14 个回答 默认排序

Andy Yang
生活、学习、思考和观察世界(不回加zenzen4649)

Lyon 等 368 人赞同了该回答

理解L1，L2 范数

L1，L2 范数即 **L1-norm** 和 **L2-norm**，自然，有L1、L2便也有L0、L3等等。因为在机器学习领域，L1 和 L2 范数应用比较多，比如作为正则项在回归中的使用 Lasso Regression(L1) 和 Ridge Regression(L2)。

因此，此两者的辨析也总被提及，或是考到。不过在说明两者定义和区别前，先来谈谈什么是范数（Norm）吧。

什么是范数?

在线性代数以及一些数学领域中，norm 的定义是

> a function that assigns a strictly positive length or size to each vector in a vector space, except for the zero vector. ——Wikipedia

简单点说，一个向量的 norm 就是将该向量**投影到 [0,) 范围内的值**，其中 0 值只有零向量的 norm 取到。看到这样的一个范围，相信大家就能想到其与现实中距离的类比，于是在机器学习中 norm 也就总被拿来**表示距离关系**：根据怎样怎样的范数，这两个向量有多远。

上面这个怎样怎样也就是范数种类，通常我们称为p-norm，严格定义是：

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}.$$

其中当 p 取 1 时被称为 1-norm，也就是提到的 **L1-norm**，同理 **L2-norm** 可得。

L1 和 L2 范数的定义

根据上述公式 L1-norm 和 L2-norm 的定义也就自然而然得到了。

先将 p=1 代入公式，就有了 L1-norm 的定义：

$$\|\boldsymbol{x}\|_1 := \sum_{i=1}^{n} |x_i|.$$

然后代入 p=2，L2-norm 也有了：

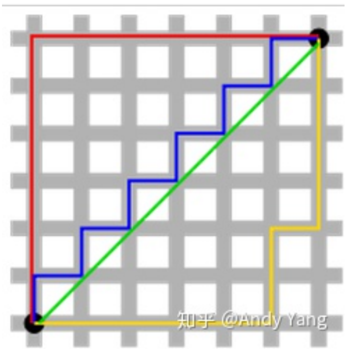
$$\|\boldsymbol{x}\|_2 := (\sum_{i=1}^{n} |x_i|^2)^{\frac{1}{2}}$$

L2 展开就是熟悉的欧几里得范数：

▲ 赞同 368 ▼ 10 条评论 分享 收藏 喜欢 ··· 收起

$$\|\boldsymbol{x}\|_2 := \sqrt{x_1^2 + \cdots + x_n^2}.$$

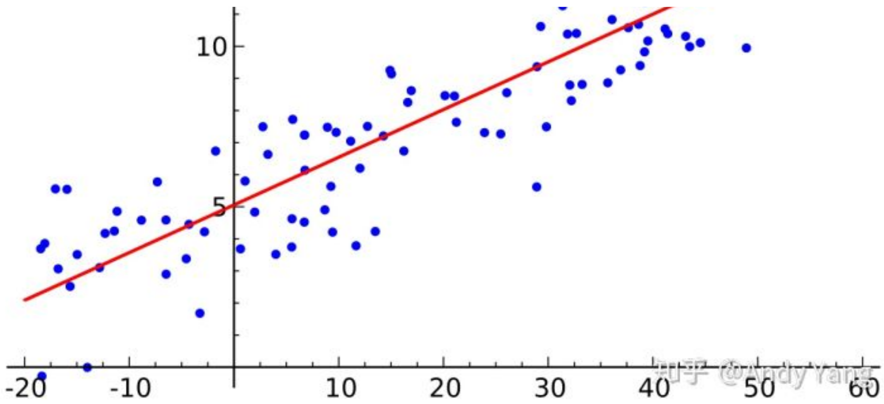
题外话，其中 L1-norm 又叫做 taxicab-norm 或者 Manhattan-norm，可能最早提出的大神直接用在曼哈顿区坐出租车来做比喻吧。下图中绿线是两个黑点的 L2 距离，而其他几根就是 taxicab 也就是 L1 距离，确实很像我们平时用地图时走的路线了。



L1 和 L2 范数在机器学习上最主要的应用大概分下面两类

- 作为**损失函数**使用
- 作为**正则项**使用也即所谓 **L1-regularization** 和 **L2-regularization**

我们可以担当损失函数



我们需要做的是，获得一条线，让数据点到线上的总距离（也就是error）最小。

还记得之前在范数介绍中提到的用来表示距离吗，于是也可以用能表示距离的 L1-norm 和 L2-norm 来作为损失函数了。

首先是 L1-norm 损失函数，又被称为 **least absolute deviation (LAD，最小绝对偏差)**

$$S = \sum_{i=1}^{n} |y_i - f(x_i)|.$$

之后是大家最熟悉的 L2-norm 损失函数，又有大名**最小二乘误差 (least squares error, LSE)**:

$$S = \sum_{i=1}^{n}(y_i - f(x_i))^2$$

这个便不多解释了。

那么问题来了，这里不谈挖掘机，为什么大家一般都用 L2 损失函数，却不用 L1 呢?

这个就说来话长了，如果你问一个学习过微积分的同学，如何求一个方程的最小值，他/她大概会想当然的说：“求导，置零，解方程。” 号称微积分届的农夫三拳。

但如果给出一个绝对值的方程，突然就会发现农夫三拳不管用了，求最小值就有点麻烦了。主要是因为绝对值的倒数是不连续的。

同样的对于 L1 和 L2 损失函数的选择，也会碰到同样的问题，所以最后大家一般用 L2 损失函数而不用 L1 损失函数的原因就是:

因为计算方便!

可以直接求导获得取最小值时各个参数的取值。

此外还有一点，**用 L2 一定只有一条最好的预测线，L1 则因为其性质可能存在多个最优解。**（更多关于L1 L2 损失函数参考索引5）

当然 L1 损失函数难道就没有什么好处了吗，也是有的，那就是**鲁棒性 (Robust) 更强，对异常值更不敏感。**

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \boxed{\lambda \sum_{i=1}^{k} |w_i|}$$

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \boxed{\lambda \sum_{i=1}^{k} w_i^2}$$

这两个正则项最主要的不同，包括两点:

- 如上面提到的，**L2 计算起来更方便**，而 L1 在特别是非稀疏向量上的计算效率就很低;
- 还有就是 L1 最重要的一个特点，**输出稀疏**，会把不重要的特征直接置零，而 L2 则不会;
- 最后，如之前多次提过，L2 有唯一解，而 L1 不是。

这里关于第二条输出稀疏我想再进行一些详细讲解，因为 L1 天然的输出稀疏性，把不重要的特征都置为 0，所以它也是**一个天然的特征选择器。**

可是为什么 L1 会有这样的性质呢，而 L2 没有呢？这里用个直观的例子来讲解。

来一步一步看吧，首先获知用梯度下降法来优化时，需要求导获得梯度，然后用以更新参数。

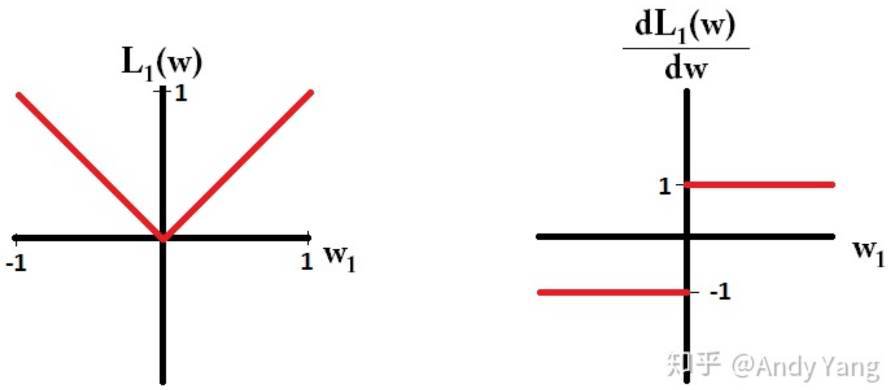
$$\theta = \theta - \alpha \frac{\partial}{\partial \theta} J(\theta)$$

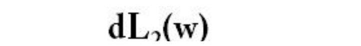
于是分别先对 L1 正则项和 L2 正则项来进行求导，可得。

$$\frac{dL_1(w)}{dw} = sign(w)$$

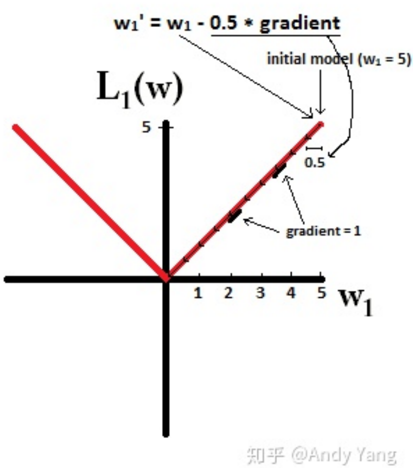
$$\frac{dL_2(w)}{dw} = w$$

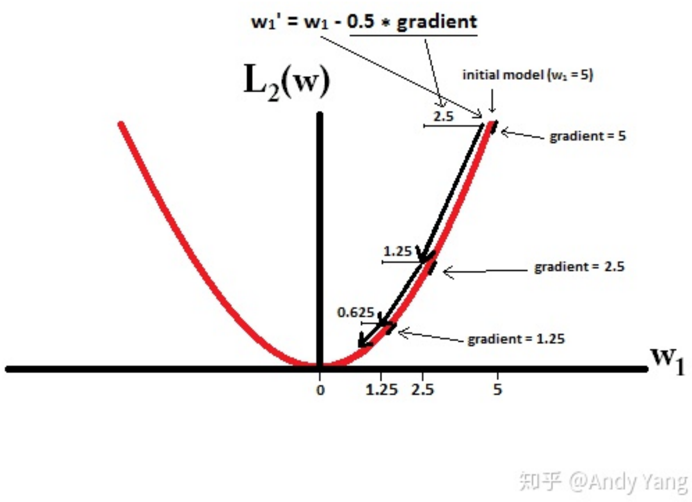
之后将 L1 和 L2 和它们的导数画在图上





于是会发现，在梯度更新时，不管 L1 的大小是多少（只要不是0）梯度都是1或者-1，所以每次更新时，它都是稳步向0前进。





也就是说加了 L1 正则的话基本上经过一定步数后很可能变为0，而 L2 几乎不可能，因为在值小的时候其梯度也会变小。于是也就造成了 L1 输出稀疏的特性。

Reference

1. Differences between L1 and L2 as Loss Function and Regularization
2. Why L1 norm for sparse models
3. L1 Norms versus L2 Norms
4. Norm (mathematics)-Wiki
5. Why we use "least squares" regression instead of "least absolute deviations" regression