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

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

因而给定一个输入 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

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

逻辑回归

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

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

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

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

从而

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

$\log\frac{p}{1-p}$被称作 p 的 logit。

判决条件为

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

以上被称为最佳贝叶斯分类器 Optimal Bayes Classifier.
解以下不等式

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

可以得到

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

从而以上判决条件可以简化为

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

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$y=\boldsymbol{w}^T\boldsymbol{x}$

学习方法：权重 w 通过最小化 NLL 来决定

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

从以上可知，

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

从而

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

将以上二式合并，有

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

从而其对数似然函数为

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

为了找到权重 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：

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

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

$$J'(w)=\frac{\mathrm{d}J(w)}{\mathrm{d}w}=\lim_{\epsilon\to 0}\frac{J(w+\epsilon)-J(w)}{\epsilon}$$

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：

$$w\coloneqq w-\alpha J'(w)$$

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 w 的函数$J(\boldsymbol{w})$，其中$\boldsymbol{w}=(w_0,w_1,w_2,\dots,w_D)^T$，其中函数$J$在点$\boldsymbol{w}$处的梯度定义为

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

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$J(\boldsymbol{w})$的梯度下降方法有如下两个步骤：

1. 初始化权重$\boldsymbol{w}$
2. 重复以下公式直至收敛

$$\boldsymbol{w}\coloneqq\boldsymbol{w}-\alpha\nabla_{\boldsymbol{w}}J(\boldsymbol{w})$$

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

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

进而将上式代入更新公式当中，有

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

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

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

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：

$$J(\boldsymbol{w}) = \mathrm{NLL}(\boldsymbol{w}) + \frac{1}{2}\lambda\big||\boldsymbol{w}|\big|_2^2$$

从而偏导变成

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

权值更新公式变为

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

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：

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

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：

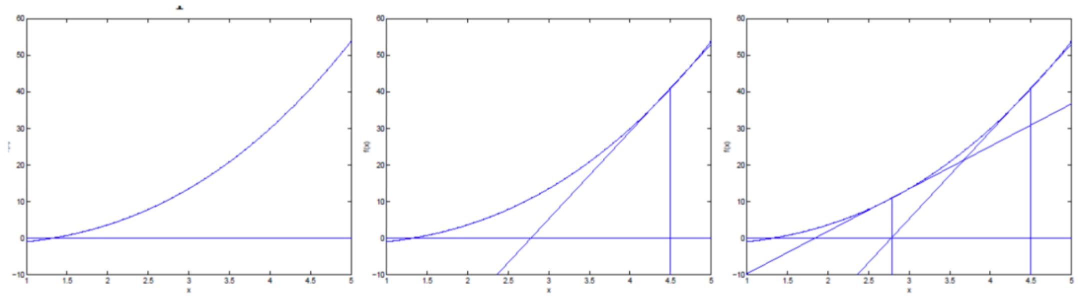
1. 初始化权重 w

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 w 的函数$J(\boldsymbol{w})$，其中$\boldsymbol{w} = (w_0, w_1, w_2, \ldots, w_N)^T$，其中函数的一阶导数，即为函数$J$在点$\boldsymbol{w}$处的梯度：

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

函数的二阶导数，是为海森矩阵 Hessian matrix：

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

从而权值更新公式为

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

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

$$J'(w) = \frac{\mathrm{d}J(w)}{\mathrm{d}w} = \lim_{\epsilon \to 0} \frac{J(w+\epsilon) - J(w)}{\epsilon}$$

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$\frac{\mathrm{d}J(w+\epsilon)}{\mathrm{d}\epsilon} = 0$，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

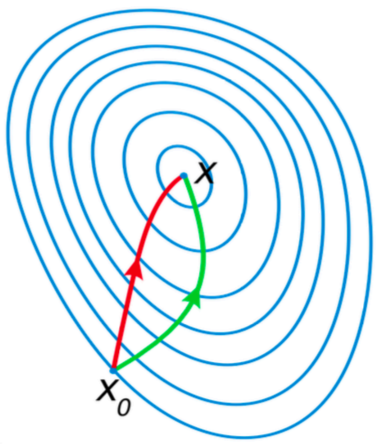
在梯度下降方法当中，我们仅有一次项参与：

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

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$D = \{\boldsymbol{x}_i, y_i\}_{i=1}^N$，其中$y_i \in \{1,2,\dots,C\}$, C ≥ 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$\boldsymbol{w}_c = \left(w_{c,0}, w_{c,1}, w_{c,2}, \ldots, w_{c,N}\right)^T$，从而权重矩阵$\boldsymbol{W} = (\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3, \ldots, \boldsymbol{w}_C)^T$
2. 给定$\boldsymbol{x}$，其属于类别 c 的概率是

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

   以上方程可以保证给定$\boldsymbol{x}$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：

$$\hat{y} = \arg\max_{y} p(y|\boldsymbol{x}, \boldsymbol{W})$$

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 W 的最大似然估计，需要最小化其负对数似然函数 NLL：

$$\mathrm{NLL}(\boldsymbol{w}) = -\log p(D|\boldsymbol{w}) = -\Sigma_{i=1}^{N} \log p(y_i|\boldsymbol{x}_i, \boldsymbol{w})$$

$$= -\sum_{i=1}^{N}\sum_{c=1}^{C} \mathbb{I}(y_i = c) \log p(y_i = c|\boldsymbol{x}_i, \boldsymbol{w})$$

$$= -\sum_{i=1}^{N}\sum_{c=1}^{C} \mathbb{I}(y_i = c) \log \frac{\exp(\boldsymbol{w}_c^T \boldsymbol{x_i})}{\Sigma_{c'=1}^{C} \exp\left(\boldsymbol{w}_{c'}^T \boldsymbol{x_i}\right)}$$

$$= -\sum_{i=1}^{N}\sum_{c=1}^{C} \mathbb{I}(y_i = c) \left[\boldsymbol{w}_c^T \boldsymbol{x_i} - \log \sum_{c'=1}^{C} \exp\left(\boldsymbol{w}_{c'}^T \boldsymbol{x_i}\right)\right]$$

$$= -\sum_{i=1}^{N}[\sum_{c=1}^{C} \mathbb{I}(y_i = c) \boldsymbol{w}_c^T \boldsymbol{x_i} - \log \sum_{c'=1}^{C} \exp\left(\boldsymbol{w}_{c'}^T \boldsymbol{x_i}\right)]$$

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$w_{\mathrm{c},j}$的偏导为：

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

$C = 2, c = 1$时退化为

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

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$\boldsymbol{w}_c$的梯度为

$$\nabla_{\boldsymbol{w}} NLL(\boldsymbol{w}_c) = \left(\frac{\partial NLL(\boldsymbol{w}_c)}{\partial w_{c,0}}, \frac{\partial NLL(\boldsymbol{w}_c)}{\partial w_{c,1}}, \dots, \frac{\partial NLL(\boldsymbol{w}_c)}{\partial w_{c,N}}\right)^T$$

$$= -\sum_{i=1}^{N} [\mathbb{I}(y_i = c) - p(y_i = c|\boldsymbol{x}_i, \boldsymbol{W})]\, \boldsymbol{x}_i$$

进而其权值更新公式为：

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

重复以上公式直至收敛。

线性分类器及其优化方法：凸优化问题

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

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

学习方法：权重 w 通过最小化 NLL 来决定

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

判决条件为

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

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$D = \{\boldsymbol{x}_i, y_i\}_{i=1}^{N}$，其中$y_i \in \{-1,1\}$，学习一个线性分类器 linear classifier:

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

其中$\boldsymbol{x} = (x_1, x_2, \dots, x_N)^T, \boldsymbol{w} = (w_1, w_2, \dots, w_N)^T$；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 $L(\boldsymbol{w}, b)$确定参数$\boldsymbol{w}$和b:

$$L(\boldsymbol{w}, b) = \sum_{i=1}^{N} L_{0/1}(y_i, \widehat{y_i})$$

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

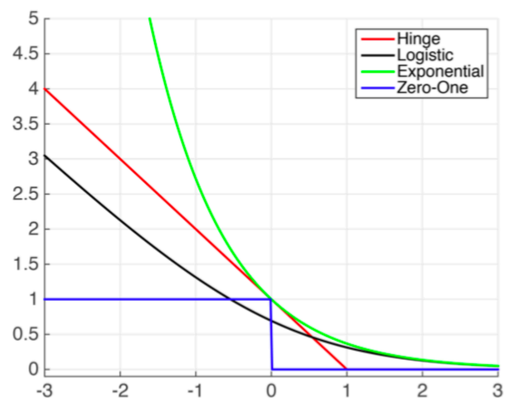
$$L_{0/1}(y,\hat{y})=\mathbb{I}(y(\boldsymbol{w}^T\boldsymbol{x}+b)\leq 0)=\mathbb{I}(yz\leq 0)$$

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：

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

以上的关于$\boldsymbol{w}$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**: $L_{0/1}(y,z)=\mathbb{1}(yz\leq 0)$.
- **Logistic loss**: $L_{log}(y,z)=\frac{1}{\log 2}\log(1+\exp(-yz))$
- **Hinge loss**: $L_{hin}(y,z)=\max\{0,1-yz\}$
- **Exponential loss**: $L_{exp}(y,z)=\exp(-yz)$
- **Squared loss**: $L_{sqr}(y,z)=(y-z)^2$



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。