基于 Logistic 回归模型的 ADMM 算法与 $L_{\frac{1}{2}}$ 正则化

摘要

交替方向乘子法（ADMM）是一种求解优化问题的计算框架, 适用于求解分布式凸优化问题, 特别是统计学习问题。ADMM 通过分解协调过程, 将大的全局问题分解为多个较小、较容易求解的局部子问题，并通过协调子问题的解而得到大的全局问题的解。$L_{1/2}$ 正则子在 2008 年由徐宗本院士提出, 其具有无偏性、稀疏性及 Oracle 等优良理论性质。本文将通过 Logistic 模型将两者具体化地阐述。本文首先对 Logistic 回归模型与损失函数正则化进行了简要介绍, 然后介绍了 ADMM 算法的基本性质以及在具体模型中的应用与优势，最后引入了 $L_{1/2}$ 正则子并通过实验证明其可行性与收敛性。本文中所有数据库与算法百度网盘链接：链接：链接：https://pan.baidu.com/s/15uZbHKKOoRlq1cQOMA9wYw 提取码：ADMM

关键词 逻辑斯谛回归，交替方向乘子法，$L_{1/2}$ 正则子

目录

第 1 章　Logistic 回归模型与正则化

1.1　Logistic 回归模型简介

以评估冰红茶等饮料为例，口味、价格、容量是选取一款冰红茶的基本因素，他们可以简单的构成评估指标，我们通常会将这些指标进行一个整合，再做评估和筛选

$$Y=A\ 口味\ +B\ 价格\ +C\ 容量\ +D$$

这就是一个简单的线性回归，我们不妨只关注于其中一个指标-容量，在图中我们可以得到一个估算的结果，容量越多的评估结果也可能越高，我们在图中就可以筛选出评估结果最高的几位，但是问题来了，一些钟爱粉对自己喜爱的品类打出超高的口味分数，这些极端值导致某些品类在下图中的评估结果远远偏离正常的分布范围（即图中的蓝色点）

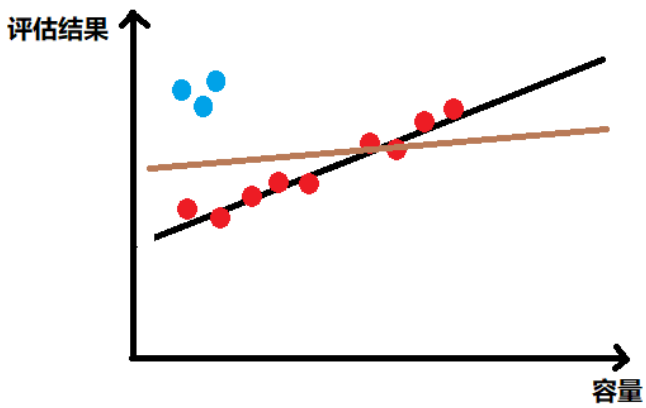


图 1.1　评估结果

这会导致误差会特别大，那么怎么办呢，我们通常会用一些平滑曲线去减小这些极端值对于整体分布的影响，从而使整体的分布更加集中。比如 sigmoid 函数，也称 S 型生长曲线，我们可以把极端值产生的影响变得微小，且将输出限制在 0-1 之间

其中 Sigmoid 函数的表达式为

$$P = Sigmoid(\mathbf{y}) = \frac{1}{1+e^{-y}}$$

我们将线性回归的函数 $y = wx + b$ 与 Sigmoid 函数一拼接，就组成了 Logistic 回归函数

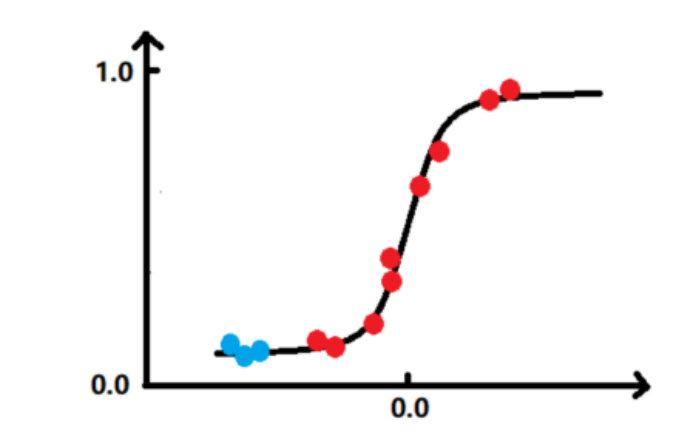


图 1.2　Sigmoid 函数图像

$$P = \frac{1}{1+e^{-wx+b}}$$

普通的线性回归输出值不受限制，可以是任意的数值，因此也常用作估计价格，产量等等，而 Logistic 回归模型输出在 0-1 之间，作概率预测的时候被更多用在分类上，这也是为什么他被用于二分类问题上，例如股票买还是卖，面容识别通过还是不通过，病人患病还是未患病等等。有了模型，接下来我们该如何评估逻辑回归中的准确度？在线性回归中，我们引入了平方差作为衡量数据离散程度的指标，而在 Logistic 回归模型中，我们需要度量两个事件或者分布之间的距离，因此我们引入交叉熵作为损失函数

$$Loss = -\sum \{y_{ture}\, ln\,(p) + (1 - y_{ture})\, ln\,(1 - p)\}$$

在信息论中，交叉熵是一种度量两个概率分布之间差异的方法，用于评估预测结果和真实标签之间的差异，其中 y 为真实标签（0/1），p 为模型预测为正概率，交叉熵函数有几个优点：1. 非负 2. 对称（对数运算）3. 凸性（两个凸函数相加，海塞矩阵半正定，梯度下降法求解）4. 可加性。交叉熵函数和平方差一样，当然是越小越好，因此在训练过程中，借助交叉熵函数，我们可以不断调整模型的参数使得交叉熵函数值达到最小，从而模型拟合准确率能达到最高。Logistic 模型优点很突出，不仅能计算概率，还能减小极端值影响。

1.2 正则化简介

接下来我们简单引入一下正则化，首先我们来谈谈为什么要正则化，什么是正则化，我们都知道过拟合是对所见数据非常完美几乎一个不落地串珠子串起来，如果一个模型过拟合，那么他将是一个比较复杂的非线性方程

$$y = a + k_1x + k_2x^2 + k_3x^3 + k_4x^4 + ...$$

多次项参数 k_1, k_2, k_3, k_4 等等…将会让曲线弯来弯去，使劲往每一个数据点上靠，过拟合会造成不必要的数据联系，从而影响最终的结果，就比如我们要找郑智结果出来郑智化。比起黑色的复杂曲线，我们可能更希望得到的是这条棕色曲线，他能更有效的概括数据总体趋势，相比起黑色的曲线，他将除 k_1 外的参数 k_2, k_3, k_4… 都设为 0，虽然误差变大了，但概括数据能力却更强。那么我们怎样在机器学习中学出这样的参数呢，这就是正则化出现的原因了。

常见的正则化分为 L_1 和 L_2 正则化，对于刚刚的模型线条，我们一般用平方差来表示模型相较于真实值的误差

$$Loss(i) = [y_i(x) - y]^2$$

而 L_1, L_2 只是在平方差的基础上加上一些惩罚式子，让误差不仅仅取决于数据偏离的多少，而且取决于像 k_1, k_2, k_3, k_4 等等多次项参数值的大小，如果是每个参数的二范数也就是平方和的开根号，我们称之为 L_2 正则化

$$Loss(i) = [y_i(x) - y]^2 + \sqrt{k_1^2 + k_2^2 + k_3^2 + ...}$$

如果是每个参数的一范数，也就是绝对值，我们称之为 L_1 正则化

$$Loss(i) = [y_i(x) - y]^2 + (|k_1^2| + |k_2^2| + |k_3^2| + ...)$$

这能使得机器学习在减小误差过程中约束了非线性更强的参数随心所欲变化。这就是正则化的基本思想。那 L_1, L_2 正则化有什么不同呢？假设现在只有两个参数 k_1, k_2，蓝色的圆心是误差最小的地方，每条蓝线上的误差都一样，正规化就是在红线上产生额外误差，也就上述惩罚式子，在红线上额外误差的值也都一样，所以在蓝线和红线交点处，能让两个误差的和最小。

值得一提的是，使用 L_1 方法的时候，很可能得到的结果是 $k_2 = 0$，只有 k_1 的特征被保留了下来，所以模型参数的某些维度更容易在 0 处产生解，常常产生稀疏解，比如区分老鼠和猪的时候，k_1 是体型大小，k_2 是耳朵长短，应用 L_1 就能只把 k_1 体型大小特征留下，更方便区分。但是 L_1 不是很稳定，受噪声影响，每次训练都会产生稍稍不同的误差曲线，而 L_2 针对这种变化，切点的移动不会很大，而 L_1 的切点会跳到不同的地方，因此 L_2 对噪声友好。

最后我们会选用 λ 来控制这种正规化的强度（两方向梯度相同），p 来代表对参数正规化的程度

$$Loss(i) = [y_i(x) - y]^2 + \lambda \sum \| k_j \|_p$$

只有在 L_p 范数的 p 大于等于 1 的时候，构成的集合才是凸集，对应一个凸优化问题，L_1, L_2 某种程度上也是应用了凸集的特性。这就是正规化的基本表达式。

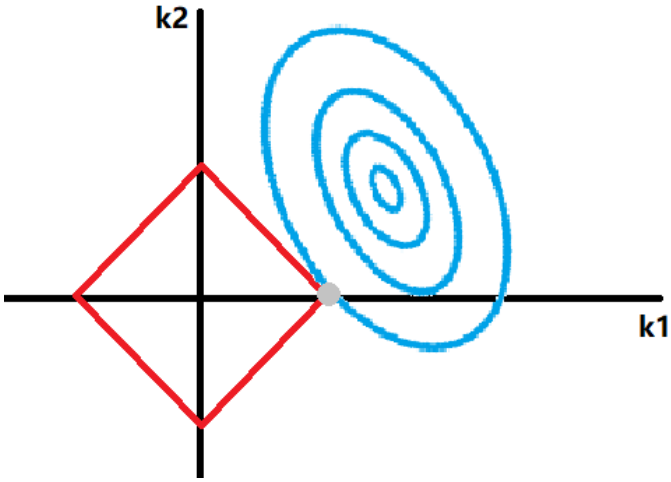


图 1.3　L_1 正则化

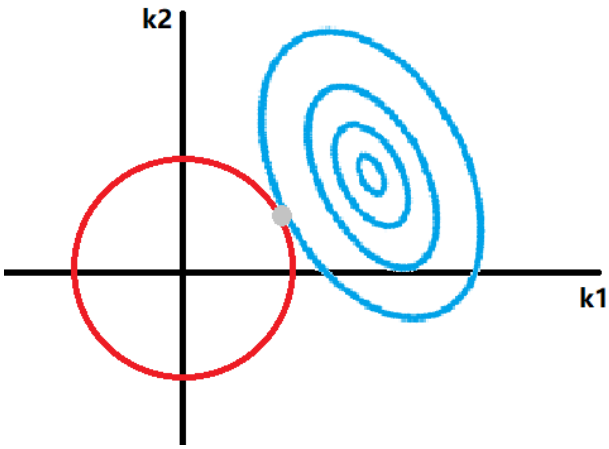


图 1.4　L_2 正则化

第 2 章　ADMM 算法

交替方向乘子法（ADMM）是一种求解优化问题的计算框架, 适用于求解分布式凸优化问题, 特别是统计学习问题。ADMM 通过分解协调过程, 将大的全局问题分解为多个较小、较容易求解的局部子问题，并通过协调子问题的解而得到大的全局问题的解。ADMM 最早分别由 Glowinski Marrocco 及 Gabay Mercier 于 1975 年和 1976 年提出，并被 Boyd 等人于 2011 年重新综述并证明其适用于大规模分布式优化问题。由于 ADMM 的提出早于大规模分布式计算系统和大规模优化问题的出现，所以在 2011 年以前，这种方法并不广为人知。

ADMM 是一种求解具有可分离的凸优化问题的重要方法，由于处理速度快，收敛性能好，ADMM 算法在统计学习、机器学习等领域有着广泛应用。ADMM 被广泛应用于各种不同类型的优化问题，包括线性规划、二次规划、稀疏优化、信号处理、统计学习等。

2.1　ADMM 算法的基本框架

交替方向乘子法（ADMM）通常用于解决存在两个优化变量的等式约束优化类问题，其一般形式为:

$$\begin{aligned} &\min_{x,z} f(\mathbf{x}) + g(\mathbf{z}) \\ &s.t.\ A(\mathbf{x}) + B(\mathbf{z}) = c \end{aligned}$$

其中 $x \in R^n$ 和 $z \in R^m$ 是优化变量，f(x) 和 g(z) 是凸函数，$A \in R^{p\times n}$，$B \in R^{p\times m}$，$c \in R$

标准的 ADMM 算法解决的是一个等式约束的问题，且该问题两个函数 f 和 g 是成线性加法的关系。这意味着两者实际上是整体优化的两个部分，两者的资源占用符合一定等式，对整体优化贡献不同，但是是简单加在一起的。

ADMM 算法的核心是原始对偶算法的增广拉格朗日法。拉格朗日函数是解决了多个约束条件下的优化问题，这种方法可以求解一个有 n 个变量与 k 个约束条件的优化问题。增广拉格朗日函数就是在关于原问题的拉格朗日函数之后增加了一个和约束条件有关的惩罚项，惩罚项参数 $\rho > 0$。惩罚项参数影响迭代效率。

我们通过引入增广拉格朗日函数，就可将原来的等式约束优化问题问题等价转化为对增广拉格朗日函数求极值的无约束优化问题。

$$L(\mathbf{x}, \mathbf{u}, \mathbf{z}) = f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{u}^T(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - c) + \frac{\rho}{2}\left\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - c\right\|_2^2$$

2.2 ADMM 算法步骤

1. 定义拉格朗日乘子 $\mathbf{u}^{(0)}$ 和参数 $\rho > 0$，并初始化优化变量 $\mathbf{x}^{(0)}$ 和 $\mathbf{z}^{(0)}$

2. 对于，$k = 0, 1, 2, ...$ 重复以下步骤

$$\mathbf{x}^{(k+1)} = arg\ \underset{\mathbf{x}}{min}\ L(\mathbf{x}, \mathbf{z}^{(k)}, \mathbf{u}^{(k)})$$
$$\mathbf{z}^{(k+1)} = arg\ \underset{\mathbf{z}}{min}\ L(\mathbf{x}^{(k+1)}, \mathbf{z}, \mathbf{u}^{(k)})$$
$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \rho(\mathbf{A}\mathbf{x}^{(k+1)} + \mathbf{B}\mathbf{z}^{(k+1)} - c)$$

3. 当收敛条件满足后，中止迭代，收敛条件可以是优化目标函数值的变化量或者优化变量的变化量小于某个给定的阈值。

2.3 Logistic 回归模型中的 ADMM 算法

在 Logistic 算法中，我们需要优化的目标函数为交叉熵损失函数

$$min\ Loss(\mathbf{w}) = -\sum_{i=1}^{N} \{y_i\, ln\, [h_w(\mathbf{x}_i)] + (1 - y_i)\, ln\, [1 - h_w(\mathbf{x}_i)]\}$$

其中样本个数为 N，$h_w(\mathbf{x}_i) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}_i}}$，$\mathbf{x}_i$ 为第 i 个样本，$\mathbf{y}$ 为样本的分类标签

加上 L_1 正则化项即得

$$min\ Loss(\mathbf{w}) = -\sum_{i=1}^{N} \{y_i\, ln\, [h_w(\mathbf{x}_i)] + (1 - y_i)\, ln\, [1 - h_w(\mathbf{x}_i)]\} + \lambda \left\|\mathbf{w}\right\|_1$$

将上述损失函数改写成等式约束的形式

$$min\ Loss(\mathbf{w}) = -\sum_{i=1}^{N} \{y_i\, ln\, [h_w(\mathbf{x}_i)] + (1 - y_i)\, ln\, [1 - h_w(\mathbf{x}_i)]\} + \lambda \left\|\boldsymbol{\xi}\right\|_1$$
$$s.t.\ \mathbf{w} - \boldsymbol{\xi} = 0$$

改写为增广拉格朗日函数

$$min\ Loss(\mathbf{w}, \boldsymbol{\xi}, \mathbf{u}) =$$
$$-\sum_{i=1}^{N} \{y_i\, ln\, [h_w(\mathbf{x}_i)] + (1 - y_i)\, ln\, [1 - h_w(\mathbf{x}_i)]\} + \lambda \left\|\boldsymbol{\xi}\right\|_1 + \mathbf{u}^T (\mathbf{w} - \boldsymbol{\xi}) + \frac{\rho}{2} \left\|\mathbf{w} - \boldsymbol{\xi}\right\|_2^2$$

其中 $\mathbf{y}, \mathbf{x}_i \in R^{N\times 1}$;$\mathbf{w}, \boldsymbol{\xi}, \mathbf{u} \in R^{p\times 1}$, N 为样本个数，p 为样本特征个数

我们对 L 用 ADMM 算法，先对 w 进行优化，通过之前的推导，我们可以看出来 L 是由凸函数的和组成的多项式，所以为求

$$\mathbf{w}^{(k+1)} = arg\ \underset{\mathbf{w}}{min}\ L(\mathbf{w}, \boldsymbol{\xi}^{(k)}, \mathbf{u}^{(k)})$$

可采用梯度下降法，ΔL 的每个分量为

$$\frac{\partial L}{\partial \mathbf{w}_j} = \sum_{i=1}^{N} (h_w(\mathbf{x}_i) - y_i)\mathbf{x}_i^{(j)} + \mathbf{u}_j + \rho(\mathbf{w}_j - \boldsymbol{\xi}_j)$$

所以得到 $\mathbf{w}$ 的更新式为 $\mathbf{w}=\mathbf{w}-\alpha\Delta L$ 其中 α 为学习率，由此通过迭代我们可以得到在参数 $\boldsymbol{\xi}^k,\mathbf{u}^k$ 下的最佳参数 $\mathbf{w}^{k+1}$

接下来，我们更新 ξ，首先我们去除 L 中与无关的项，可得

$$\boldsymbol{\xi}^{(k+1)}=arg\ \underset{\mathbf{w}}{min}\ \left\{\lambda\left\|\boldsymbol{\xi}\right\|_1+\mathbf{u}^T\left(\mathbf{w}-\boldsymbol{\xi}\right)+\tfrac{\rho}{2}\left\|\mathbf{w}-\boldsymbol{\xi}\right\|_2^2\right\}$$

等式右边乘上常数 $\frac{2}{\rho}$, 不改变所求的参数值，故可等价地改写成

$$\boldsymbol{\xi}^{(k+1)}=arg\ \underset{\mathbf{w}}{min}\ \left\{\tfrac{2\lambda}{\rho}\left\|\boldsymbol{\xi}\right\|_1+\left\|\boldsymbol{\xi}-(\tfrac{\mathbf{u}}{\rho}+\mathbf{w})\right\|_2^2\right\}$$

根据近端梯度下降法，上述形式可以用软阈值算子求解，即

$$\boldsymbol{\xi}^{(k+1)}=S_{\frac{\lambda}{\rho}}(\tfrac{\mathbf{u}^{(k)}}{\rho}+\mathbf{w}^{(k+1)})$$

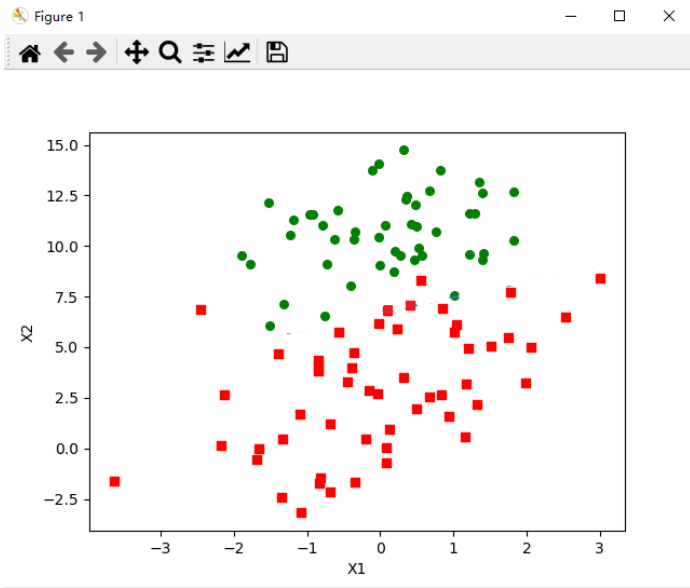
其中，软阈值算子的表达式为

$$\left[S_{\frac{\lambda}{\rho}}\left(\frac{u^{(k)}}{\rho}+w^{(k+1)}\right)\right]_i=\begin{cases}\dfrac{u_i}{\rho}+w_i-\dfrac{\lambda}{\rho},\dfrac{u_i}{\rho}+w_i>\dfrac{\lambda}{\rho}\\ 0,\left|\dfrac{u_i}{\rho}+w\right|\le\dfrac{\lambda}{\rho}\\ \dfrac{u_i}{\rho}+w_i+\dfrac{\lambda}{\rho},\dfrac{u_i}{\rho}+w_i<-\dfrac{\lambda}{\rho}\end{cases}$$

最后，我们来更新 $\mathbf{u}$，$\mathbf{u}^{(k+1)}=\mathbf{u}^k+\rho\left(\mathbf{w}^{(k+1)}-\boldsymbol{\xi}^{(k+1)}\right)$

2.4　ADMM 算法数值模拟

我们找到了 100 份二维样本数据并运用上述算法进行测试，前 80 组数据作为训练集，后 20 个数据作为测试集，其大致分类如图，相同颜色为一类。



如果运用传统的梯度下降方法进行分类，我们测试出来当 λ 取 0.2-0.3 时分类效果最好，误分类点为 2，测试集错误率为 0。

在 ADMM 算法下，我们仍取 $\lambda = 0.2$，经训练后，其训练集与测试集错误率如下

也就是说，在 ADMM 算法之下产生了 3 个误分类点，我们猜测可能是由于 ρ 取的不合理导致了误分类点的增多。此外在测试算法的过程中，我们还发现 ADMM 算法相较于传统的梯度下降法有一些不同之处。我们在相同的阈值之下用梯度下降法求得最优的权值一共是迭代

了 500 多次，ADMM 算法总计迭代了 435 次，而且 ADMM 算法有一个很好的特点就是它的对权值 w 的迭代可以做到每次量都比较少，且一次比一次少，在我们的测试中，对权值 w 的第一轮迭代只有 30 次。

第 3 章　$L_{\frac{1}{2}}$ 正则化

在 2008 年由徐宗本院士提出基于非凸罚的 $L_{1/2}$ 正则子, 并证明其具有无偏性、稀疏性及 Oracle 等优良理论性质。给出一种重赋权迭代算法, 将求解 $L_{1/2}$ 正则子转化为一系列 L_1 正则化子迭代求解。与经典的 L_0 正则子相比,$L_{1/2}$ 正则子更容易求解, 而与当今流行的 L_1 正则子相比,$L_{1/2}$ 正则子产生更稀疏的解. 实验表明,$L_{1/2}$ 正则子可替代 L_p $(0 < p < 1)$ 正则子, 具有重要而广泛的应用价值。

3.1　$L_{\frac{1}{2}}$ 正则化子 [1]

首先给出 $L_{\frac{1}{2}}$ 正则子的一个基本结构

$$\boldsymbol{\beta} = \underset{\boldsymbol{\beta}}{arg\,min} \left\{ \frac{1}{n} \sum_{i=1}^{n} (\mathbf{Y}_i - \mathbf{X}_i^T \boldsymbol{\beta})^2 + \lambda \sum_{i=1}^{p} |\beta_i|^{\frac{1}{2}} \right\}$$

正则子的约束区域是一个旋转的正方形，Lasso 的解是损失函数的等高线与正方形顶点相交所对应的取值，当等高线交于坐标轴时产生一个零系数的解，从而解具有稀疏性。从图中可以看出交于坐标轴的可能性很小，从而解不具有稀疏性，而正则子的约束区域与等高线更容易相交于坐标轴，所以解会相比于正则子更具有稀疏性。

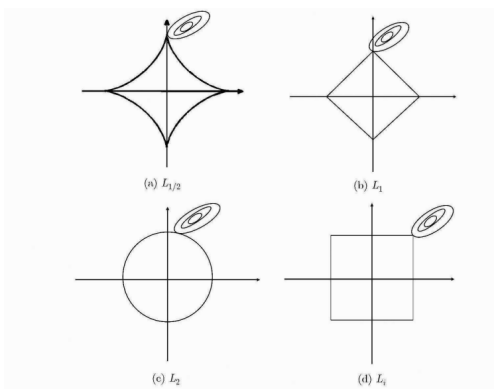


图 3.1　不同的正则化子

3.2 $L_{\frac{1}{2}}$ 正则 Logistic 回归算法 [2]

步骤一：令 t=0，设最大迭代次数 K，初始化 $\boldsymbol{\beta}^0 = (1, ..., 1)^T$

步骤二：求解 $\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta} = \underset{\boldsymbol{\beta}}{arg\,min} \left\{ \frac{1}{n} \sum_{i=1}^{n} (\mathbf{Y}_i - \mathbf{X}_i^T \boldsymbol{\beta})^2 + \lambda \sum_{i=1}^{p} |\boldsymbol{\beta}_i|^{\frac{1}{2}} \right\}$

步骤三：当 $t < K$ 时，转步骤二，当 $t = K$ 时，输出 $\boldsymbol{\beta}^t$

在上述算法中，由步骤一设定的最大迭代次数 K 作为算法终止的条件，且初始化参数 $\beta^0 = (1, ..., 1)^T$ 于是步骤二中的第一次迭代对应于求解一个 L_1 正则子问题，第二次迭代中相当于求解一个加权的 L_1 正则子，经由简单线性变化仍可化为求解 L_1 正则子的问题。

由于在第二次迭代后可能会出现 $\beta_i^t = 0$ 的情况，为了避免该情况，使用 $\frac{1}{\sqrt{|\beta_i^t| + \epsilon}}$ 来代替 $\frac{1}{\sqrt{|\beta_i^t|}}$，其中 ϵ 为任给的一个正数。

对于步骤 2，由于该目标函数第二项是凹函数，故求解时并不能使用传统的梯度下降法或者牛顿法，所以我们需要使用坐标下降法对其进行求解。

为了方便阅读，我们将接下来要用到的符号先进行说明：$\{\mathbf{X}_i, y_i\}_{i=1}^n$ 为我们的数据集，个数为 n，每个数据集的特征数为 p，即 $X_i = (x_i^{(1)}, x_i^{(2)}, ..., x_i^{(p)})^T$，记 $\boldsymbol{\beta} = (\beta_0, \beta_1, ..., \beta_p)$ 为带偏置的权值，其中 β_0 为常数. 同时令 $\mathbf{x}_i = (1, X_i^T)^T$ $(i = 1, 2, ..., p)$ 为对应于 $\boldsymbol{\beta}$ 的数据。

根据前面对 Logistic 回归模型损失函数以及正则化项的介绍，我们这里直接给出 $L_{\frac{1}{2}}$ 正则化子的损失函数

$$Loss = \underset{\boldsymbol{\beta}}{min} \left\{ -\sum_{i=1}^{n} [y_i (\mathbf{x}_i^T \boldsymbol{\beta}) - ln(1 + e^{(\mathbf{x}_i^T \boldsymbol{\beta})})] + \lambda \sum_{i=1}^{p} |\beta_i|^{\frac{1}{2}} \right\}$$

先考虑不带正则化项的损失函数，记

$$I(\boldsymbol{\beta}) = \underset{\boldsymbol{\beta}}{min} \left\{ -\sum_{i=1}^{n} [y_i (\mathbf{x}_i^T \boldsymbol{\beta}) - ln(1 + e^{(\mathbf{x}_i^T \boldsymbol{\beta})})] \right\}$$

前文有介绍过，这是一个凸函数，故求此最优解，可用牛顿方法，先对 β 求导

$$\frac{\partial I(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = -\sum_{i=1}^{n} \mathbf{x}_i (y_i - \mu(\mathbf{x}_i; \boldsymbol{\beta}))$$

其中

$$\mu(\mathbf{x}_i; \boldsymbol{\beta}) = \frac{e^{(\mathbf{x}_i^T \boldsymbol{\beta})}}{1 + e^{(\mathbf{x}_i^T \boldsymbol{\beta})}}$$

当求得 $I(\boldsymbol{\beta})$ 的最优解时 $\frac{\partial I(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = 0$, 接下来采用 Newtown-Raphson 迭代，求得其 Hessian 矩阵为

$$\frac{\partial^2 I(\boldsymbol{\beta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T} = \sum_{i=1}^{n} x_i x_i^T \mu(\mathbf{x}_i; \boldsymbol{\beta})(1 - \mu(\mathbf{x}_i; \boldsymbol{\beta}))$$

所以 β 的迭代式为

$$\boldsymbol{\beta}^{new} = \boldsymbol{\beta}^{old} - (\frac{\partial^2 I(\boldsymbol{\beta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T})^{-1} \frac{\partial I(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}$$

为了方便表示，记 $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_n)^T$,$\mathbf{W} = diag(w_i)$ $(i = 1, ..., n)$, 其中 $w_i = \mu(\mathbf{x}_i; \boldsymbol{\beta})(1 - \mu(\mathbf{x}_i; \boldsymbol{\beta}))$,$\mu = (\boldsymbol{\mu}(\mathbf{x}_1; \boldsymbol{\beta}), ..., \mu(\mathbf{x}_n; \boldsymbol{\beta}))^T$ 由此，上述迭代式可表示成

$$\begin{aligned}\boldsymbol{\beta}^{new} &= \boldsymbol{\beta}^{old} + (\mathbf{X^{TWX}})^{-1}\mathbf{X}^T(\mathbf{y}-\boldsymbol{\mu}) \\ &= (\mathbf{X^{TWX}})^{-1}\mathbf{X^{TWz}}\end{aligned}$$

其中 $\mathbf{z} = \mathbf{X}\boldsymbol{\beta}^{old} + \mathbf{W}^{-1}(\mathbf{y}-\boldsymbol{\mu})$

将我们带 $L_{\frac{1}{2}}$ 正则子的损失函数写成矩阵元素相加的形式

$$Loss = \min_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{n} w_i(z_i - \beta_0 - \mathbf{x}_i^T\boldsymbol{\beta})^2 + \lambda \sum_{i=1}^{p} |\beta_i|^{\frac{1}{2}} \right\}$$

由坐标下降法，每次只优化一个变量，分步地优化出每一个 β_i 的值从而得到最优的权值函数 $\boldsymbol{\beta}$ 的值，在只有一个变量变动，其他变量固定的情况下，损失函数可以看作一个变量变动的一元函数，记对应于变量 β_k 的一元函数为 $f_k(\beta_k)$，其中 $k = (0, 1, ..., n)$。

当 $k = 0$ 时，

$$f_0(\beta_0) = \sum_{i=1}^{n} w_i(z_i - \beta_0 - \mathbf{x}_i^T\boldsymbol{\beta})^2 + \lambda \sum_{i=1}^{p} |\beta_i|^{\frac{1}{2}}$$

由于式子里除了 β_0 为变量，其余符号均代表一个常量，所以 $f_0(\beta_0)$ 的极值在其一阶导数取 0 的时候取到，即

$$\frac{df_0(\beta_0)}{d\beta_0} = -2\sum_{i=1}^{n} w_i(z_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j) = 0$$

解得当其余变量固定时，最优的

$$\beta_0 = \frac{\sum_{i=1}^{n} w_i(z_i - \sum_{j=1}^{p} x_{ij}\beta_j)}{\sum_{i=1}^{n} w_i}$$

同理，接下来求其余的 β_k

$$f_k(\beta_k) = \sum_{i=1}^{n} w_i(z_i - \beta_0 - x_i k\beta_k - \sum_{j\neq k} x_{ij}\beta_j)^2 + \lambda(|\beta_k|^{\frac{1}{2}} + \sum_{j\neq k} |\beta_j|^{\frac{1}{2}})$$

上式中含有变量 β_k 的项为

$$\sum_{i=1}^{n} w_i x_{ik}^2\beta_k^2 - 2\sum_{i=1}^{n} w_i(z_i - \beta_0 - \sum_{j\neq k} x_{ij}\beta_j)x_{ik}\beta_k + \lambda|\beta_k|^{\frac{1}{2}}$$

对 $f_k(\beta_k)$ 求极小化即对上述式子求极小化，我们假定 $\sum_{i=1}^{n} w_i x_{ik}^2 \neq 0$ 并对上述式子除以 $\sum_{i=1}^{n} w_i x_{ik}^2$，得到一个等价的优化目标函数

$$h_k(\beta_k) = \beta_k^2 - 2\beta_k \frac{\sum_{i=1}^{n} w_i(z_i - \beta_0 - \sum_{j\neq k} x_{ij}\beta_j)x_{ik}}{\sum_{i=1}^{n} w_i x_{ik}^2} + \frac{\lambda}{\sum_{i=1}^{n} w_i x_{ik}^2}|\beta_k|^{\frac{1}{2}}$$

为方便后续表示，我们记上述多项式第二和第三项的系数为 C_k 和 λ_k，即

$$C_k = \frac{\sum_{i=1}^{n} w_i(z_i - \beta_0 - \sum_{j\neq k} x_{ij}\beta_j)x_{ik}}{\sum_{i=1}^{n} w_i x_{ik}^2},\ \lambda_k = \frac{\lambda}{\sum_{i=1}^{n} w_i x_{ik}^2}$$

则得到 $h_k(\beta_k) = \beta_k^2 - 2C_k\beta_k + \lambda_k|\beta_k|^{\frac{1}{2}}$ 对其求导，可得

$$h_k^{'}(\beta_k) = 2\beta_k - 2C_k + \lambda_k \frac{sign(\beta_k)}{2\sqrt{|\beta_k|}}$$

当 $\beta_k > 0$ 时，令 $t = \sqrt{|\beta_k|}$, 则得到

$$h_k^{'}(\beta_k) = \frac{2}{t}(t^3 - C_k t + \frac{\lambda_k}{4})$$

令 $g(t) = (t^3 - C_k t + \frac{\lambda_k}{4})$, 由于在任何情况下 $\frac{2}{t} \neq 0$ 故 $h_k^{'}(\beta_k)$ 的零点完全由 $g(t)$ 决定，故这个问题即转换为一个一元三次函数求根问题。

我们将 $g(t)$ 化为一元三次函数求根的标准形式，即令 $p = \frac{-1}{3}C_k$,$q = \frac{1}{8}\lambda_k$, 得到

$$g(t) = t^3 + 3pt + 2q$$

根据一元三次函数的求根判别式 $\Delta = (\frac{2q}{2})^2 + (\frac{3p}{3})^3$ 可知，当 $\Delta > 0$ 即 $C_k > \frac{3}{4}\lambda_k^{\frac{2}{3}}$ 时 $g(t)$ 有一个实根和两个虚根，解得其三个根为

$$t_k = 2\sqrt{\frac{C_k}{3}} cos\frac{\pi - \varphi + 2k\pi}{3}\ , k = 0, 1, 2$$

t_0 为我们需要的解，由此得到 $\beta_k = t_0^2 = \frac{2}{3}(1 + cos(\frac{2\pi - 2\varphi}{3}))$, 其中 $\varphi = arccos(\frac{\lambda_k}{8}(\frac{C_k}{3})^{\frac{-3}{2}})$

当 $C_k \leq \frac{3}{4}\lambda_k^{\frac{2}{3}}$ 时, 我们分两种情况来讨论。

第一种情况：如果 $C_k < 0$, 那么 $h_k^{'}$ 在 $t \in (0, +\infty)$ 上是单调增的，所以在 $\beta_k = 0$ 处取到最小值。

第二种情况：如果 $0 \leq C_k \leq \frac{3}{4}\lambda_k^{\frac{2}{3}}$, 则先求 $g(t)$ 的极小值点，令 $g^{'}(t) = 0$ 则得到其极小值点为 $t = \sqrt{\frac{C_k}{3}}$ 带入 g 得到

$$\begin{aligned} g(t) &= (\frac{C_k}{3})^{\frac{3}{2}} - C_k(\frac{C_k}{3})^{\frac{1}{2}} + \frac{1}{4}\lambda_k \\ &= -\frac{2\sqrt{3}}{9}C_k^{\frac{3}{2}} + \frac{1}{4}\lambda_k \\ &\geq -\frac{2\sqrt{3}}{9}(\frac{3}{4}\lambda_k^{\frac{2}{3}})^{\frac{3}{2}} + \frac{1}{4}\lambda_k = 0 \end{aligned}$$

由此可得 $h_k^{'}$ 在 $t \in (0, +\infty)$ 上是单调增的，所以在 $\beta_k = 0$ 处取到最小值。

类似的，我们可以以以上方法得到 $C_k \leq -\frac{3}{4}\lambda_k^{\frac{2}{3}}$ 与 $C_k > -\frac{3}{4}\lambda_k^{\frac{2}{3}}$ 时候的情况

综上所述，当 $|C_k| \geq \frac{3}{4}\lambda_k^{\frac{2}{3}}$ 时，$\beta_k = t_0^2 = \frac{2}{3}(1 + cos(\frac{2\pi - 2\varphi}{3}))$；当 $|C_k| < \frac{3}{4}\lambda_k^{\frac{2}{3}}$ 时,$\beta_k = 0$

3.3　$L_{\frac{1}{2}}$ 算法收敛性分析 [1]

对于 $L_{\frac{1}{2}}$ 正则化的损失函数

$$\boldsymbol{\beta} = \underset{\boldsymbol{\beta}}{arg\,min} \left\{ \frac{1}{n}\sum_{i=1}^{n}(\mathbf{Y}_i - \mathbf{X}_i^T\boldsymbol{\beta})^2 + \lambda\sum_{i=1}^{p}|\beta_i|^{\frac{1}{2}} \right\}$$

我们将第一项与第二项分别记作

$$L_n(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{Y}_i - \mathbf{X}_i^T\boldsymbol{\beta})^2$$
$$\lambda P(\boldsymbol{\beta}) = \lambda\sum_{i=1}^{p}|\beta_i|^{\frac{1}{2}}$$

则有

$$R_n(\boldsymbol{\beta}) = L_n(\boldsymbol{\beta}) + \lambda P(\boldsymbol{\beta})$$

求 $\underset{\boldsymbol{\beta}}{arg\,min} \left\{ \frac{1}{n}\sum_{i=1}^{n}(\mathbf{Y}_i - \mathbf{X}_i^T\boldsymbol{\beta})^2 + \lambda\sum_{i=1}^{p}|\beta_i|^{\frac{1}{2}} \right\}$ 即为求 $R_n(\boldsymbol{\beta})$ 的稳定点，那么就有

$$\bigtriangledown L_n(\boldsymbol{\beta}^{t+1}) = -\bigtriangledown \lambda P(\boldsymbol{\beta}^t)$$

因为 $L_n(\boldsymbol{\beta})$ 是凸函数，所以具有如下性质:

$$\bigtriangledown L_n(\boldsymbol{\beta}^t) \geq \bigtriangledown L_n(\boldsymbol{\beta}^{t+1}) + (\boldsymbol{\beta}^t - \boldsymbol{\beta}^{t+1}) \bigtriangledown L_n(\boldsymbol{\beta}^{t+1}) \tag{3.3.1}$$

同理，由于 $\lambda P(\beta^t)$ 是凹函数，所以有如下性质:

$$\bigtriangledown \lambda P(\boldsymbol{\beta}^{t+1}) \leq \bigtriangledown \lambda P(\boldsymbol{\beta}^t) + (\boldsymbol{\beta}^{t+1} - \boldsymbol{\beta}^t) \bigtriangledown \lambda P(\boldsymbol{\beta}^t) \tag{3.3.2}$$

(3.3.1)+(3.3.2) 得

$$\bigtriangledown \lambda P(\boldsymbol{\beta}^{t+1}) + \bigtriangledown L_n(\boldsymbol{\beta}^{t+1}) + (\beta^t - \beta^{t+1}) \bigtriangledown L_n(\beta^{t+1}) \leq$$
$$\bigtriangledown \lambda P(\boldsymbol{\beta}^t) + (\boldsymbol{\beta}^{t+1} - \boldsymbol{\beta}^t) \bigtriangledown \lambda P(\boldsymbol{\beta}^t) + \bigtriangledown L_n(\boldsymbol{\beta}^t)$$

由于有 $\bigtriangledown L_n(\boldsymbol{\beta}^{t+1}) = -\bigtriangledown \lambda P(\boldsymbol{\beta}^t)$，故替换上式不等号左侧第三项可得

$$\bigtriangledown \lambda P(\boldsymbol{\beta}^{t+1}) + \bigtriangledown L_n(\boldsymbol{\beta}^{t+1}) + (-\beta^t + \beta^{t+1}) \bigtriangledown \lambda P(\boldsymbol{\beta}^t) \leq$$
$$\bigtriangledown \lambda P(\boldsymbol{\beta}^t) + (\boldsymbol{\beta}^{t+1} - \boldsymbol{\beta}^t) \bigtriangledown \lambda P(\boldsymbol{\beta}^t) + \bigtriangledown L_n(\boldsymbol{\beta}^t)$$

消去两边相同部分即有

$$\bigtriangledown \lambda P(\boldsymbol{\beta}^{t+1}) + \bigtriangledown L_n(\boldsymbol{\beta}^{t+1}) \leq \bigtriangledown \lambda P(\boldsymbol{\beta}^t) + \bigtriangledown L_n(\boldsymbol{\beta}^t)$$

即得到 $R_n(\boldsymbol{\beta}^{t+1}) \leq R_n(\boldsymbol{\beta}^t)$, 又因为 $R_n(\boldsymbol{\beta}^t) \geq 0$ 故 $R_n(\boldsymbol{\beta}^t)$ 随 t 的增长收敛到一个极小值点。

3.4 $L_{\frac{1}{2}}$ 正则算法 German Credit 数据数值模拟 [2]

在具体的 Logistic 算法中，$L_{\frac{1}{2}}$ 正则化的具体步骤如下：

Step1: 输入数据集 $\{\mathbf{x}_i, y_i\}_{i=1}^n$，初始化 $\boldsymbol{\beta} = (\beta_0, \beta_1, ..., \beta_p)$

Step2: 重复 Step3-Step4 至达到终止条件

Step3: 按照（3.2）介绍的坐标下降法求解最优的 $\boldsymbol{\beta}$

Step4: 更新 $\boldsymbol{\mu}$、$\mathbf{W}$、$\mathbf{z}$

Step5: 更新 $\boldsymbol{\beta}$

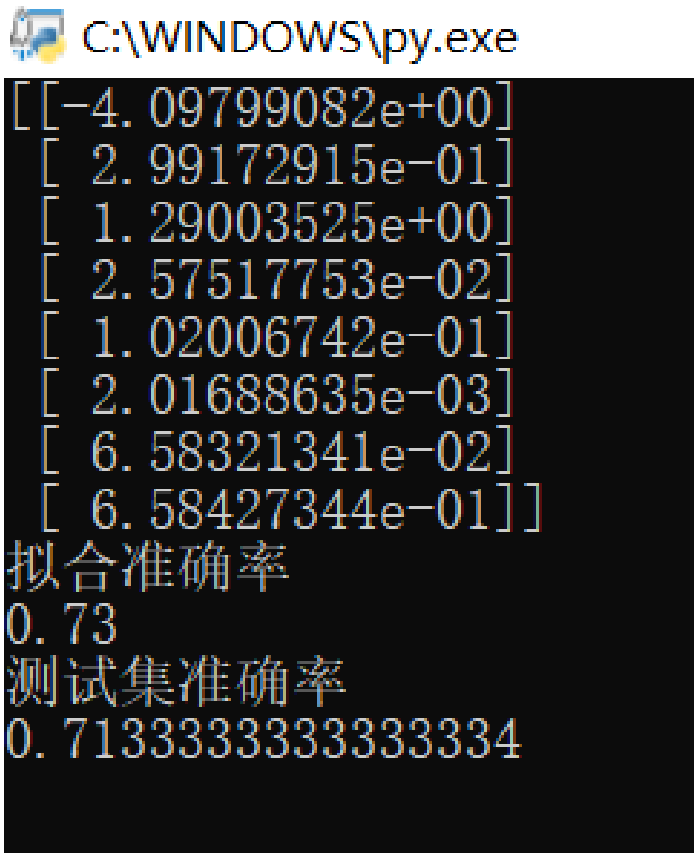
赵谦老师的文章里使用了 $L_{\frac{1}{2}}$ 正则化算法对 German Credit 数据库进行了训练以及测试，其测试结果如下（German Credit 数据库详情见附录）其测试的准确率在 0.8 左右。

表 2 German 数据实验结果

Table 2 Experimental results on German data

方法	测试误差	选择的特征个数
Logistic 回归	64/300	20
L_1 正则化 Logistic 回归	63/300	17
$L_{1/2}$ 正则化 Logistic 回归	58/300	16

我们在 UCI 数据库中找到了 German Credit 数据库并且运用上述算法，尝试复现赵谦老师的结果，最终复现结果如下 为了与赵谦老师的结果作对照，我们同样使用 German Credit 数据库（见附件），我们选择前 700 组数据作为训练集，后 300 组数据作为测试集，经过我们的测试，我们选择了对准确率提升效果最为显著的七个特征（为第 1、3、6、7、10、11、15 个特征）我测试得迭代 9 次得到的准确率最高，同时我们注意到，当迭代超过 25 次时会引发梯度爆炸使算法失效，我们最终得到的最佳权重（第一项为偏置）与训练集、测试集准确率如下：


C:\WINDOWS\py.exe
[[-4.09799082e+00]
[2.99172915e-01]
[1.29003525e+00]
[2.57517753e-02]
[1.02006742e-01]
[2.01688635e-03]
[6.58321341e-02]
[6.58427344e-01]]
拟合准确率
0.73
测试集准确率
0.7133333333333334

第 4 章　总结与思考

通过理论分析与数值模拟之后，我们确实感受到了 ADMM 算法与 $L_{\frac{1}{2}}$ 正则化在处理某些问题时较传统方法的不同之处。首先是 ADMM 算法，其特点与优势是它可以将一个复杂问题转换成几个较为简单的子问题，来加速问题的迭代速度与降低问题的复杂度，整体上可以减少问题的迭代次数并且让每一步迭代更易于处理。特别的是，这个分解的过程似乎是可以一直持续下去的，但是要注意，随着问题的分解，需要迭代的变量个数也会因此增加，在处理一些较为简单的问题时，迭代的次数可能会因此不减反增。然后是 $L_{\frac{1}{2}}$ 正则化，$L_{\frac{1}{2}}$ 正则化在处理某些稀疏优化问题时可能比传统的正则化子有着更好的优势，但是其是凹函数的性质注定了它的求解是较为困难的，就目前来看，$L_{\frac{1}{2}}$ 正则化使用坐标下降法的效率仍然是比较低的，若以后有更高效的算法来求解此类问题，相信 $L_{\frac{1}{2}}$ 正则化的前景还是美好的。

参考文献

[1] 常象宇 徐宗本 张海, 王尧. L1/2 正则化. 中国科学: 信息科学, 40(412-422), 2010.

[2] 孟德宇 徐宗本 赵谦. L1/2 正则化 logistic 回归. 模式识别与人工智能, 25(721-728), 2012.