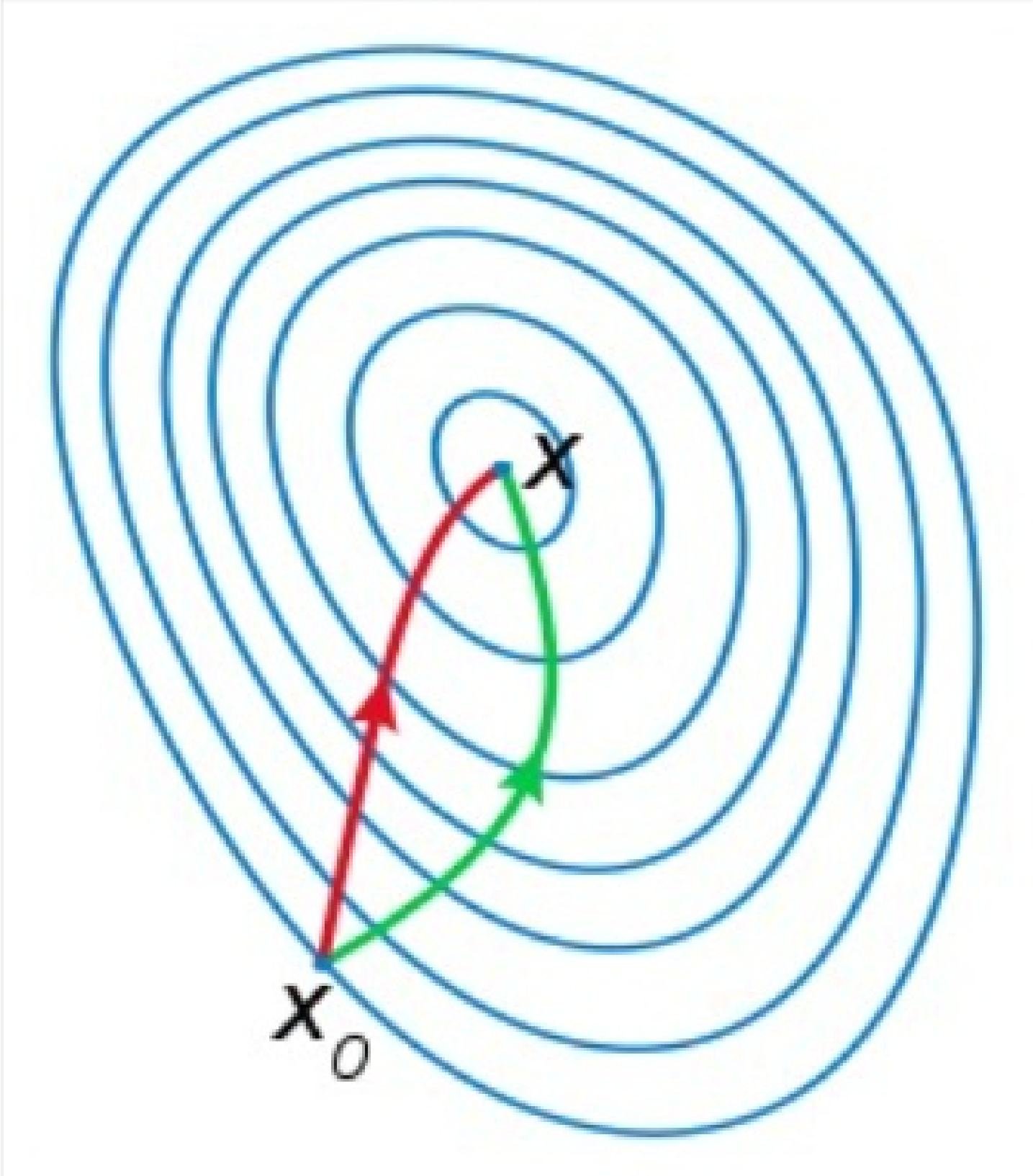
知乎 首发于
Eureka机器学习读书笔记



梯度下降法、牛顿法和拟牛顿法

Eureka

关注他

365 人赞同了该文章

知乎 首发于
Eureka机器学习读书笔记

本文结构如下

1： 梯度下降法

1.1 梯度

知乎 首发于 Eureka机器学习读书笔记

$\Delta x \to 0 \ \Delta x \qquad \Delta x \to 0 \qquad \Delta x$

当函数是多元时，倒数就变成了偏导数： $f_x(x,y)$ 表示当 y 不变时， $f(x,y)$ 沿着 x 轴的变化变化率； $f_y(x,y)$ 表示当 x 不变时， $f(x,y)$ 沿着 y 轴的变化变化率。

但是多元函数是一个平面，方向有很多， x 轴 y 轴只是其中两个方向而已，假如我们需要其他方向的变化率怎么办呢？因此方向导数就有用了，顾名思义，方向导数可以表示任意方向的倒数。

假如二次函数 $f(x,y)$ ，方向 $u = cos\theta\, i + sin\theta\, j$ （为单位向量）的方向导数公式如下：
$\lim_{t \to 0} \frac{f(x + t\, cos\theta, y + t\, sin\theta) - f(x)}{t}$ ，记为 $D_u f$ 。其中
$D_u f = f_x(x,y)cos\theta + f_y(x,y)sin\theta = [\, f_x(x,y) \quad f_y(x,y)\,] \begin{bmatrix} cos\theta \\ sin\theta \end{bmatrix}$ 。我们记为：
$D_u f = \mathbf{A} \times \mathbf{I} = |\mathbf{A}||\mathbf{I}|cos\alpha$ ，其中 α 是两向量的夹角。我们可以知道，当 α 为0时，方向导数 $D_u f$ 达到最大，此时的方向导数即为梯度。

更多的关于梯度的知识点可以看里面的回答：如何直观形象的理解方向导数与梯度以及它们之间的关系？

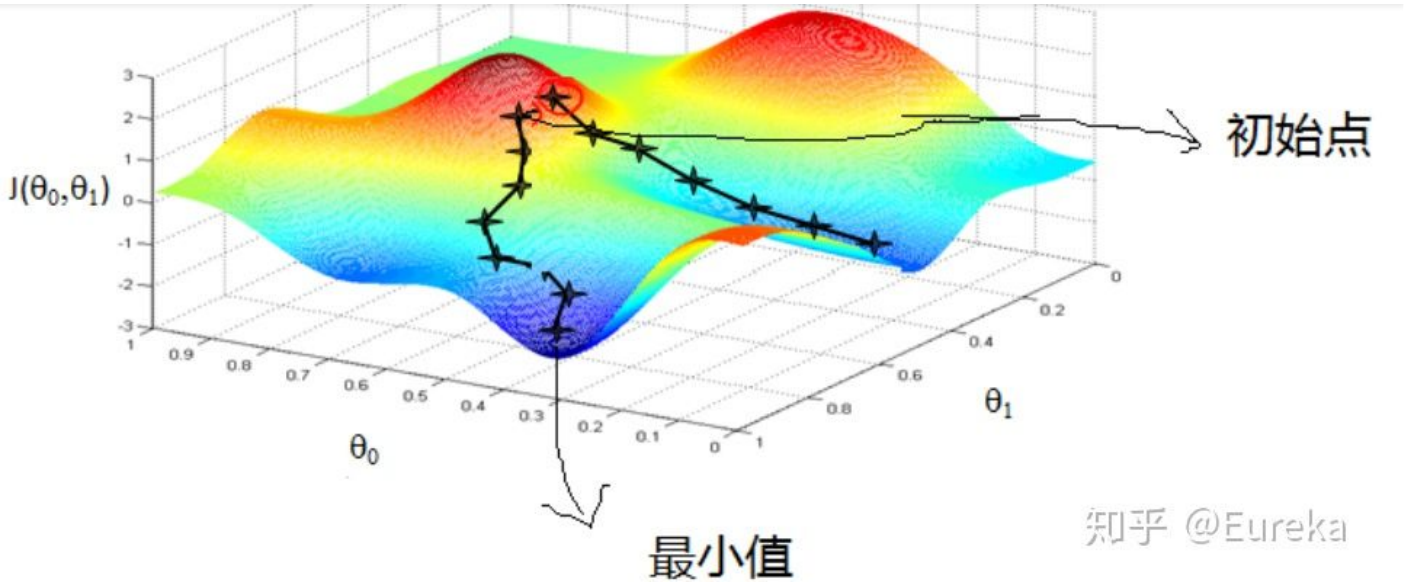
从几何意义上来说，梯度向量就是函数变化增加最快的地方。具体来说，对于函数 $f(x,y)$,在点 (x_0, y_0) ，沿着梯度向量的方向就是 $\begin{bmatrix} \frac{\partial f}{\partial x_0} \\ \frac{\partial f}{\partial y_0} \end{bmatrix}$ 的方向是 $f(x,y)$ 增加最快的地方（还记得梯度怎么来的吗？方向导数的最大值，粗暴点，就是导数最大值）。或者说，沿着梯度向量的方向，更加容易找到函数的最大值。反过来说，沿着梯度向量相反的方向（去负号），则就是更加容易找到函数的最小值。

1.2 梯度下降的直观解释

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（局部最小值）。

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。

知乎 首发于 Eureka机器学习读书笔记



1.3 梯度下降算法法

梯度下降法(Gradient descent)或最速下降法(steepest descent)是求解无约束最优化问题的一种常用的、实现简单的方法。

假设 $f(x)$ 是 R^n 上具有一阶连续偏导数的函数。求解

$$\min_{x \in R^n} f(x)$$

无约束最优化问题。 f^* 表示目标函数 $f(x)$ 的极小点。

梯度下降法是一种迭代算法。选取适当的初值 $x^{(0)}$ ，不断迭代，更新 x 的值，进行目标函数的极小化，直到收敛。由于负梯度方向是使函数下降最快的的方向，在迭代的每一步，以负梯度方向更新 x 的值，从而达到减少函数值的目的。

第 $k+1$ 次迭代值

$$x^{(k+1)} \leftarrow x^{(k)} + \lambda_k p_k$$

其中， p_k 是搜索方向，取负梯度方向 $p_k = -\nabla f\left(x^{(k)}\right)$ ， λ_k 是步长，由一维搜索确定，即 λ_k 使得

$$f\left(x^{(k)} + \lambda_k p_k\right) = \min_{\lambda \geq 0} f\left(x^{(k)} + \lambda p_k\right)$$

> **梯度下降算法：**
> 输入：目标函数 $f(x)$ ，梯度函数 $g\left(x^{(k)}\right) = \nabla f\left(x^{(k)}\right)$ ，计算精度 ε
> 输出： $f(x)$ 的极小点 x^*

3. 计算梯度 $g_k = g\left(x^{(k)}\right)$ ，当 $\|g_k\| < \varepsilon$ 时，停止迭代，令 $x^* = x^{(k)}$ ；否则，令 $p_k = -g\left(x^{(k)}\right)$ ，求 λ_k ，使

$$f\left(x^{(k)} + \lambda_k p_k\right) = \min_{\lambda \geq 0} f\left(x^{(k)} + \lambda p_k\right)$$

4. 置 $x^{(k+1)} = x^{(k)} + \lambda_k p_k$ ，计算 $f\left(x^{\left(k+1\right)}\right)$

当 $\|f\left(x^{\left(k+1\right)}\right) - f\left(x^{(k)}\right)\| < \varepsilon$ 或 $\|x^{\left(k+1\right)} - x^{(k)}\| < \varepsilon$ 时，停止迭代，令 $x^* = x^{k+1}$

5. 否则，置 $k = k + 1$ ，转3.

1.4 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。**步长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。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。

1.5 梯度下降法大家族（BGD，SGD，MBGD）

1.5.1 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。

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

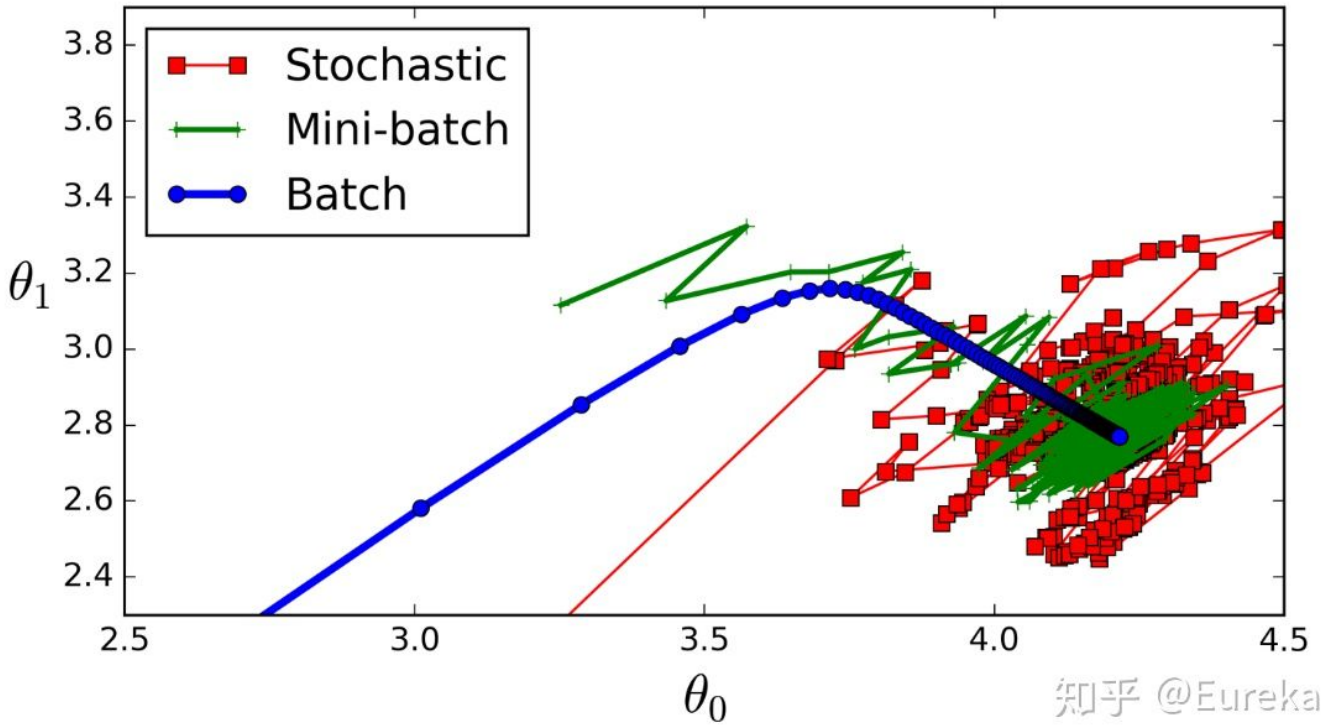
随机梯度下降法，其实和批量梯度下降法原理类似，区别在与求梯度时没有用所有的样本的数据，而是仅仅选取一个样本来求梯度。

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

知乎 首发于 Eureka机器学习读书笔记

> **1.5.3 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 $x = 16, 32, 64...$ ，当然根据样本的数据，可以调整这个 x 的值。

上述三种方法得到局部最优解的过程如下：



1.6 批量梯度下降法python实现

```
def train(X, y, W, B, alpha, max_iters):
    '''
    使用了所有的样本进行梯度下降
    X: 训练集,
    y: 标签,
    W: 权重向量,
    B: bias,
    alpha: 学习率,
    max_iters: 最大迭代次数.
    '''
    dW = 0 # 权重梯度收集器
    dB = 0 # Bias梯度的收集器
    m = X.shape[0] # 样本数
```

```
        dB = 0
        for j in range(m):
            # 1. 迭代所有的样本
            # 2. 计算权重和bias的梯度保存在w_grad和b_grad,
            # 3. 通过增加w_grad和b_grad来更新dW和dB
            W = W - alpha * (dW / m) # 更新权重
            B = B - alpha * (dB / m) # 更新bias

    return W, B
```

1.7 梯度下降法和最小二乘法

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。

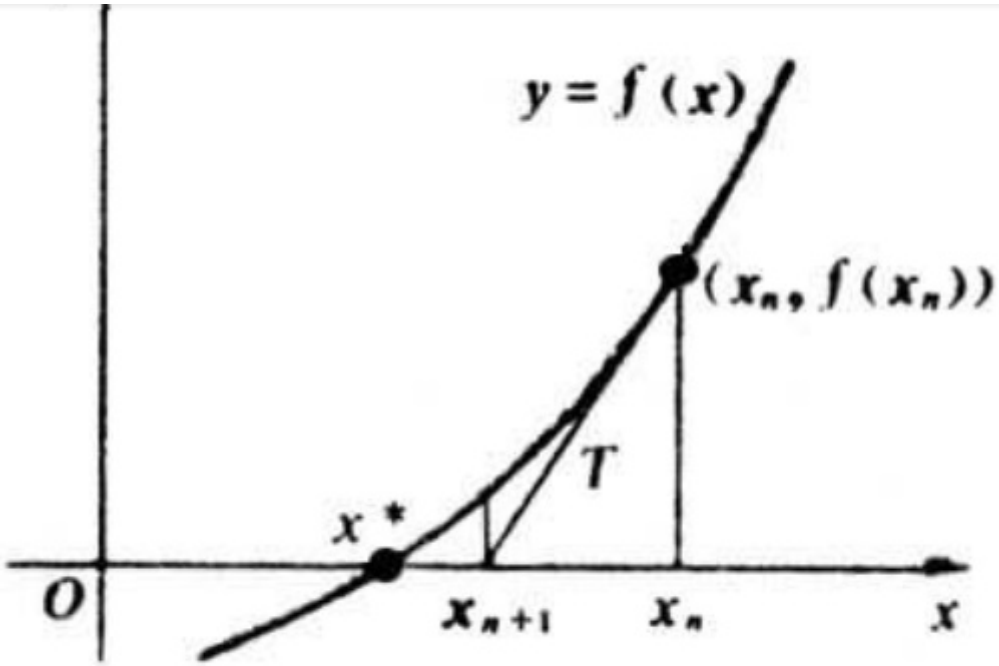
2： 牛顿法

一般来说，牛顿法主要应用在两个方面，1：求方程的根；2：最优化。

2.1 求解方程

并不是所有的方程都有求根公式，或者求根公式很复杂，导致求解困难。利用牛顿法，可以迭代求解。

原理是利用泰勒公式，在 x_0 处展开，且展开到一阶，即 $f(x)=f(x_0)+f'(x_0)(x-x_0)$ 。求解方程 $f(x)=0$ ，等价于 $f(x_0)+f'(x_0)(x-x_0)=0$ ，求解 $x=x_1=x_0-f(x_0)/f'(x_0)$ 。因为这是利用泰勒公式的一阶展开， $f(x)=f(x_0)+f'(x_0)(x-x_0)$ 处并不是完全相等，而是近似相等，这里求得的 x_1 并不能让 $f(x)=0$ ，只能说 $f(x_1)$ 的值比 $f(x_0)$ 的值更接近于0，于是迭代的想法就很自然了，可以进而推出 $x_{n+1}=x_n-f(x_n)/f'(x_n)$ ，通过迭代，这个式子必然在 $f(x^*)=0$ 的时候收敛。整个过程如下：



牛顿法求实根图示

2.2 最优化

无约束最优化问题

$$\min_{x \in R^n} f(x)$$

其中 x^* 为目标函数的极小点。

设 $f(x)$ 具有二阶连续偏导数，若第 k 次迭代值为 $x^{(k)}$ ，则可将 $f(x)$ 在 $x^{(k)}$ 附近进行二阶泰勒展开

$$f(x) = f\left(x^{(k)}\right) + g_k^T\left(x - x^{(k)}\right) + \frac{1}{2}\left(x - x^{(k)}\right)^T H\left(x^{(k)}\right)\left(x - x^{(x)}\right)$$

其中， $g_k = g\left(x^{(k)}\right) = \nabla f\left(x^{(k)}\right)$ 是 $f(x)$ 的梯度向量在点 $x^{(k)}$ 的值， $H\left(x^{(k)}\right)$ 是 $f(x)$ 的海赛矩阵

$$H(x) = \left[\frac{\partial^2 f}{\partial x_i \partial x_j}\right]_{n \times n}$$

在点 $x^{(k)}$ 的值。

正定矩阵时，函数 $f(x)$ 的极值为极小值。

我们为了得到一阶导数为0的点，可以用到2.1里的求解方程方法。根据二阶泰勒展开，对 $\nabla f(x)$ 在 $x^{(k)}$ 进行展开得（也可以对上述泰勒公式再进行求导）

$$\nabla f(x) = g_k + H_k\left(x - x^{(k)}\right)$$

其中， $H_k = H\left(x^{(k)}\right)$ ，则

$$g_k + H_k\left(x^{(k+1)} - x^{(k)}\right) = 0$$
$$x^{(k+1)} = x^{(k)} - H_k^{-1} g_k$$

我们令

$$H_k p_k = -g_k$$

则得到迭代公式

$$x^{(k+1)} = x^{(k)} + p_k$$

最终可在 $\nabla f(x^*) = 0$ 收敛。

> **牛顿法**:
> 输入：目标函数 $f(x)$ ，梯度 $g(x) = \nabla f(x)$ ，海赛矩阵 $H(x)$ ，精度要求 ε
> 输出： $f(x)$ 的极小点 x^*
> 1. 取初始点 $x^{(0)}$ ，置 $k = 0$
> 2. 计算 $g_k = g\left(x^{(k)}\right)$
> 3. 若 $\|g_k\| < \varepsilon$ 则停止计算，得近似解 $x^* = x^{(k)}$
> 4. 计算 $H_k = H\left(x^{(k)}\right)$ ，并求 p_k
>
> $$H_k p_k = -g_k$$
>
> 5. 置 $x^{(k+1)} = x^{(k)} + p_k$
> 6. 置 $k = k + 1$ ，转2.

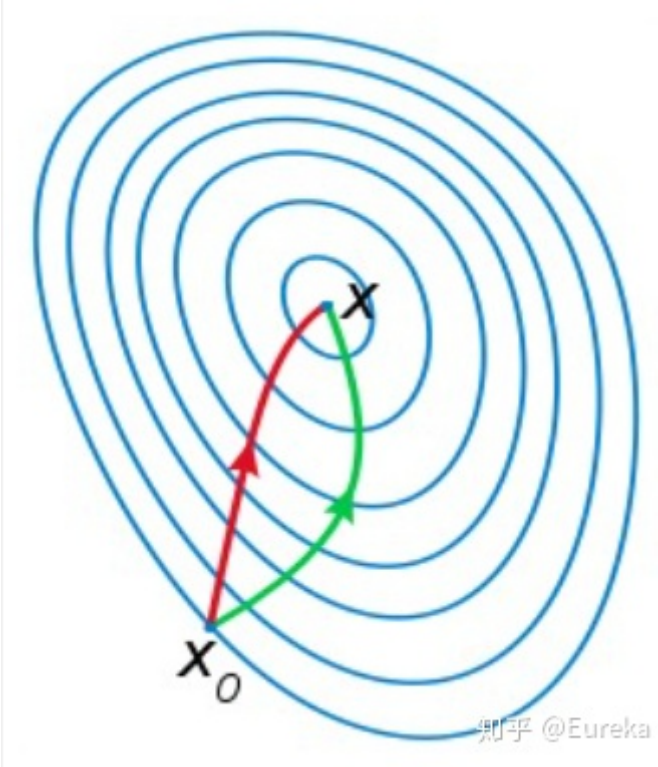
2.3： 牛顿法与梯度下降法

梯度下降法和牛顿法相比，两者都是迭代求解，不过梯度下降法是梯度求解，而牛顿法是用二阶的海森矩阵的逆矩阵求解。相对而言，使用牛顿法收敛更快（迭代更少次数）。但是每次迭代的时间比梯度下降法长。

知乎 首发于 Eureka机器学习读书笔记

牛顿法： $x^{(k+1)} = x^{(k)} - \lambda (H^{(k)})^{-1} \nabla f(x^k)$

如下图是一个最小化一个目标方程的例子，红色曲线是利用牛顿法迭代求解，绿色曲线是利用梯度下降法求解。



至于为什么牛顿法收敛更快，通俗来说梯度下降法每次只从你当前所处位置选一个坡度最大的方向走一步，牛顿法在选择方向时，不仅会考虑坡度是否够大，还会考虑你走了一步之后，坡度是否会变得更大。所以，可以说牛顿法比梯度下降法看得更远一点，能更快地走到最底部。更多的可见：最优化问题中，牛顿法为什么比梯度下降法求解需要的迭代次数更少？

3： 拟牛顿法的思路

在牛顿法的迭代中，需要计算海森矩阵的逆矩阵 H^{-1} ，这一计算比较复杂，考虑用一个 n 阶矩阵 $G_k = G(x^{(k)})$ 来近似代替 $H_k^{-1} = H^{-1}(x^{(k)})$ 。这就是拟牛顿法的基本想法。

要找到近似的替代矩阵，必定要和 H_k 有类似的性质。先看下牛顿法迭代中海森矩阵 H_k 满足的条件。首先 H_k 满足以下关系：

$$\nabla f(x) = g_k + H_k\left(x - x^{(k)}\right)$$

得

$$g_{k-1} - g_k = H_k\left(x^{(k-1)} - x^{(k)}\right)$$

记 $y_k = g_k - g_{k-1}$， $\delta_k = x^{(k)} - x^{(k-1)}$ ，则

$$y_{k-1} = H_k\delta_{k-1}$$
$$H_k^{-1}y_{k-1} = \delta_{k-1}$$

称为拟牛顿条件(Secant equation)。

注：在李航老师《统计机器学习》附录B中，取的是 $x = x^{(k+1)}$ ，得到的拟牛顿条件是 $H_k^{-1}y_k = \delta_k$ ，但是用 $x = x^{(k-1)}$ 的到的拟牛顿条件和下面推导吻合，但还没找到原因，还望高手指教。

其次，如果 H_k 是正定的（ H_k^{-1} 也是正定的），那么保证牛顿法的搜索方向 p_k 是下降方向。这是因为搜索方向是 $p_k = -H_k^{-1}g_k$ ，
由

$$x^{(k+1)} = x^{(k)} - H_k^{-1}g_k$$

有

$$x = x^{(k)} - \lambda H_k^{-1}g_k = x^{(k)} + \lambda p_k$$

则 $f(x)$ 在 $x^{(k)}$ 的泰勒展开可近似为

$$f(x) = f\left(x^{(k)}\right) - \lambda g_k^T H_k^{-1}g_k$$

由于 H_k^{-1} 正定，故 $g_k^T H_k^{-1}g_k > 0$ 。当 λ 为一个充分小的正数时，有 $f(x) < f\left(x^{(x)}\right)$ ，即搜索方向 p_k 是下降方向。

因此拟牛顿法将 G_k 作为 H_k^{-1} 近似。要求 G_k 满足同样的条件。首先，每次迭代矩阵 G_k 是正定的。同时， G_k 满足下面的拟牛顿条件：

按照拟牛顿条件，在每次迭代中可以选择更新矩阵 G_{k+1} ：

$$G_{k+1} = G_k + \Delta G_k$$

这种选择有一定的灵活性，因此有很多具体的实现方法，下面介绍Broyden类拟牛顿法。

4： DFP(Davidon-Fletcher-Powell)算法(DFP algorithm)

DFP算法中选择 G_k 作为 H_k^{-1} 的近似，假设每一步迭代中矩阵 G_{k+1} 是由 G_k 加上两个附加项构成，即

$$G_{k+1} = G_k + P_k + Q_k$$

其中， P_k 与 Q_k 是待定矩阵。则

$$G_{k+1}y_k = G_k y_k + P_k y_k + Q_k y_k$$

为使 G_{k+1} 满足拟牛顿条件，可使 P_k 与 Q_k 满足

$$\begin{aligned} P_k y_k &= \delta_k \\ Q_k y_k &= -G_k y_k \end{aligned}$$

可取

$$\begin{aligned} P_k &= \frac{\delta_k \delta_k^T}{\delta_k^T y_k} \\ Q_k &= -\frac{G_k y_k y_k^T G_k}{y_k^T G_k y_k} \end{aligned}$$

可得矩阵 G_{k+1} 的迭代公式

$$G_{k+1} = G_k + \frac{\delta_k \delta_k^T}{\delta_k^T y_k} - \frac{G_k y_k y_k^T G_k}{y_k^T G_k y_k}$$

可以证明，如果初始矩阵 G_0 是正定的，则迭代过程中的每个矩阵 G_k 都是正定的。

知乎 首发于 Eureka机器学习读书笔记

> 输出： $f(x)$ 的极小点 x^*
> 1. 取初始点 $x^{(0)}$ ，取 G_0 为正定矩阵，置 $k=0$
> 2. 计算 $g_k=g\left(x^{(k)}\right)$ ， 若 $\|g_k\|<\varepsilon$ 则停止计算，得近似解 $x^*=x^{(k)}$ ；否则，转3.
> 3. 置 $p_k=-G_k g_k$
> 4. 一维搜索，求 λ_k 使
>
> $$f\left(x^{(k)}+\lambda_k p_k\right)=\min_{\lambda\geq 0} f\left(x^{(k)}+\lambda p_k\right)$$
>
> 5. 置 $x^{(k+1)}=x^{(k)}+\lambda p_k$
> 6. 计算 $g_{k+1}=g\left(x^{(k+1)}\right)$ ，若 $\|g_{k+1}\|<\varepsilon$ 则停止计算，近似解 $x^*=x^{(k+1)}$ ；否则，计算
>
> $$G_{k+1}=G_k+\frac{\delta_k\delta_k^T}{\delta_k^T y_k}-\frac{G_k y_k y_k^T G_k}{y_k^T G_k y_k}$$
>
> 7. 置 $k=k+1$ ，转3.

5： BFGS(Broyden-Fletcher-Goldfard-Shano)算法(BFGS algorithm)

DFP是用 G_k 逼近海赛矩阵的逆矩阵 H_k^{-1} ，而BFGS算法中选择 B_k 逼近海赛矩阵 H_k ，相应的拟牛顿条件

$$B_{k+1}\delta_k=y_k$$

假设每一步迭代中矩阵 B_{k+1} 是由 B_k 加上两个附加项构成，即

$$B_{k+1}=B_k+P_k+Q_k$$

其中， P_k 与 Q_k 是待定矩阵。则

$$B_{k+1}y_k=B_k y_k+P_k y_k+Q_k y_k$$

为使 B_{k+1} 满足拟牛顿条件，可使 P_k 与 Q_k 满足

$$P_k\delta_k=y_k$$
$$Q_k\delta_k=-B_k y_k\delta_k$$

知乎 首发于 Eureka机器学习读书笔记

$$P_k = \frac{y_k y_k^T}{y_k^T \delta_k}$$

$$Q_k = -\frac{B_k \delta_k \delta_k^T B_k}{\delta_k^T B_k \delta_k}$$

可得矩阵 B_{k+1} 的迭代公式

$$B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T \delta_k} - \frac{B_k \delta_k \delta_k^T B_k}{\delta_k^T B_k \delta_k}$$

可以证明，如果初始矩阵 B_0 是正定的，则迭代过程中的每个矩阵 B_k 都是正定的。

> **BFGS算法**:
> 输入：目标函数 $f(x)$ ，梯度 $g(x) = \nabla f(x)$ ，精度要求 ε
> 输出： $f(x)$ 的极小点 x^*
> 1. 取初始点 $x^{(0)}$ ，取 B_0 为正定矩阵，置 $k = 0$
> 2. 计算 $g_k = g\left(x^{(k)}\right)$ ，若 $\|g_k\| < \varepsilon$ 则停止计算，得近似解 $x^* = x^{(k)}$ ；否则，转3.
> 3. 由 $B_k p_k = -g_k$ 求出 p_k
> 4. 一维搜索，求 λ_k 使
>
> $$f\left(x^{(k)} + \lambda_k p_k\right) = \min_{\lambda \geq 0} f\left(x^{(k)} + \lambda p_k\right)$$
>
> 5. 置 $x^{(k+1)} = x^{(k)} + \lambda p_k$
> 6. 计算 $g_{k+1} = g\left(x^{(k+1)}\right)$ ，若 $\|g_{k+1}\| < \varepsilon$ ，则停止计算，近似解 $x^* = x^{(k+1)}$ ；否则，计算
>
> $$B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T \delta_k} - \frac{B_k \delta_k \delta_k^T B_k}{\delta_k^T B_k \delta_k}$$
>
> 7. 置 $k = k + 1$ ，转3.

6： Broyden类算法(Broyden's algorithm)

我们可以从BFGS算法矩阵 $B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T \delta_k} - \frac{B_k \delta_k \delta_k^T B_k}{\delta_k^T B_k \delta_k}$ 的迭代公式得到BFGS算法关于 G_k 的迭代公式。

$$G_k = B_k^{-1}, \quad G_{k+1} = B_{k+1}^{-1}$$

两次应用Sherman-Morrison公式，得

$$G_{k+1} = \left(I - \frac{\delta_k y_k^T}{\delta_k^T y_k}\right) G_k \left(I - \frac{\delta_k y_k^T}{\delta_k^T y_k}\right)^T + \frac{\delta_k \delta_k^T}{\delta_k^T y_k}$$

称为BFGS算法关于 G_k 的迭代公式。

其中Sherman-Morrison公式：假设 A 是 n 阶可逆矩阵， u, v 是 n 维向量，且 $A + uv^T$ 也是可逆矩阵，则：

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1} u v^T A^{-1}}{1 + v^T A^{-1} u}$$

令由DFP算法 G_k 的迭代公式得到的 G_{k+1} 记作 G^{DFP} ，由BFGS算法 G_k 的迭代公式得到的 G_{k+1} 记作 G^{BFGS} ，由于 G^{DFP} 和 G^{BFGS} 均满足拟牛顿条件，则两者的线性组合

$$G_{k+1} = \alpha G^{DFP} + (1 - \alpha) G^{BFGS}$$

也满足拟牛顿条件，而且是正定的。其中， $0 \leq \alpha \leq 1$ 。该类算法称为Broyden类算法。

7： 参考文献

李航：《统计学习方法》，清华大学出版社

Jacobian矩阵和Hessian矩阵

梯度下降（Gradient Descent）小结

hackernoon.com/gradient...

牛顿法与拟牛顿法学习笔记

编辑于 2018-06-18

知乎 首发于 **Eureka机器学习读书笔记**

赞赏

2 人已赞赏

机器学习 统计学习方法（书籍） 最优化

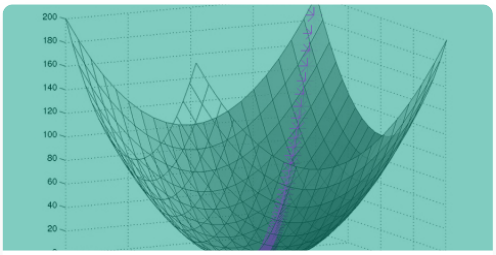
▲ 赞同 365 ▼ 14 条评论 分享 收藏 ···

文章被以下专栏收录

Eureka机器学习读书笔记
机器学习笔记

关注专栏

推荐阅读



一文看懂各种神经网络优化算法：从梯度下降到Adam方法

量子学园 发表于量子位

牛顿法和拟牛顿法——（书中附录B）

牛顿法(Newton method)和拟牛顿法(quasi-Newton method)也是求解无约束最优化问题的常用方法，具有收敛速度快的优点。牛顿法是迭代算法，每一步需要求解目标函数的海赛矩阵的逆矩阵，计算比...

Arlen 发表于深入理解 ...

一个框架 SGD/A

Adam那 念不忘 (算法 机器 们每天的 据），架 六味真火

Juliuszh

14 条评论

⇌ 切换为时间排序

写下你的评论...

林中象 2018-06-18

quasi newton的BFGS method的想法，主要是：在最后两次迭代中，f的梯度与近似函数m 的梯度相同（m是将f在当前迭代点处进行一阶taylor展开得到的） 导出secant equation

知乎 首发于 **Eureka机器学习读书笔记**

zeee 2018-09-12

好文

1

胡满超 2018-10-16

总结的很好

赞

king 2018-11-30

5的公式错了吧？Bk+1δk吧

赞

还有好多要学 2018-12-11

已赞赏 感谢

赞

李磊 2019-07-15

尝试回答注的问题：按我理解，我们统一此刻的x为x{k}，那么根据步长和搜索方向的乘积得到下一时刻即k+1时刻的x_{k+1}，在求解下降方向p_{k}和λ时用的是目标函数值极小化进行求解，此时带入目标函数的是：x_{k+1}，所以才能展开为x_{k}+λ*p_{k}，才能进行后续的计算，所以此处的牛顿法或者拟牛顿条件的海森阵H才能根据x_{k}进行计算，不好求才有了拟牛顿方法用G去代替；至于为什么取x_{k-1}可以，因为等号左右的负号抵消得到的形式与前面一样，差别在于此处不该是H_{k}，而应该是H_{k-1}。

3

半碗鱼 2019-07-23

请问最后一法的Gk是替换的海塞矩阵还是海塞矩阵的逆

赞

vincent 2019-08-11

2.3： 牛顿法与梯度下降法 中牛顿法的公式 写错了吧，没有 系数

1

深海181 2019-11-12

问下，为什么能直接带入在k+1处的导数等于零，那前面不等于零怎么办

赞

取嘛，这玩意儿还能够动态？谢谢指教

1

零零零 02-04

明白了，算法中的最优化lambda步长是理想状态的，但是无法实现，所以现实求解中才用固定lambda，靠验证集来选取一个合适的学习率，应该莫得问题把这种理解

赞

谁杀死了知更鸟 03-19

写得太好了，比较系统，而且一看就懂

赞

张莉苑 04-13

[赞同][赞同][赞同]

赞

似水如风 05-23

“拟牛顿条件(Secant equation)” 上面记 y, sigema 下标有问题，应该是k-1，而不是k

赞