搜索算法

陈 鑫

2023 年 3 月 21 日

目录

目录

搜索算法

搜索算法是利用计算机的高性能来有目的地穷举一个问题解空间的部分或所有的可能情况，从而求出问题的解的一种方法。该类算法通常具有如下的基本算法结构：从可行区域中任意初始点开始搜索，在每一个搜索点的局部可行区域中寻找一个能使目标函数值下降的方向并沿着该方向移动至下一可行点。按此方式循环迭代，直至无法继续移动为止。此时输出搜索结束时所在的可行点。由于从最终输出的可行点出发无法再找到一个能局部降低目标函数的下一个可行点，因而算法输出的解必定是一个局部最优解。

梯度下降算法

梯度下降（Gradient Descent）算法是解决优化问题的重要方法之一，在机器学习中应用十分广泛，不论是在线性回归还是 Logistic 回归中，其主要目的都是通过迭代找到目标函数的最小值。梯度下降算法的基本思想是：假定目标函数可微，算法从空间中任意给定的初始点开始进行指定轮次的搜索（或达到某个条件为止），在每一轮搜索中都计算目标函数在当前点的梯度，并沿着与梯度相反的方向，按照一定的步长移动到下一个可行点。

案例实战 4.1：运用梯度下降算法模拟下山过程，假设“山函数”为 $f(x) = cos^2(1/x)$。

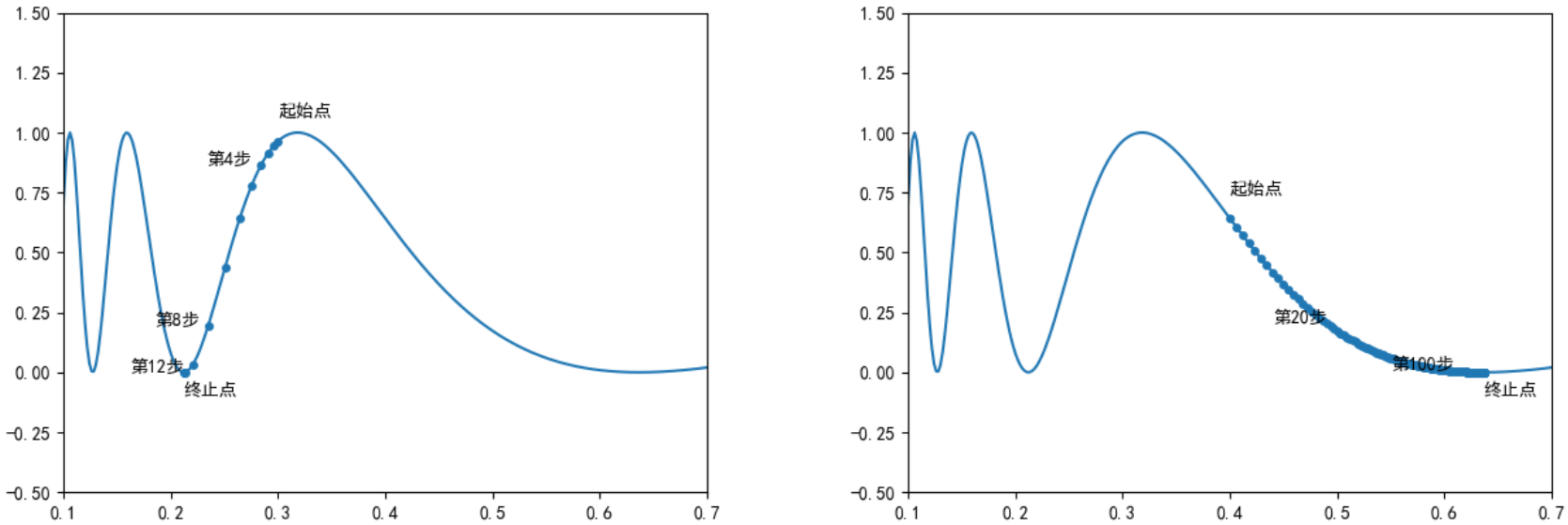

图 1: 模拟下山的梯度下降法

梯度下降算法的任务是一个无约束优化问题：

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

其中目标函数 f 是一个可微的 n 元实函数。该算法包含两个参数，一个是搜索的轮次 N，另一个是搜索的步长 $\eta(\eta > 0)$。初始时，设定 $\boldsymbol{x}$ 为全 $\mathbf{0}$ 向量，开始 N 轮循环，在每一轮循环中都沿着梯度 $\nabla f(\boldsymbol{x})$ 的反方向前进一小步，步长为 η：

$$\boldsymbol{x} \leftarrow \boldsymbol{x} - \eta\nabla f(\boldsymbol{x})$$

将 η 称为学习速率。

目录

梯度下降算法

Algorithm 1: 梯度下降算法

求函数 $f(\boldsymbol{x})$ 的极值点 $\boldsymbol{x}$

Input: $\boldsymbol{x} = 0$

Output: $\boldsymbol{x}$

for $i = 1, 2, \cdots, N$: **do**

$\quad \boldsymbol{x} \leftarrow \boldsymbol{x} - \eta \nabla f(\boldsymbol{x})$

return $\boldsymbol{x}$

用梯度下降算法求解线性回归问题时，设定目标函数为均方误差 $f(\boldsymbol{w}) = \frac{1}{m}||\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}||^2$。$f(\boldsymbol{w})$ 的梯度为

$$\nabla f(\boldsymbol{w}) = \frac{2}{m}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$$

梯度下降收敛到局部最小值的原理

以 $f(x)$ 为一元函数为例，分析一下梯度下降算法的收敛原理。对于任意一点 x_0，将函数 $f(x)$ 在 x_0 处按泰勒公式展开，有 $f(x)=f(x_0)+f'(x_0)(x-x_0)+\frac{1}{2!}f''(x_0)(x-x_0)^2+\cdots$ 取 $x=x_0-\eta\nabla f(x_0)$ 并将之代入上式，可得到 $f(x)=f(x_0)-\eta f'(x_0)^2+o(\eta)$，其中 $o(\eta)$ 是关于 η 的高阶无穷小量，当 η 足够小时，该项可以忽略。只要 $f'(x_0)\neq 0$，就有 $\eta f'(x_0)^2>0$，那么 $f(x)<f(x_0)$，这就说明使用迭代算法1，目标函数值能沿着梯度相反的方向局部下降，这就是梯度下降算法能收敛到局部最优值的原理。

案例实战 4.2：使用梯度下降法对加州房价问题进行预测。

用梯度下降算法求解线性回归时，设定目标目标函数为均方误差 $F(\boldsymbol{w}) = ||\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}||^2/m$。$F(\boldsymbol{w})$ 的梯度为 $\nabla F(\boldsymbol{w}) = \frac{2}{m}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$

在第 3 章，我们已经对加州房价使用了正规方程算法进行了预测，但由于正规方程算法需要计算特征方阵 $(X^TX)^{-1}$，当数据的特征个数很多的时候，求解正规方程的计算量非常大，从而导致正规方程求解算法效率低下，甚至不可行，因此，对于特征个数较多的线性回归问题，梯度下降算法是一个更为实用的求解方法。

详见书中代码 linear_regression_gd.py

目录

随机梯度下降算法

随机梯度下降算法是梯度下降算法的一种改进算法，每次只随机抽取一个样本进行梯度计算迭代，训练速度可以提高 m 倍（m 是数据样本总数），对于准确度来说，随机梯度下降法由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，不能很快地收敛到局部最优解。但当数据样本数量 m 很大时，随机选取的一条训练数据的梯度的数学期望等于整体数据的梯度，所以，可以认为随机梯度下降也能收敛到局部最优解，这就是随机梯度下降算法的核心思想。

随机梯度下降算法

随机梯度下降算法的输入是一组训练数据 S，参数是搜索步数 N 以及学习率 η_0 和 η_1。算法循环执行 N 轮搜索。在每一轮的循环 t 中，随机选取一条训练数据 $(\boldsymbol{x}^{(i)}, y^{(i)})$，并计算当前模型在该样本上的经验损失的梯度，然后算法沿着梯度的反方向调整 w 的值，步长设置为 $\eta_t = \frac{\eta_0}{\eta_1 + t}$，步长随着搜索轮次 t 的增加而减少。这样，随着算法搜索越接近最优解，搜索的步长 η_t 随着 t 的增加而变小，以确保不会跳过最优解。算法的输出是所有搜索点的平均值，以增强结果的稳定性。

Algorithm 2: 随机梯度下降算法

$w = 0, w_s = 0$

for $i = 1, 2, \cdots, N$: **do**

从训练集 S 中随机选取一条训练数据 $(\boldsymbol{x}^{(i)}, y^{(i)})$

$\eta_t = \frac{\eta_0}{\eta_1 + t}$

$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta_t \nabla \ell(h_{\boldsymbol{w}}(\boldsymbol{x}^{(i)}), y^{(i)})$

$\boldsymbol{w}_s \leftarrow \boldsymbol{w}_s + \boldsymbol{w}$

return $\bar{\boldsymbol{w}} = \frac{\boldsymbol{w}_s}{N}$

实例和算法对比

案例实战 4.3：加州房价预测问题的线性回归之随机梯度下降算法，代码详见 linear_regression_sgd.py 和 california_house_sgd.py

梯度下降算法与随机梯度下降算法的效果对比 gd_vs_sgd.py

目录

小批量梯度下降算法

小批量梯度下降算法（Mini-Batch Gradient Descent）对梯度下降和随机梯度下降算法进行折中考虑，在每次梯度下降的过程中，只选取数据的一部分 B 条样本数据进行计算梯度。这种方法减少了参数更新时的变化，能够更加稳定地收敛。同时，也能利用高度优化的矩阵，进行高效的梯度计算。在数据量较大的项目中，可以明显地减少梯度计算的时间。

案例实战代码参考 linear_regression_mbgd.py 和 california_house_mbgd.py

次梯度下降算法

（定义）设 $f:\mathbb{R}^n \to \mathbb{R}$ 为一个 n 元函数。如果 $\boldsymbol{x},\boldsymbol{y} \in \mathbb{R}^n$ 满足如下性质：$f(\boldsymbol{z}) \geq f(\boldsymbol{x}) + \boldsymbol{y}^T(\boldsymbol{z}-\boldsymbol{x})$，$\forall \boldsymbol{z} \in \mathbb{R}^n$ 则称 $\boldsymbol{y}$ 是 f 在 $\boldsymbol{x}$ 处的一个次梯度。称集合 $\partial f(\boldsymbol{x}) = \{\boldsymbol{y} \in \mathbb{R}^n | \boldsymbol{y}$ 为 f 在 $\boldsymbol{x}$ 处的次梯度$\}$ 为 f 在 $\boldsymbol{x}$ 处的次梯度集。

以 sklearn 工具库中的 LASSO 回归为例，应用次梯度方法，对加州房价进行预测。具体代码参考 california_house_lasso.py 文件。

目录

牛顿迭代法（Newton's method）

牛顿迭代法（Newton's method）是一种在实数域和复数域上近似求解方程的方法。因其收敛速度具有平方收敛，迭代步数较少，在许多机器学习算法的优化部分都可以用该迭代法来实现，缺点是需要计算目标函数的海森矩阵，当特征的个数较大时，比较耗时。

Definition

函数 $g(x)$ 的一个不动点是指一个实数 P，满足 $P = g(P)$。

Definition

迭代 $p_{n+1} = g(p_n)$，其中 $n = 0, 1, \ldots$，称为不动点迭代

Theorem

设 g 是一个连续函数，且 $\{p_n\}_{n=0}^{\infty}$ 是由不动点迭代生成的序列。如果 $\lim_{n\to\infty} p_n = P$，则 P 是 $g(x)$ 的不动点。

Theorem

（牛顿-拉夫森定理）设 $f \in C^2[a,b]$，且存在数 $p \in [a,b]$，满足 $f(p) = 0$。如果 $f'(p) \neq 0$，则存在一个数 $\delta > 0$，对任意初始近似值 $p_0 \in [p-\delta, p+\delta]$，使得由如下迭代定义的序列 $\{p_k\}_{k=0}^{\infty}$ 收敛到 p：

$$p_k = g(p_{k-1}) = p_{k-1} - \frac{f(p_{k-1})}{f'(p_{k-1})}, \quad k = 1, 2, \cdots \tag{1}$$

定义函数

$$g(x) = x - \frac{f(x)}{f'(x)}$$

并将之称为牛顿-拉夫森迭代函数。由于 $f(p) = 0$，显然 $g(p) = p$。这样通过寻找 $g(x)$ 的不动点，可以实现寻找方程 $f(x) = 0$ 的根的牛顿-拉夫森迭代。

由牛顿-拉夫森定理，我们可以得到求解函数 $f(x)$ 根的牛顿迭代算法。

Algorithm 3: $f(x)$ 函数 0 点的牛顿迭代算法

$x = 0$

while $|f(x)| > \epsilon$ **: do**

$\quad x \leftarrow x - \frac{f(x)}{f'(x)}$

Return x

案例实战 4.4：使用牛顿迭代法求解 $f(x) = (x-1)^2$ 的 0 点。具体代码见 newton_find_root.py

牛顿迭代法的一个有意思的应用是求解任意大于 0 常数 A 的 n 次方根。

求 100 的 3 次方根，即求 $f(x) = 100–x^3$ 的根。可将其转化成牛顿迭代法求解，参考代码 cube_root.py

在机器学习中，很多的优化问题归结为求函数的极值问题。对于一元函数 $f(x)$ 来说，可将求极值问题 $\min_{x\in\mathbb{R}} f(x)$ 转化为求该函数驻点，即导函数 $f'(x)$ 的 0 点，从而可以利用牛顿迭代算法。

Algorithm 4: 一元函数 $f(x)$ 优化问题（极值）的牛顿迭代算法

$x = 0$

while $|f'(x)| > \epsilon$ **: do**

$$x \leftarrow x - \frac{f'(x)}{f''(x)}$$

Return x

利用算法4求 $f(x) = x^2 + 3x + 4$ 的极小值，参考代码 func_extreme_point.py

一元函数的导数在高维空间中对应于多元函数的梯度。相应的二阶导数对应于海森（Hessian）方阵。将一元函数的牛顿迭代算法推广到多元优化问题。

给定函数 $f: R^n \to R$ 为二阶可微的 n 元函数，算法的目标是计算 $\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x})$。

Algorithm 5: 多元函数优化问题的牛顿迭代算法

$\boldsymbol{x} = 0$

while $|\nabla f(\boldsymbol{x})| > \epsilon$ **:do**

$\quad \boldsymbol{x} \leftarrow \boldsymbol{x} - \nabla^2 f(\boldsymbol{x})^{-1} \nabla f(\boldsymbol{x})$

Return $\boldsymbol{x}$

案例实战 4.6：线性回归问题的牛顿迭代算法。

在具有 n 个样本特征的线性回归问题中，目标函数的均方误差为 $f(\boldsymbol{w}) = \frac{1}{m}||X\boldsymbol{w} - \boldsymbol{y}||^2$，其中 $\boldsymbol{w}$ 为待求参数，是一个 $n\times1$ 的向量，X 是样本，为一个 $m\times n$ 的矩阵，$\boldsymbol{y}$ 是标签，为一个 $m\times1$ 的向量，m 为样本数量。

初始化 $\boldsymbol{w} = \mathbf{0}$，则有

$$\nabla f(\mathbf{0}) = \frac{2}{m}X^T(X\boldsymbol{w} - \boldsymbol{y}) = \frac{2}{m}X^TX\boldsymbol{w} - \frac{2}{m}X^T\boldsymbol{y} \overset{w=0}{=} -\frac{2}{m}X^T\boldsymbol{y}$$

$$\nabla^2 f(\mathbf{0}) = \frac{2}{m}X^TX$$

按照算法中 $f(\boldsymbol{w})$ 的更新规则，可以得到 $f(\boldsymbol{w}) = \mathbf{0} - \nabla^2 f(\mathbf{0})^{-1}\nabla f(0) = (X^TX)^{-1}X^T\boldsymbol{y}$，这说明将 $\boldsymbol{w} = \mathbf{0}$ 设置为初值，则牛顿迭代法一步就收敛了，这个解刚好就是前面我们用正规方程的解。所以，正规方程算法可认为是牛顿迭代算法在线性回归中的具体应用，程序代码请参考 linear_regression.py。

目录

坐标下降法（Coordinate Descent）是一个简单但却高效的非梯度优化算法。与梯度优化算法沿着梯度最速下降的方向寻找函数最小值不同，坐标下降法依次沿着坐标轴的方向最小化目标函数值。

坐标下降的核心思想是将一个复杂的优化问题分解为一系列简单的优化问题以进行求解。对高维的损失函数 $f(x_1, \cdots, x_n)$，求最小值有时并不是一件容易的事情，而坐标下降法就是迭代地通过将大多数自变量 $(x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n)$ 固定（即看作已知常量），而只针对剩余的自变量 x_i 求极值的过程。这样，一个高维的优化问题就被分解成了多个一维的优化问题，从而大大降低了问题的复杂性。

$f(x,y) = 5x^2 - 6xy + 5y^2$ 坐标下降求极值

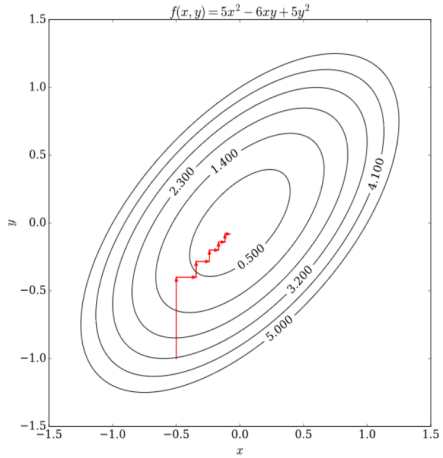

图 2: $f(x,y) = 5x^2 - 6xy + 5y^2$

假设我们有目标函数 $f(x,y) = 5x^2 - 6xy + 5y^2$，其等高线如图2所示，求 (x,y) 以使得目标函数在该点的值最小。

由图中红色式子标示的起始点 $(-0.5, -1.0)$，此时 $f = 3.25$。先固定 x，将 f 看称时关于 y 的一元二次方程，并求当 f 最小时 y 的值。由 $f(x,y) = 5x^2 - 6xy + 5y^2$，当 $x = -0.5$ 时，
$f(y|x = -0.5) = 5(-0.5)^2 - 6(-0.5)y + 5y^2 = 5y^2 + 3y + 1.25$，在这个条件下，函数 $f(x,y)$ 的极小值为当 $f'(y|x = -0.5) = 10y + 3 = 0$ 时取得，即当 $y = -0.3$ 时取得极小值，此时的自变量为 $(-0.5, -0.3)$ 取得极小值 $f = 0.8$。在此基础上，将新的到的 y 值固定（$y = -0.3$），将 f 看成关于 x 的一元二次方程，并求当 $f(x, -0.3)$ 最小时的 x 值。
$f(x|y = -0.3) = 5x^2 + 1.8x + 0.45$，函数 $f(x,y)$ 的极小值为当 $f'(x|y = -0.3) = 10x + 1.8 = 0$ 时取得，即 $x = -0.18$ 时取得极小值，此时的自变量为 $(-0.18, -0.3)$ 取得极小值 $f = 0.288$。以此类推，经过多次迭代之后，自变量 (x,y) 的运动轨迹就如图2所示。

Sklearn 的随机梯度下降算法模块

案例实战 4.7 使用随机梯度下降算法 SDG 训练的支持向量机来绘制最大间隔可分离超平面 max_separable_surface_sdg_sklearn.py

案例实战 4.8 在鸢尾花数据集上使用 Sklearn 的多分类 SGD 分类器绘制分类边界图 iris_SGD_sklearn.py