

2.2 Logistic and SoftMax Regression

回忆上一章节，关于回归的概率模型 Probabilistic Learning Regression

对于训练集 $D = \{\mathbf{x}_i, y_i\}_{i=1}^N$ ，学习一个模型，根据给定输入得到预测输出
概率模型 $p(y|\mathbf{x}, \theta)$ 满足

$$p(y|\mathbf{x}, \theta) = N(y|\mu(\mathbf{x}), \sigma^2) = N(y|\mathbf{w}^T \mathbf{x}, \sigma^2)$$

因而给定一个输入 \mathbf{x} 时，可以得到一个 y 的高斯分布

为了得到 y 的点估计，可以使用均值，如

$$\hat{y} = \mu(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$$

逻辑回归

分类：对于训练集 $D = \{\mathbf{x}_i, y_i\}_{i=1}^N$ ，学习一个模型，根据给定输入得到所属类别
概率模型满足

$$p(y|\mathbf{x}, \mathbf{w}) = \text{Ber}(y|\sigma(\mathbf{w}^T \mathbf{x}))$$

其中 $\sigma(z)$ 为 Sigmoid/logistic/logit function 该函数将实域映射至 $[0,1]$ 范围内。

$$\sigma(z) = \frac{1}{1 + \exp(-z)} = \frac{e^z}{e^z + 1}$$

重写以上概率模型，有

$$p(y = 1|\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})}$$

从而

$$\log \frac{p(y = 1|\mathbf{x}, \mathbf{w})}{1 - p(y = 1|\mathbf{x}, \mathbf{w})} = \mathbf{w}^T \mathbf{x}$$

$\log \frac{p}{1-p}$ 被称作 p 的 logit。

判决条件为

$$\hat{y} = 1 \text{ iff } p(y = 1|\mathbf{x}, \mathbf{w}) > 0.5, \text{ otherwise } \hat{y} = 0$$

以上被称为 最佳贝叶斯分类器 Optimal Bayes Classifier。

解以下不等式

$$p(y = 1|\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})} > 0.5$$

可以得到

$$\mathbf{w}^T \mathbf{x} > 0$$

从而以上判决条件可以简化为

$$\hat{y} = 1 \text{ iff } \mathbf{w}^T \mathbf{x} > 0$$

因而实际上逻辑回归 Logistics Regression是一个线性分类器，其决策边界 decision boundary是 $y = \mathbf{w}^T \mathbf{x}$

学习方法：权重 \mathbf{w} 通过最小化 NLL 来决定

$$NLL(\mathbf{w}) = -\log p(D|\mathbf{w})$$

从以上可知，

$$p(y = 1|\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{w}^T \mathbf{x})$$

从而

$$p(y = 0|\mathbf{x}, \mathbf{w}) = 1 - \sigma(\mathbf{w}^T \mathbf{x})$$

将以上二式合并，有

$$p(y|\mathbf{x}, \mathbf{w}) = (\sigma(\mathbf{w}^T \mathbf{x}))^y (1 - \sigma(\mathbf{w}^T \mathbf{x}))^{1-y}$$

从而其对数似然函数为

$$\log p(y|\mathbf{x}, \mathbf{w}) = y \log \sigma(\mathbf{w}^T \mathbf{x}) + (1 - y) \log 1 - \sigma(\mathbf{w}^T \mathbf{x})$$

为了找到权重 \mathbf{w} 的最大似然估计 MLE，我们使以上函数有最大值，或者是使得负对数似然函数有最小值：

$$\begin{aligned} NLL(\mathbf{w}) &= -\log p(D|\mathbf{w}) = -\sum_{i=1}^N \log p(y_i|\mathbf{x}_i, \mathbf{w}) \\ &= -\sum_{i=1}^N [y_i \log \sigma(\mathbf{w}^T \mathbf{x}_i) + (1 - y_i) \log 1 - \sigma(\mathbf{w}^T \mathbf{x}_i)] \end{aligned}$$

不同于线性回归的是，以上的 NLL 无法给出解析解 closed-form solution，因而需要采用优化算法进行数值计算，得到一个近似的数值解 numerical solution。可用的方法有梯度下降以及牛顿方法。

梯度下降

对于一个标量 w 的函数 $J(w)$ ，其导数定义如下

$$J'(w) = \frac{dJ(w)}{dw} = \lim_{\epsilon \rightarrow 0} \frac{J(w + \epsilon) - J(w)}{\epsilon}$$

当 ϵ 很小时，取泰勒展开式前两项，有

$$J(w + \epsilon) \approx J(w) + \epsilon J'(w)$$

从而若要减小 $J(w)$ ，应该：

1. 若 $J'(w) > 0$ ，则使 ϵ 为负数，减小 w ;
2. 若 $J'(w) < 0$ ，则使 ϵ 为正数，增大 w ;

简而言之，就是让 w 往导数的反方向移动，即为梯度下降。

实施以上想法的做法是，利用以下公式 update w ：

$$w := w - \alpha J'(w)$$

以上更新公式当中的 $-\alpha J'(w)$ 代表向导数的反方向移动，其中 α 决定了每次移动

多少，在优化中称其为步长 step size，在 ML 中称其为学习率 learning rate。
 拓展以上定义，对于一个向量 \mathbf{w} 的函数 $J(\mathbf{w})$ ，其中 $\mathbf{w} = (w_0, w_1, w_2, \dots, w_D)^T$ ，
 其中函数 J 在点 \mathbf{w} 处的梯度定义为

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \left(\frac{dJ}{dw_0}, \frac{dJ}{dw_1}, \dots, \frac{dJ}{dw_N} \right)^T$$

梯度方向是函数增长得最快的方向，若想减小函数值，则按照梯度的反方向移动。最小化 $J(\mathbf{w})$ 的梯度下降方法有如下两个步骤：

1. 初始化权重 \mathbf{w}
2. 重复以下公式直至收敛

$$\mathbf{w} := \mathbf{w} - \alpha \nabla_{\mathbf{w}} J(\mathbf{w})$$

上式当中的学习率 α 常随着每次迭代有所不同。学习率过小，收敛速度太慢；
 学习率过大，会导致无法收敛。Line Search 是一个较好的选取学习率的方法。

我们将以上梯度下降的方法运用于一开始讨论的逻辑回归当中，需要计算 **NLL** 关于 \mathbf{w} 的偏导。在此之前，给出 **Sigmoid** 函数的导数如下：

$$\sigma'(z) = \sigma(z)(1 - \sigma(z))$$

令 $z_i = \mathbf{w}^T \mathbf{x}_i$ ，从而

$$\begin{aligned} \nabla_{\mathbf{w}} J(\mathbf{w}) &= \frac{\partial NLL(\mathbf{w})}{\partial w_j} = - \sum_{i=1}^N \left[y_i \frac{\partial \log \sigma(z_i)}{\partial w_j} + (1 - y_i) \frac{\partial \log(1 - \sigma(z_i))}{\partial w_j} \right] \\ &= - \sum_{i=1}^N \left[y_i \frac{1}{\sigma(z_i)} + (1 - y_i) \frac{1}{1 - \sigma(z_i)} \right] \frac{\partial \sigma(z_i)}{\partial w_j} \\ &= - \sum_{i=1}^N \left[y_i \frac{1}{\sigma(z_i)} + (1 - y_i) \frac{1}{1 - \sigma(z_i)} \right] \sigma(z_i)(1 - \sigma(z_i)) \frac{\partial z_i}{\partial w_j} \\ &= - \sum_{i=1}^N [y_i(1 - \sigma(z_i)) + (1 - y_i)\sigma(z_i)] x_{i,j} \\ &= - \sum_{i=1}^N [y_i - \sigma(z_i)] x_{i,j} \\ &= - \sum_{i=1}^N [y_i - \sigma(\mathbf{w}^T \mathbf{x})] x_{i,j} \end{aligned}$$

进而将上式代入更新公式当中，有

$$\mathbf{w} := \mathbf{w} - \alpha \nabla_{\mathbf{w}} J(\mathbf{w}) = \mathbf{w} + \alpha \sum_{i=1}^N [y_i - \sigma(\mathbf{w}^T \mathbf{x})] x_{i,j}$$

采用上述更新权重公式的方法称为批量梯度下降方法 Batch Gradient Descent，其更新方法为：

1. 若 $y_i - \sigma(\mathbf{w}^T \mathbf{x}) > 0$ ，即预测值 $\sigma(\mathbf{w}^T \mathbf{x})$ 小于实际值 y_i ，则增加 \mathbf{w} ;
2. 若 $y_i - \sigma(\mathbf{w}^T \mathbf{x}) < 0$ ，即预测值 $\sigma(\mathbf{w}^T \mathbf{x})$ 大于实际值 y_i ，则减小 \mathbf{w} ;

批量梯度下降方法在 N 即样本量非常大时，计算的 cost 非常大。此时我们采用随机梯度下降法 Stochastic Gradient Descent，其权值更新方法为

1. 初始化权重 \mathbf{w}
2. 随机选取样本 D 中的一个子集 B ，即 $B \subset \{1, 2, \dots, N\}$ ，重复以下公式直至收敛

$$\mathbf{w} := \mathbf{w} - \alpha \nabla_{\mathbf{w}} J(\mathbf{w}) = \mathbf{w} + \alpha \sum_{i \in B} [y_i - \sigma(\mathbf{w}^T \mathbf{x})] x_{i,j}$$

以上随机选取的样本子集 B 被称为 **minibatch**，其大小称为 **batch size**。上一章节当中，为了避免过拟合，我们倾向于使用脊回归更多于线性回归。同样的，我们需要在逻辑回归的目标函数当中加入 **L2 正则项** 来避免过拟合：

$$J(\mathbf{w}) = \text{NLL}(\mathbf{w}) + \frac{1}{2} \lambda \|\mathbf{w}\|_2^2$$

从而偏导变成

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = - \sum_{i=1}^N [y_i - \sigma(\mathbf{w}^T \mathbf{x})] x_{i,j} + \lambda w_j$$

权值更新公式变为

$$\mathbf{w} := \mathbf{w} - \alpha \nabla_{\mathbf{w}} J(\mathbf{w}) = \mathbf{w} + \alpha [-\lambda w_j + \sum_{i \in B} (y_i - \sigma(\mathbf{w}^T \mathbf{x})) x_{i,j}]$$

对比未加上正则化的公式，由于 $0 < \alpha \lambda < 1$ ，正则化使得权重变小：

$$|(1 - \alpha \lambda) w_j| < |w_j|$$

牛顿方法

对于一个标量 w 的函数 $J(w)$ ，其最小值处导数值 $J'(w) = 0$ 。

令 $f(w) = J'(w)$ ，求 $J(w)$ 最小值的点可以转化为求 $f(w) = 0$ 的点。

牛顿方法 Newton's Method 在此基础上使用以下方法更新权重：

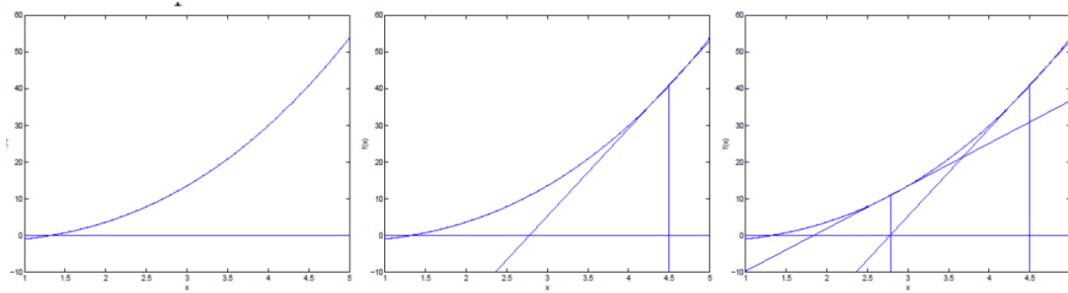
1. 初始化权重 w

2. 重复以下公式直至收敛

$$w := w - \frac{f(w)}{f'(w)}$$

解释：这种方法根据现在的权重值 w_0 ，采用函数的正切 tangent 来逼近函数值：

$$y = f'(w_0)w + f(w_0) - f'(w_0)w_0$$



使以上方程为 0，得到下一个权重值 w

$$f'(w_0)w + f(w_0) - f'(w_0)w_0 = 0$$

得到

$$w = w_0 - \frac{f(w_0)}{f'(w_0)}$$

因为 $f(w) = J'(w)$ ，从而以上的权值更新公式变为

$$w := w - \frac{J'(w)}{J''(w)}$$

拓展以上定义，对于一个向量 \mathbf{w} 的函数 $J(\mathbf{w})$ ，其中 $\mathbf{w} = (w_0, w_1, w_2, \dots, w_N)^T$ ，其中函数的一阶导数，即为函数 J 在点 \mathbf{w} 处的梯度：

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \left(\frac{dJ}{dw_0}, \frac{dJ}{dw_1}, \dots, \frac{dJ}{dw_N} \right)^T$$

函数的二阶导数，是为海森矩阵 Hessian matrix：

$$\mathbf{H} = [H_{ij}], \quad H_{ij} = \frac{\partial^2 J(\mathbf{w})}{\partial w_i \partial w_j}$$

从而权值更新公式为

$$\mathbf{w} := \mathbf{w} - \frac{\nabla_{\mathbf{w}} J(\mathbf{w})}{\mathbf{H}} = \mathbf{w} - \mathbf{H}^{-1} \nabla_{\mathbf{w}} J(\mathbf{w})$$

以上公式也可以从另外一个角度推得。对于一个标量 w 的函数 $J(w)$ ，其导数定义如下

$$J'(w) = \frac{dJ(w)}{dw} = \lim_{\epsilon \rightarrow 0} \frac{J(w + \epsilon) - J(w)}{\epsilon}$$

取泰勒展开式前三项，有

$$J(w + \epsilon) \approx J(w) + \epsilon J'(w) + \frac{1}{2} J''(w) \epsilon^2$$

我们想令 $\frac{dJ(w+\epsilon)}{d\epsilon} = 0$, 从而

$$J'(w) + J''(w)\epsilon = 0$$

可以得到

$$\epsilon = -\frac{J'(w)}{J''(w)}$$

梯度下降 vs 牛顿方法

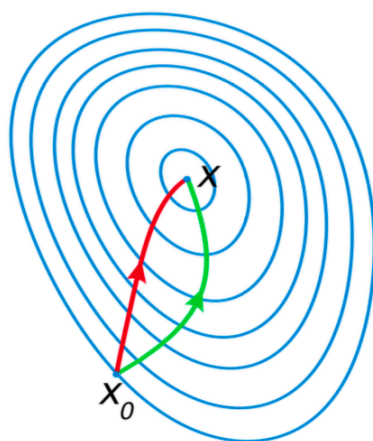
在梯度下降方法当中，我们仅有一次项参与：

$$J(w + \epsilon) \approx J(w) + \epsilon J'(w)$$

牛顿方法当中，我们有二次项参与：

$$J(w + \epsilon) \approx J(w) + \epsilon J'(w) + \frac{1}{2} J''(w) \epsilon^2$$

这种差别使得牛顿方法在其他条件相同时有更快的收敛速度，或者是更少的迭代次数；但与此同时，牛顿方法每一步所需时间要比梯度下降方法更多。下图红线代表牛顿方法，绿线代表梯度下降方法。



SoftMax 方法

以上讨论的逻辑回归是一个二分类模型，当分类的种类多于两种时，引入多分类逻辑回归 **Multi-Class Logistic Regression**，又称为 **SoftMax Regression**.

分类：对于训练集 $D = \{x_i, y_i\}_{i=1}^N$ ，其中 $y_i \in \{1, 2, \dots, C\}, C \geq 2$. 学习一个模型，根据给定输入得到所属类别。

其概率模型如下：

1. 对于每一个分类 c ，其权重向量为 $\mathbf{w}_c = (w_{c,0}, w_{c,1}, w_{c,2}, \dots, w_{c,N})^T$ ，从而权重矩阵 $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, \dots, \mathbf{w}_C)^T$
2. 给定 \mathbf{x} ，其属于类别 c 的概率是

$$p(y = c | \mathbf{x}, \mathbf{W}) = \frac{\exp(\mathbf{w}_c^T \mathbf{x})}{\sum_{c'=1}^C \exp(\mathbf{w}_{c'}^T \mathbf{x})}$$

以上方程可以保证给定 \mathbf{x} 所属各个类别的概率之和等于 1，称之为 partition function.

3. 判决条件为：

$$\hat{y} = \arg \max_y p(y | \mathbf{x}, \mathbf{W})$$

当 $C = 2$ 时，将两个分类 1,2 重命名为 0,1，则 SoftMax 退化为逻辑回归。

为了取得 \mathbf{W} 的最大似然估计，需要最小化其负对数似然函数 NLL：

$$\begin{aligned} \text{NLL}(\mathbf{w}) &= -\log p(D | \mathbf{w}) = -\sum_{i=1}^N \log p(y_i | \mathbf{x}_i, \mathbf{w}) \\ &= -\sum_{i=1}^N \sum_{c=1}^C \mathbb{I}(y_i = c) \log p(y_i = c | \mathbf{x}_i, \mathbf{w}) \\ &= -\sum_{i=1}^N \sum_{c=1}^C \mathbb{I}(y_i = c) \log \frac{\exp(\mathbf{w}_c^T \mathbf{x}_i)}{\sum_{c'=1}^C \exp(\mathbf{w}_{c'}^T \mathbf{x}_i)} \\ &= -\sum_{i=1}^N \sum_{c=1}^C \mathbb{I}(y_i = c) \left[\mathbf{w}_c^T \mathbf{x}_i - \log \sum_{c'=1}^C \exp(\mathbf{w}_{c'}^T \mathbf{x}_i) \right] \\ &= -\sum_{i=1}^N \left[\sum_{c=1}^C \mathbb{I}(y_i = c) \mathbf{w}_c^T \mathbf{x}_i - \log \sum_{c'=1}^C \exp(\mathbf{w}_{c'}^T \mathbf{x}_i) \right] \end{aligned}$$

其中 $\mathbb{I}(y_i = c)$ 为零一函数， $y_i = c$ 时取值为 1，否则取 0.

以上 NLL 对 $w_{c,j}$ 的偏导为：

$$\frac{\partial \text{NLL}(\mathbf{w})}{\partial w_{c,j}} = -\sum_{i=1}^N [\mathbb{I}(y_i = c) - p(y_i = c | \mathbf{x}_i, \mathbf{W})] x_{i,j}$$

$C = 2, c = 1$ 时退化为

$$\frac{\partial \text{NLL}(\mathbf{w})}{\partial w_j} = -\sum_{i=1}^N [y_i - \sigma(\mathbf{w}^T \mathbf{x})] x_{i,j}$$

即为逻辑回归的梯度。

从而 SoftMax 的 NLL 对于每个分类 c 的权重向量 \mathbf{w}_c 的梯度为

$$\begin{aligned}\nabla_{\mathbf{w}} NLL(\mathbf{w}_c) &= \left(\frac{\partial NLL(\mathbf{w}_c)}{\partial w_{c,0}}, \frac{\partial NLL(\mathbf{w}_c)}{\partial w_{c,1}}, \dots, \frac{\partial NLL(\mathbf{w}_c)}{\partial w_{c,N}} \right)^T \\ &= - \sum_{i=1}^N [\mathbb{I}(y_i = c) - p(y_i = c | \mathbf{x}_i, \mathbf{W})] \mathbf{x}_i\end{aligned}$$

进而其权值更新公式为：

$$\mathbf{w}_c := \mathbf{w}_c + \alpha \sum_{i=1}^N [\mathbb{I}(y_i = c) - p(y_i = c | \mathbf{x}_i, \mathbf{W})] \mathbf{x}_i$$

重复以上公式直至收敛。

线性分类器及其优化方法

考虑二分类问题的概率模型：对于训练集 $D = \{\mathbf{x}_i, y_i\}_{i=1}^N$ ，学习一个模型，根据给定输入得到所属类别

概率模型满足

$$p(y|\mathbf{x}, \mathbf{w}) = \text{Ber}(y|\sigma(\mathbf{w}^T \mathbf{x}))$$

学习方法：权重 \mathbf{w} 通过最小化 NLL 来决定

$$NLL(\mathbf{w}) = -\log p(D|\mathbf{w})$$

判决条件为

$$\hat{y} = 1 \text{ iff } p(y = 1|\mathbf{x}, \mathbf{w}) > 0.5, \text{ otherwise } \hat{y} = 0$$

上文提到实际上逻辑回归 **Logistics Regression** 是一个线性分类器。

给定一个数据集 $D = \{\mathbf{x}_i, y_i\}_{i=1}^N$ ，其中 $y_i \in \{-1, 1\}$ ，学习一个线性分类器 **linear classifier**:

$$\hat{y} = \text{sign}(\mathbf{w}^T \mathbf{x} + b)$$

其中 $\mathbf{x} = (x_1, x_2, \dots, x_N)^T$, $\mathbf{w} = (w_1, w_2, \dots, w_N)^T$ ；这里 drop 掉了 $x_0 = 1$ ，同时采用 b 来代替 w_0 ；同时，sign 函数定义如下：

$$\text{sign}(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases}$$

我们通过最小化训练误差 training error $L(\mathbf{w}, b)$ 确定参数 \mathbf{w} 和 b :

$$L(\mathbf{w}, b) = \sum_{i=1}^N L_{0/1}(y_i, \hat{y}_i)$$

其中 $L_{0/1}(y_i, \hat{y}_i) = \mathbb{I}(y_i \neq \hat{y}_i)$ 称为零一损失函数 zero/one loss function

令 $z = \mathbf{w}^T \mathbf{x} + b$ ，则有

$$L_{0/1}(y, \hat{y}) = \mathbb{I}(y(\mathbf{w}^T \mathbf{x} + b) \leq 0) = \mathbb{I}(yz \leq 0)$$

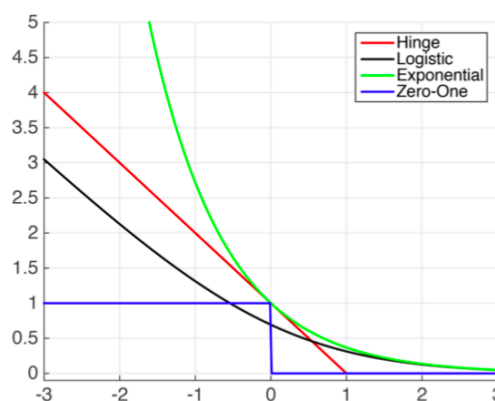
从而 $L_{0/1}(y, \hat{y})$ 也可写作 $L_{0/1}(y, z)$ 。因此线性分类器的损失函数常写作：

$$L(\mathbf{w}, b) = \sum_{i=1}^N \mathbb{I}(y_i(\mathbf{w}^T \mathbf{x}_i + b) \leq 0)$$

以上的关于 \mathbf{w} 和 b 的零一损失函数因为不是凸函数 convex function，所以不好优化；因而我们采用一系列凸函数来拟合零一损失函数，进而来优化它，拟合函数产生的误差称之为代理误差 Surrogate Loss，拟合的函数称之为代理损失函数 Surrogate Loss Function；代理误差必须是真正误差函数的上确界 upper bound，从而保证在减小代理误差的同时，也在减小实际误差。

常用的代理误差函数有：

- **Zero/one loss:** $L_{0/1}(y, z) = \mathbb{I}(yz \leq 0)$.
- **Logistic loss:** $L_{\log}(y, z) = \frac{1}{\log 2} \log(1 + \exp(-yz))$
- **Hinge loss:** $L_{\text{hin}}(y, z) = \max\{0, 1 - yz\}$
- **Exponential loss:** $L_{\text{exp}}(y, z) = \exp(-yz)$
- **Squared loss:** $L_{\text{sqr}}(y, z) = (y - z)^2$



当采用代理误差函数时，有两个指标判别模型在训练集上的表现：**training loss** 和 **training error**；两个指标判别模型在测试集上的表现：**test loss** 和 **test error**。类似于线性回归，**test loss** 和 **test error** 可以通过正则化方法实现。