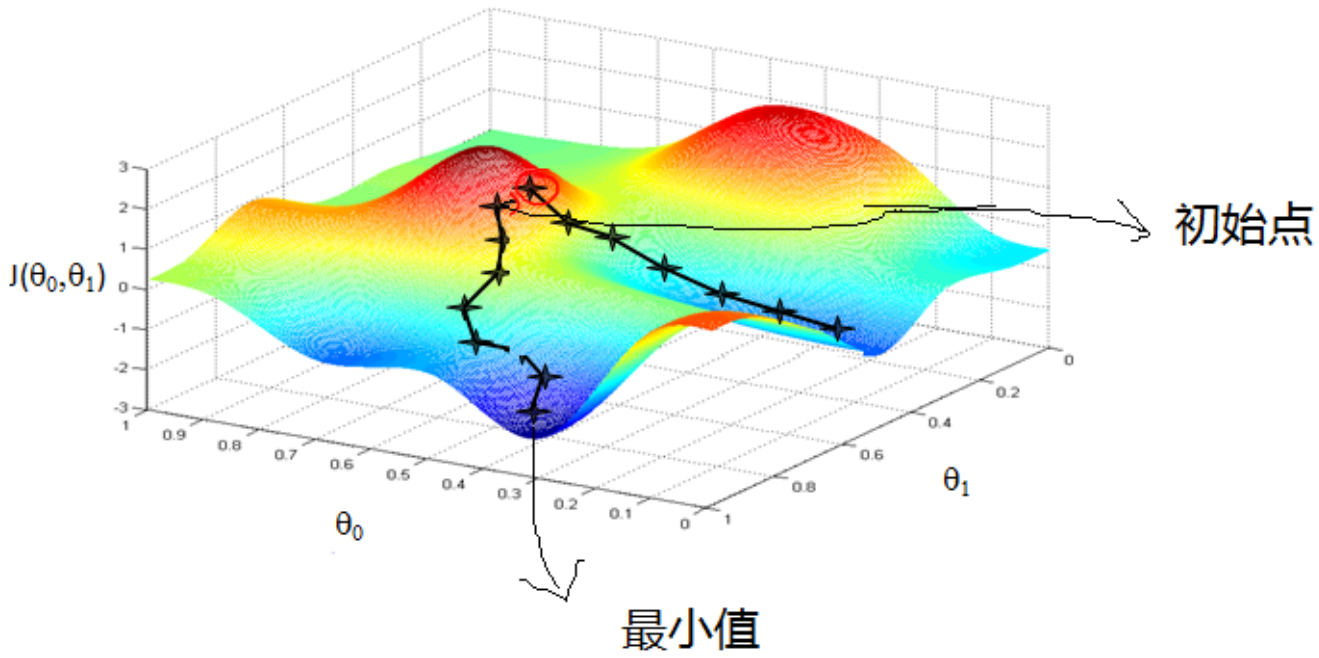
梯度下降法和梯度上升法是可以互相转化的。比如我们需要求解损失函数$f(\theta)$的最小值，这时我们需要用梯度下降法来迭代求解。但是实际上，我们可以反过来求解损失函数$-f(\theta)$的最大值，这时梯度上升法就派上用场了。

3. 梯度下降法算法详解

3.1 梯度下降的直观解释

首先来看看梯度下降的一个直观的解释。比如我们在一座大山上的某处位置，由于我们不知道怎么下山，于是决定走一步算一步，也就是在每走到一个位置的时候，求解当前位置的梯度，沿着梯度的负方向，也就是当前最陡峭的位置向下走一步，然后继续求解当前位置梯度，向这一步所在位置沿着最陡峭最易下山的位置走一步。这样一步步的走下去，一直走到觉得我们已经到了山脚。当然这样走下去，有可能我们不能走到山脚，而是到了某一个局部的山峰低处。

从上面的解释可以看出，梯度下降不一定能够找到全局的最优解，有可能是一个局部最优解。当然，如果损失函数是凸函数，梯度下降法得到的解就一定是全局最优解。



到此，我们就有个疑问了，**为什么沿着梯度负方向走就能得到最小值呢？**接着看。

3.1.1 最优化问题

- **最优化问题是求解函数极值的问题**，包括极大值和极小值。
- 微积分为我们求函数的极值提供了一个统一的思路：找函数的导数等于0的点，因为在极值点处，导数必定为0。这样，只要函数的可导的，我们就可以用这个万能的方法解决问题，幸运的是，在实际应用中我们遇到的函数基本上都是可导的。
- 机器学习之类的实际应用中，我们一般将最优化问题统一表述为求解函数的极小值问题，即:

$$\min_x f(x)$$

- 其中x称为优化变量，f称为目标函数。极大值问题可以转换成极小值问题来求解，只需要将目标函数加上负号即可：

$$\min_x -f(x)$$

3.1.2 导数与梯度

- 梯度是多元函数对各个自变量偏导数形成的向量。多元函数的梯度表示：

$$\nabla f(x) = (\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n})^T$$

- 如果Hessian矩阵正定，函数有极小值；如果Hessian矩阵负定，函数有极大值；如果Hessian矩阵不定，则需要进一步讨论。
- 如果二阶导数大于0，函数有极小值；如果二阶导数小于0，函数有极大值；如果二阶导数等于0，情况不定。

问题：为何不直接求导，令导数等于零去求解？

- 直接求函数的导数，有的函数的导数方程组很难求解，比如下面的方程：

$$f(x,y) = x^5 + e^x y - y^3 + 10y^2 - 100\sin(xy) - 2x^2$$

3.2 梯度下降的推导过程

- 回顾一下泰勒展开式

$$f(x)=\frac{f(x_0)}{0!}+\frac{f'(x_0)}{1!}(x-x_0)+\frac{f''(x_0)}{2!}(x-x_0)^2+\ldots+\frac{f^{(n)}(x_0)}{n!}(x-x_0)^n+R_n(x)$$

- 多元函数f(x)在x处的泰勒展开：

$$f(x+\Delta x)=f(x)+f'(x)\Delta x+\frac{1}{2}f''(x)\Delta x^2$$

目标是求多元函数f(x)的极小值。**梯度下降法**是通过不断迭代得到函数极小值，即如能保证$f(x+\Delta x)$比$f(x)$小，则不断迭代，最终能得到极小值。想象你在山顶往山脚走，如果每一步到的位置比之前的位置低，就能走到山脚。问题是像哪个方向走，能最快到山脚呢? 由泰勒展开式得:

$$f(x+\Delta x)-f(x)=(\Delta f(x))^T\Delta x+o(\Delta x)$$

如果Δx足够小，可以忽略$o(\Delta x)$，则有：

$$f(x+\Delta x)-f(x)=(\Delta f(x))^T\Delta x$$

于是只有：

$$(\Delta f(x))^T\Delta x<0$$

能使

$$f(x+\Delta x)<f(x)$$

因为`∇f(x)`与`Δx`均为向量，于是有：

$$(\Delta f(x))^T\Delta x=\|\Delta f(x)\|\,\|\Delta x\|\cos\theta$$

其中，θ是向量$\Delta f(x)$与Δx的夹角，$\|\nabla f(x)\|$与$\|\Delta x\|$是向量对应的模。可见只有当$cos\theta<0$才能使得$(\nabla f(x))^T\Delta x<0$,又因$cos\theta\geq-1$, 可见，只有当$cos(\theta)=-1$, 即$\theta=\pi$时，函数数值降低最快。此时梯度和$\Delta x$反向，即夹角为180度。因此当向量$\Delta x$的模大小一定时，取

$$\Delta x=-\alpha\Delta f(x)$$

即在梯度相反的方向函数值下降的最快。此时函数的下降值为：

$$(\Delta f(x))^T\Delta x=-\|\Delta f(x)\|\,\|\Delta x\|=-\alpha\|\Delta f(x)\|^2$$

只要梯度不为0，往梯度的反方向走函数值一定是下降的。直接用可能会有问题，因为`x+Δx`可能会超出x的邻域范围之外，此时是不能忽略泰勒展开中的二次及以上的项的，因此步伐不能太大。

一般设：

$$\Delta x=-\alpha\Delta f(x)$$

其中`α`为一个接近于0的正数，称为**步长**，由人工设定，用于保证$x+\Delta x$在`x`的邻域内，从而可以忽略泰勒展开中二次及更高的项，则有:

$$(\Delta f(x))^T\Delta x=-\|\Delta f(x)\|\,\|\Delta x\|=-\alpha\|\Delta f(x)\|^2<0$$

此时，x的迭代公式是：

$$x_{k+1} = x_k - \alpha \Delta f(x_k)$$

只要没有到达梯度为0的点，则函数值会沿着序列x_k递减，最终会收敛到梯度为0的点，这就是梯度下降法。迭代终止的条件是函数的梯度值为0（实际实现时是接近于0），此时认为已经达到极值点。注意我们找到的是梯度为0的点，这不一定就是极值点。

3.3 梯度下降与机器学习相关的概念

1. 步长（**Learning rate**）：步长决定了在梯度下降迭代的过程中，每一步沿梯度负方向前进的长度。用上面下山的例子，步长就是在当前这一步所在位置沿着最陡峭最易下山的位置走的那一步的长度。

2.特征（**feature**）：指的是样本中输入部分，比如2个单特征的样本(x_0, y_0)，(x_1, y_1),则第一个样本特征为x_0，第一个样本输出为y_0。

3. 假设函数（**hypothesis function**）：在监督学习中，为了拟合输入样本，而使用的假设函数，记为$h_\theta(x)$。比如对于单个特征的m个样本$(x_i, y_i)(i = 1, 2, \ldots, m)$,可以采用拟合函数如下： $h_\theta(x) = \theta_0 + \theta_1 x$。
4. 损失函数（**loss function**）：为了评估模型拟合的好坏，通常用损失函数来度量拟合的程度。损失函数极小化，意味着拟合程度最好，对应的模型参数即为最优参数。在线性回归中，损失函数通常为样本输出和假设函数的差取平方。比如对于m个样本$(x_i, y_i)(i = 1, 2, \ldots m)$,采用线性回归，损失函数为：

$$J(\theta_0, \theta_1) = \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

其中x_i表示第i个样本特征，y_i表示第i个样本对应的输出，$h_\theta(x_i)$为假设函数。

3.4 梯度下降的描述方式

梯度下降法的算法可以有**代数法**和**矩阵法**（也称向量法）两种表示，如果对矩阵分析不熟悉，则代数法更加容易理解。不过矩阵法更加的简洁，且由于使用了矩阵，实现逻辑更加的一目了然。这里先介绍代数法，后介绍矩阵法。

3.4.1 代数方式描述

1. **先决条件**： 确认优化模型的假设函数和损失函数。比如对于线性回归，假设函数表示为 $h_\theta(x_1, x_2, \ldots, x_n) = \theta_0 + \theta_1 x_1 + \ldots + \theta_n x_n$, 其中$\theta_i (i = 0, 1, 2 \ldots n)$为模型参数，$x_i (i = 0, 1, 2 \ldots n)$为每个样本的n个特征值。这个表示可以简化，我们增加一个特征$x_0 = 1$ ，这样:

$$h_\theta(x_0, x_1, \ldots, x_n) = \sum_{i=0}^{n} \theta_i x_i$$

同样是线性回归，对应于上面的假设函数，损失函数为：

$$J(\theta_0, \theta_0, \ldots, \theta_n,) = \frac{1}{2m} \sum_{j=0}^{m} (h_\theta(x_0^j, x_1^j, \ldots, x_n^j) - y_i)^2$$

2. **算法相关参数初始化**：主要是初始化$\theta_0, \theta_1 \ldots, \theta_n$,算法终止距离$\varepsilon$以及步长$\alpha$。在没有任何先验知识的时候，可以将所有的θ初始化为0， 将步长初始化为1。在调优的时候再优化。
3. **算法过程**：

1）确定当前位置的损失函数的梯度，对于θ_i,其梯度表达式如下：

$$\frac{\partial}{\partial\theta_i}J(\theta_0,\theta_0,\ldots,\theta_n)$$

2）用步长乘以损失函数的梯度，得到当前位置下降的距离，即

$$\alpha\frac{\partial}{\partial\theta_i}J(\theta_0,\theta_0,\ldots,\theta_n)$$

3）确定是否所有的θ_i，梯度下降的距离都小于ε，如果小于ε则算法终止，当前所有的$\theta_i(i=0,1,...n)$即为最终结果。否则进入步骤4.

4）更新所有的θ，对于θ_i，其更新表达式如下。更新完毕后继续转入步骤1.

$$\theta_i=\theta_i-\alpha\frac{\partial}{\partial\theta_i}J(\theta_0,\theta_0,\ldots,\theta_n)$$

下面用**线性回归**的例子来具体描述梯度下降。假设我们的样本是
$(x_1^{(0)},x_2^{(0)},..x_n^{(0)}),(x_1^{(1)},x_2^{(1)},..x_n^{(1)}),\ldots,(x_1^{(m)},x_2^{(m)},..x_n^{(m)})$,

损失函数如前面先决条件所述：

$$J(\theta_0,\theta_0,\ldots,\theta_n)=\frac{1}{2m}\sum_{j=0}^{m}(h_\theta(x_0^{(j)},x_0^{(j)},\ldots,x_0^{(j)})-y_i)^2$$

则在算法过程步骤1中对于θ_i的偏导数计算如下：

$$\frac{\partial}{\partial\theta_i}J(\theta_0,\theta_0,\ldots,\theta_n)=\frac{1}{m}\sum_{j=0}^{m}(h_\theta(x_0^{(j)},x_0^{(j)},\ldots,x_0^{(j)})-y_i)x_i^{(j)}$$

由于样本中没有x_0上式中令所有的x_0^j为1.

步骤4中θ_i的更新表达式如下：

$$\theta_i=\theta_i-\alpha\frac{1}{m}\sum_{j=0}^{m}(h_\theta(x_0^{(j)},x_0^{(j)},\ldots,x_0^{(j)})-y_i)x_i^j$$

从这个例子可以看出当前点的梯度方向是由所有的样本决定的，加$\frac{1}{m}$是为了好理解。由于步长也为常数，他们的乘机也为常数，所以这里$\alpha\frac{1}{m}$可以用一个常数表示。

在下面会讲到梯度下降法的变种，他们主要的区别就是**对样本**的采用方法不同。这里我们采用的是用所有样本。

3.4.2 矩阵方式描述

这一部分主要讲解梯度下降法的矩阵方式表述，相对于上节的代数法，要求有一定的矩阵分析的基础知识，尤其是矩阵求导的知识。

1. **先决条件**：类似的， 需要确认优化模型的假设函数和损失函数。对于线性回归，假设函数
   $h_\theta(x_1,x_2,,\ldots x_n)=\theta_0+\theta_1x_1+\ldots+\theta_nx_n$的矩阵表达方式为：$h_\theta(X)=X\theta$，其中， 假设函数$h_\theta(X)$为

$m \times 1$的向量,θ为$(n+1) \times 1$的向量，里面有$n+1$个代数法的模型参数。**X**为$m \times (n+1)$维的矩阵。m代表样本的个数，$n+1$代表样本的特征数。

损失函数的表达式为：$J(\theta) = \frac{1}{2}(X\theta - Y)^T(X\theta - Y)$, 其中Y是样本的输出向量，维度为$m \times 1$.

2. 算法相关参数初始化: θ向量可以初始化为默认值，或者调优后的值。算法终止距离ε，步长α。
3. 算法过程：

1）确定当前位置的损失函数的梯度，对于θ向量,其梯度表达式为：$\frac{\partial}{\partial\theta}J(\theta)$

2）用步长乘以损失函数的梯度，得到当前位置下降的距离，即$\alpha\frac{\partial}{\partial\theta}J(\theta)$对应于前面登山例子中的某一步。

3）确定θ向量里面的每个值,梯度下降的距离都小于ε，如果小于ε则算法终止，当前θ向量即为最终结果。否则进入步骤4.

4）更新θ向量，其更新表达式如下。更新完毕后继续转入步骤1.

$$\theta = \theta - \alpha\frac{\partial}{\partial\theta}J(\theta)$$

还是用线性回归的例子来描述具体的算法过程。

损失函数对于θ向量的偏导数计算如下：

$$\frac{\partial}{\partial\theta}J(\theta) = X^T(X\theta - Y)$$

步骤4中θ向量的更新表达式如下：$\theta = \theta - \alpha X^T(X\theta - Y)$

这里面用到了矩阵求导链式法则，和两个个矩阵求导的公式。

公式1：$\frac{\partial}{\partial X}(X^TX) = 2X$,`x`为向量

公式2：$\nabla_X f(AX+B) = A^T\nabla_Y f, Y = AX+B$,$f(Y)$为标量

如果需要熟悉矩阵求导建议参考张贤达的《矩阵分析与应用》一书。

3.5 梯度下降的算法调优

在使用梯度下降时，需要进行调优。哪些地方需要调优呢？

1. 算法的步长选择。在前面的算法描述中，提到取步长为1，但是实际上取值取决于数据样本，可以多取一些值，从大到小，分别运行算法，看看迭代效果，如果损失函数在变小，说明取值有效，否则要增大步长。前面说了。步长太大，会导致迭代过快，甚至有可能错过最优解。步长太小，迭代速度太慢，很长时间算法都不能结束。所以算法的步长需要多次运行后才能得到一个较为优的值。
2. 算法参数的初始值选择。 初始值不同，获得的最小值也有可能不同，因此梯度下降求得的只是局部最小值；当然如果损失函数是凸函数则一定是最优解。由于有局部最优解的风险，需要多次用不同初始值运行算法，关键损失函数的最小值，选择损失函数最小化的初值。

3.归一化。由于样本不同特征的取值范围不一样，可能导致迭代很慢，为了减少特征取值的影响，可以对特征数据归一化，也就是对于每个特征x，求出它的期望$\bar{x}$和标准差$std(x)$，然后转化为：

$$\frac{x - \bar{x}}{std(x)}$$

这样特征的新期望为0，新方差为1，迭代次数可以大大加快。

4. 梯度下降法大家族（BGD，SGD，MBGD）

4.1 批量梯度下降法（Batch Gradient Descent）

批量梯度下降法，是梯度下降法最常用的形式，具体做法也就是在更新参数时使用**所有的样本**来进行更新，这个方法对应于前面线性回归的梯度下降算法。

$$\theta_i = \theta_i - \alpha \sum_{j=0}^{m} (h_\theta(x_0^{(j)}, x_1^{(j)}, \ldots, x_n^{(j)},) - y_i) x_i^{(j)}$$

由于我们有m个样本，这里求梯度的时候就用了所有m个样本的梯度数据。

4.2 随机梯度下降法（Stochastic Gradient Descent）

随机梯度下降法，其实和批量梯度下降法原理类似，区别在与求梯度时没有用所有的m个样本的数据，而是仅仅**选取一个样本j**来求梯度。对应的更新公式是：

$$\theta_i = \theta_i - \alpha (h_\theta(x_0^{(j)}, x_1^{(j)}, \ldots, x_n^{(j)},) - y_i) x_i^{(j)}$$

随机梯度下降法，和批量梯度下降法是两个极端，一个采用所有数据来梯度下降，一个用一个样本来梯度下降。自然各自的优缺点都非常突出。对于训练速度来说，随机梯度下降法由于每次仅仅采用一个样本来迭代，训练速度很快，而批量梯度下降法在样本量很大的时候，训练速度不能让人满意。对于准确度来说，随机梯度下降法用于仅仅用一个样本决定梯度方向，导致解很有可能不是最优。对于收敛速度来说，由于随机梯度下降法一次迭代一个样本，导致迭代方向变化很大，不能很快的收敛到局部最优解。

那么，有没有一个中庸的办法能够结合两种方法的优点呢？有！这就是**小批量梯度下降法**。

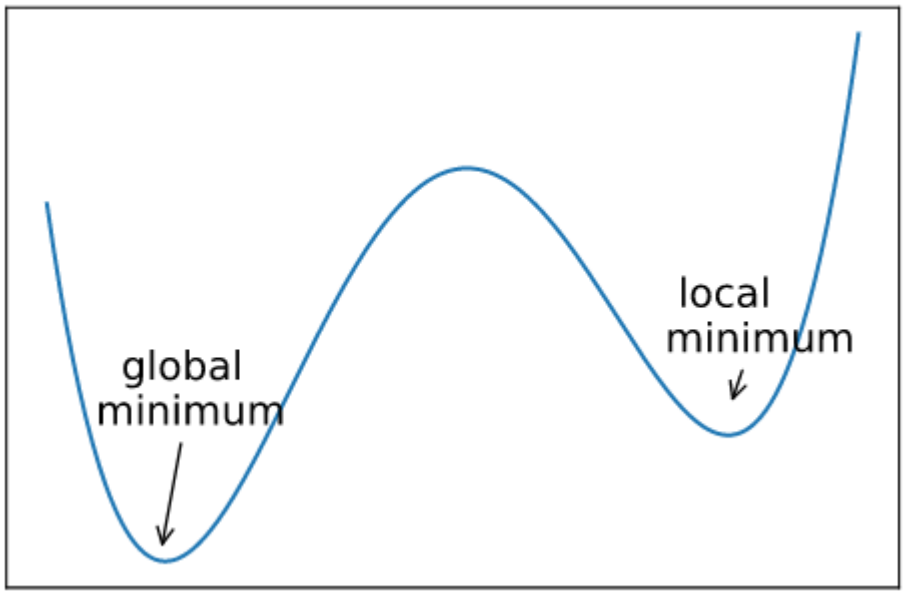
4.3 小批量梯度下降法（Mini-batch Gradient Descent）

小批量梯度下降法是批量梯度下降法和随机梯度下降法的折衷，也就是对于**m个样本**，我们采用x个样子来迭代，$1 < x < m$。一般可以取$x = 10$，当然根据样本的数据，可以调整这个x的值。对应的更新公式是：

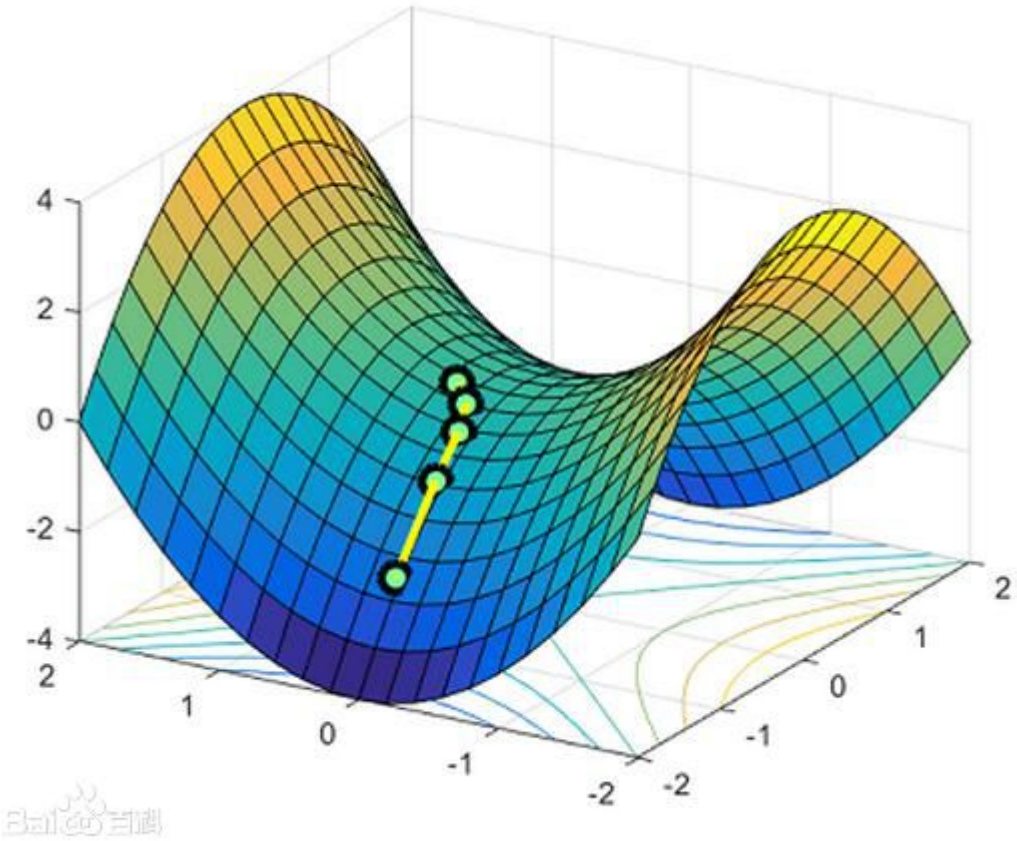
$$\theta_i = \theta_i - \alpha \sum_{j=t}^{t+x-1} (h_\theta(x_0^{(j)}, x_1^{(j)}, \ldots, x_n^{(j)},) - y_i) x_i^{(j)}$$

5 存在的问题

- 局部极小值
  - 梯度下降可能在局部最小的点收敛。



- 鞍点
  - 鞍点是指梯度为0，Hessian矩阵既不是正定也不是负定，即不定的点。如函数x^2-y^2在(0,0)点梯度为0，但显然不是局部最小的点，也不是全局最小的点。



6. 梯度下降法和其他无约束优化算法的比较

在机器学习中的无约束优化算法，除了梯度下降以外，还有前面提到的`最小二乘法`，此外还有`牛顿法`和`拟牛顿法`。

梯度下降法和最小二乘法相比，梯度下降法需要选择步长，而最小二乘法不需要。梯度下降法是迭代求解，最小二乘法是计算解析解。如果样本量不算很大，且存在解析解，最小二乘法比起梯度下降法要有优势，计算速度很快。但是如果样本量很大，用最小二乘法由于需要求一个超级大的逆矩阵，这时就很难或者很慢才能求解解析解了，使用迭代的梯度下降法比较有优势。