LN8. 梯度下降

李钦宾

先进智能计算与系统团队

邮箱： *qinbin@hust.edu.cn*

2025 年 3 月

目录

目录

回顾

- 在关于逻辑回归一讲中，我们给出了模型中参数的表达式作为待求解的优化问题，这些优化问题没有闭式解。
- 具体来说，给定数据 $\{(x_i, y_i)\}_{i=1}^n$ 且有 $x_i \in \mathbb{R}^d$，$y_i \in \{+1, -1\}$，我们观察到

$$\hat{w}_{\text{MLE}} = \arg\min_{w \in \mathbb{R}^d, b \in \mathbb{R}} \sum_{i=1}^n \log(1 + e^{-y_i(w^T x_i + b)})$$

  且

$$\hat{w}_{\text{MAP}} = \arg\min_{w \in \mathbb{R}^d, b \in \mathbb{R}} \sum_{i=1}^n \log(1 + e^{-y_i(w^T x_i + b)}) + \lambda w^T w$$

- 本节将讨论求解这些问题的一般策略。我们将问题抽象为

$$\min_w \ell(w)$$

  其中 $\ell \colon \mathbb{R}^d \to \mathbb{R}$.

回顾

本节将讨论一些基本的算法策略
我们想最小化一个**凸**的**连续可微**的损失函数 $\ell(w)$。

- ℓ 是凸的。这使得所找到的任何局部最小值也是全局最小值，且有助于简化对牛顿法的讨论。
- ℓ 至少是三次连续可微的。我们将使用泰勒展开近似。这个假设极大地简化了讨论。
- 对 w 没有约束。增加对 w 的约束会增加讨论的复杂性，我们对此不展开讨论。

本节将讨论两种被广泛使用的“爬坡”算法，梯度下降法和牛顿法 $\min_w \ell(w)$。

什么是（局部）最小值

- 问题：求解 $\min_w \ell(w)$ 实际上意味着什么。
- 我们称 w^* 为 ℓ 的局部最小值，如果满足：

局部最小值:

存在 $\epsilon > 0$ 对于 $\{w \mid \|w - w^*\|_2 < \epsilon\}$ 满足 $\ell(w^*) \leq \ell(w)$.

- 我们之前假设 ℓ 是凸的，这意味着若找到这样一个 w^*，则对于所有 $w \in \mathbb{R}^d$ 均有 $\ell(w^*) \leq \ell(w)$。
- 也可以通过 $\ell(w^*) < \ell(w)$ 来定义一个严格的局部最小值。
- 注意，一些凸函数没有严格的局部最小值，例如常数函数 $\ell(w) = 1$ 是凸函数。
- 一些凸函数没有局部最小值，例如，对于任意非零向量 $c \in \mathbb{R}^d$，$c^T w$ 是凸的，但可以使其任意小。
- 一个点是局部最小点的关键必要条件是 ℓ 在 w^* 处的梯度为 0，即 $\nabla \ell(w^*) = 0$。
- 假设函数的梯度在 w^* 处为 0，该点为严格局部最小的充分条件为其海森矩阵 $\nabla^2 \ell(w^*)$ 是正定的。

目录

泰勒展开

- 虽然我们对 ℓ 做了一些假设，但它们实际上并没有使我们得到更多关于函数 ℓ 的信息。此外，考虑类似逻辑回归的例子（存在梯度消失现象），在全局范围内考虑函数 ℓ 并不是太容易。
- 因此我们常常会利用函数 ℓ 的局部信息。通过使用一阶和二阶泰勒展开来得到该点的局部信息。

泰勒展开

一阶泰勒展开

以 w 为中心的一阶泰勒展开可以写为:

$$\ell(w+s) \approx \ell(w) + s^T g(w),$$

其中 $g(w)$ 是 ℓ 在 w 处的梯度，即 $(g(w))_j = \frac{\partial \ell}{\partial w_j}(w)$，对于 $j = 1, \ldots, d$.

二阶泰勒展开

以 w 为中心的二阶泰勒展开可以写为:

$$\ell(w+s) \approx \ell(w) + s^T g(w) + \frac{1}{2} s^T H(w) s,$$

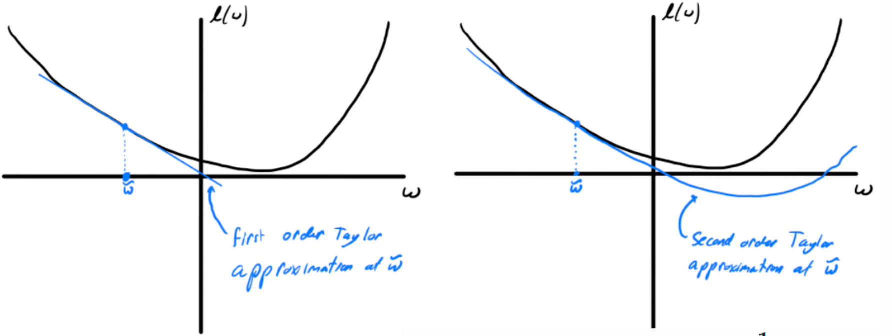
其中 $H(w)$ 是 ℓ 在 w 处的 Hessian 矩阵，即:

$$[H(w)]_{i,j} = \frac{\partial^2 \ell}{\partial w_i \partial w_j}(w),$$

对于 $j = 1, \ldots, d$.

一阶和二阶泰勒展开

- 这些对应 ℓ 的线性近似和二次近似。
- 如果 s 很小，那么这些近似是合理有效的 (一阶误差为 $\mathcal{O}(\|s\|_2^2)$，二阶误差为 $\mathcal{O}(\|s\|_2^3)$)。



$$\ell(w+p) \approx \ell(w) + g(w)^T p$$

$$\ell(w+p) \approx \ell(w) + g(w)^T p + \frac{1}{2} p^T H(w) p$$

目录

搜索方向法

损失函数

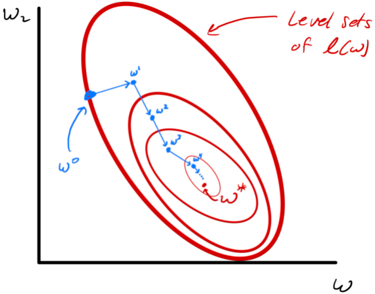
$$\min_{w} \ell(w),$$

- 核心思想是给定一个起始点 w^0，我们构造一个迭代序列 $w^1, w^2 \; \ldots,$ 目标是当 $k \to \infty$，有 $w^k \to w^*$。
- 在搜索方向法中，我们将考虑从 w^k 构造得到 w^{k+1}，将其写成 $w^{k+1} = w^k + s$，其中 s 是 w^k 更新得到 w^{k+1} 的梯度。

Input: initial guess w^0
k = 0;
While not converged:

1. Pick a step s
2. $w^{k+1} = w^k + s$ and $k = k + 1$
3. Check for convergence; if converged set $\hat{w} = w^k$

Return: $\hat{w}$



两个关键步骤

在上述算法中有两个不明确的步骤:

- **我们如何选择** s
- 如何确定何时收敛

我们将花大部分时间来解决前者，然后简要地介绍后者——稳健地保证收敛性是一个好的优化包应该做得很好的细节之一。

目录

梯度下降法

核心思想

考虑当前在这一点上，函数值下降最快的方向并朝该方向迈出一步。

考虑到展开点的线性逼近可以利用泰勒级数得到

$$\ell(w^k + s) = \ell(w^k) + s^T g(w^k)$$

那么下降最快的方向可以表示为 $s \propto -g(w^k)$。
我们在梯度下降中将 s 设为

$$s = -\alpha g(w^k)$$

其中设置步长 $\alpha > 0$。

正确性

总有一些足够小的 α

$$\ell(w^k - \alpha g(w^k)) < \ell(w^k).$$

为什么？

$$\ell(w^k + s) = \ell(w^k) + g(w^k)^T s$$

$$s = -\alpha g(w^k)$$

$$\alpha > 0$$

正确性

总能找到足够小的 α，使得

$$\ell(w^k - \alpha g(w^k)) < \ell(w^k).$$

$$\ell(w^k - \alpha g(w^k)) = \ell(w^k) - \alpha g(w^k)^T g(w^k) + \mathcal{O}(\alpha^2).$$

因为

$$g(w^k)^T g(w^k) > 0$$

并且当 $\alpha \to 0$，$\alpha^2 \to 0$ 收敛的比 α 更快.
因此我们可以得出结论，对于一个足够小的 $\alpha > 0$ 我们有 $\ell(w^k - \alpha g(w^k)) < \ell(w^k)$.

决定步长

- 在经典优化中，α 通常被称为步长 (在这种情况下 $g(w^k)$ 是搜索方向)。
- 然而，设置 α 大小固定的策略可能会产生更大的开销。问题在于将 α 设置得太小会导致收敛缓慢，而将 α 设置得太大会导致发散。

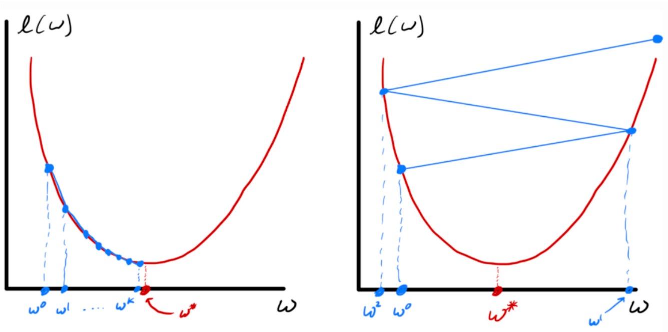


图: 步长选择导致收敛 (左) 或发散 (右)

目录

AdaGrad

- 一种选择是为每个特征自适应地设置步长。
- Adagrad 通过运行每个优化变量平方梯度的平均值来实现这一点。
- 然后，它为梯度大的变量设置一个小的学习率，为梯度小的变量设置一个大的学习率。
- 如果 w 的项添加到特征 (例如在逻辑回归中，我们可以将 w 的每项与一个特征关联起来)，而这些特征在范围或频率上是不同的，那么这一点就很重要。

Adagrad

Input: ℓ, $\nabla\ell$, parameter $\epsilon > 0$, and initial learning rate α.

Set $w_j^0 = 0$ and $z_j = 0$ for $j = 1, \ldots, d$. k = 0;

While not converged:

1. Compute entries of the gradient $g_j = \frac{\partial \ell}{\partial w_j}(w^k)$
2. $z_j = z_j + g_j^2$ for $j = 1, \ldots, d$.
3. $w_j^{k+1} = w_j^k - \alpha \frac{g_j}{\sqrt{z_j + \epsilon}}$ for $j = 1, \ldots, d$.
4. $k = k + 1$
5. Check for convergence; if converged set $\hat{w} = w^k$

Return: $\hat{w}$

关键点：每一个维度都使用自己的学习率
问题：为什么加入 ϵ?

目录

牛顿方法

核心思想

使用二阶信息 (二次近似)。

$$\ell(w^k + s) \approx \ell(w^k) + s^T g(w^k) + \frac{1}{2} s^T H(w^k) s.$$

- 我们选择一步 s，在 w^k 处显式地最小化 ℓ 的二次近似。
- 回想一下，因为 ℓ 是凸函数，对于所有 w，$H(w)$ 都是正半定的，所以这是一个明智的尝试。
- 事实上，牛顿的方法在严格的局部最小值附近具有非常好的性质，一旦足够接近一个解，它就会迅速收敛。

$$\boldsymbol{H}(\mathbf{w}) = \begin{pmatrix} \frac{\partial^2 \ell}{\partial w_1^2} & \frac{\partial^2 \ell}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 \ell}{\partial w_1 \partial w_n} \\ \vdots & \cdots & \cdots & \vdots \\ \frac{\partial^2 \ell}{\partial w_n \partial w_1} & \cdots & \cdots & \frac{\partial^2 \ell}{\partial w_n^2} \end{pmatrix},$$

牛顿方法

核心思想

使用二阶信息 (二次近似):

$$\ell(w^k + s) \approx \ell(w^k) + s^T g(w^k) + \frac{1}{2} s^T H(w^k) s.$$

- 为了简单起见，我们假设 $H(w^k)$ 是正定的。
- 我们的二次近似梯度是 $g(w^k) + H(w^k)s$。
- 这意味着线性系统 s 可以按照如下值设置:

$$g(w) + H(w)s = 0 \tag{1}$$

$$\Rightarrow s = -(H(w))^{-1} g(w). \tag{2}$$

参数更新:

$$w_{t+1} = w_t - (H(w_t))^{-1} g(w_t).$$

目录

一个简单的例子

- 有一个简单的例子清楚地说明了二阶信息是如何起作用的。
- 假设函数是一个严格的凸二次函数，即

$$\ell(w) = \frac{1}{2} w^T A w + b^T w + c$$

其中 A 是一个正定矩阵，b 是一个向量，c 是一个数值。

问题: 牛顿收敛多少步?

一个简单的例子

- 假设函数实际上是一个严格的凸二次函数，即，

$$\ell(w) = \frac{1}{2} w^T A w + b^T w + c$$

其中 A 是一个正定矩阵，b 是一个任意向量，c 是某个数字。

在这种情况下，牛顿法一步收敛 (因为 w^* 是 $Aw = b$ 的严格全局最小的唯一解)。

一个简单的例子

- 假设函数实际上是一个严格的凸二次函数，即，

$$\ell(w) = \frac{1}{2} w^T A w + b^T w + c$$

其中 A 是一个正定矩阵，b 是一个任意向量，c 是某个数字。

同时，梯度下降得到迭代序列

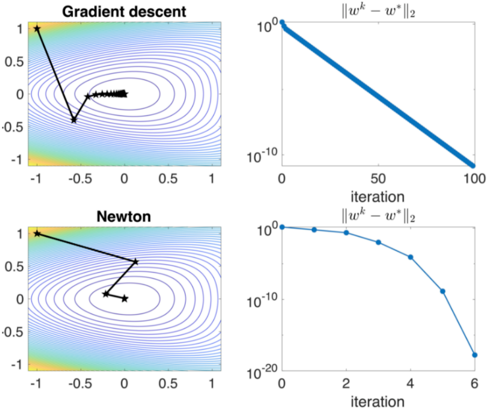
$$w^k = (I - \alpha A) w^{k-1} - \alpha b.$$

利用 $w^* = (I - \alpha A) w^* - \alpha b$ 我们可以得到

$$\begin{aligned} \|w^k - w^*\| &\leq \|I - \alpha A\|_2 \|w^{k-1} - w^*\|_2 && (3) \\ &\leq \|I - \alpha A\|_2^k \|w^0 - w^*\|_2. && (4) \end{aligned}$$

一个简单的例子

- 因此，只要 α 足够小，那么 $I-\alpha A$ 的所有特征值都在 $(-1,1)$，迭代就会收敛——但如果我们有接近 ± 1 的特征值，那么收敛速度就会很慢，。
- 更一般地，如下图所示，当接近局部最小时，牛顿法会加速收敛。



目录

最佳实践

- 矩阵 $H(w)$ 大小为 $d \times d$ 时，计算成本很高。一个好的近似是**只计算它的对角线项，并将更新乘以一个小的步长**。从本质上分析，是在做牛顿方法和梯度下降法之间的混合，其中通过逆海森矩阵来计算每个维度的步长。
- 为了避免牛顿法的发散，一种比较好的方法是从梯度下降（甚至是随机梯度下降）开始，然后完成牛顿法的优化。通常，牛顿法所使用的二阶近似更可能在最优附近更合适。

The End