机器学习算法系列（24）：机器学习中的损失函数

损失函数（loss function）是用来估量模型的预测值f(x)与真实值Y不一致的程度，它是一个非负实数值函数，通常使用$L(Y,f(x))$来表示，损失函数越小，模型的鲁棒性就越好。损失函数是经验风险函数的核心部分，也是结构风险函数的重要组成部分。模型的结构风险函数包括了经验风险项和正则项，通常可以表示成如下的式子：

$$\theta^* = argmin_\theta \frac{1}{N}\sum_{i=1}^{N} L(y_i,f(x_i;\theta)) + \lambda\Phi(\theta)$$

前面的均值函数表示的是经验风险函数，L代表的是损失函数，后面的Φ是正则化项（regularizer）或者叫惩罚项（penalty term）,它可以是L_1，也可以是L_2等其他的正则函数。整个式子表示的意思是找到使目标函数最小时的θ值。下面列出集中常见的损失函数。

一、对数损失函数（逻辑回归）

有些人可能觉得逻辑回归的损失函数就是平方损失，其实并不是。平方损失函数可以通过线性回归在假设样本是高斯分布的条件下推导得到，而逻辑回归得到的并不是平方损失。在逻辑回归的推导中，它假设样本服从伯努利分布（0-1分布），然后求得满足该分布的似然函数，接着取对数求极值等等。而逻辑回归并没有求似然函数的极值，而是把极大化当做是一种思想，进而推导出它的经验风险函数为：最小化负的似然函数（即$max F(y,f(x)) \rightarrow min - F(y,f(x))$）。从损失函数的视角来看，它就成了log损失函数了。

Log损失函数的标准形式：

$$L(Y,P(Y|X)) = -logP(Y|X)$$

刚刚说到，取对数是为了方便计算极大似然估计，因为在MLE中，直接求导比较困难，所以通常都是先取对数再求导找极值点。损失函数$L(Y.P(Y|X))$表达的是样本在分类Y的情况下，使概率$P(Y|X)$达到最大值（换言之，就是利用已知的样本分布，找到最有可能（即最大概率）导致这种分布的参数值；或者什么样的参数才能使我们观测到目前这组数据的概率最大）。因为log函数是单调递增的，所以$logP(Y|X)$也会达到最大值，因此在前面加上负号之后，最大化$P(Y|X)$就等价于最小化L了。

logistic回归的$P(y|x)$表达式如下（为了将类别标签y统一为1和0，下面将表达式分开表示）：

$$P\left(Y=y^{(i)}\,|\,x^{(i)};\theta\right)=\begin{cases}h_\theta\left(x^{(i)}\right)=\frac{1}{1+e^{-\theta^T x}},\ y^{(i)}=1\\ 1-h_\theta\left(x^{(i)}\right)=\frac{e^{-\theta^T x}}{1+e^{-\theta^T x}},\ y^{(i)}=0\end{cases}$$

将上面的公式合并在一起，可得到第i个样本正确预测的概率：

$$P(y^{(i)}\,|\,x^{(i)};\theta)=(h_\theta(x^{(i)}))^{y(i)}\cdot(1-h_\theta(x^{(i)}))^{1-y(i)}$$

上式是对一个样本进行建模的数据表达。对于所有的样本，假设每条样本生成过程独立，在整个样本空间中（N个样本）的概率分布为：

$$P(Y\mid X;\theta)=\prod_{i=1}^{N}\left(\left(h_\theta\left(x^{(i)}\right)\right)^{y^{(i)}}\left(1-h_\theta\left(x^{(i)}\right)\right)^{1-y^{(i)}}\right)$$

将上式代入到对数损失函数中，得到最终的损失函数为：

$$J(\theta)=-\frac{1}{N}\sum_{i=1}^{N}y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right)+\left(1-y^{(i)}\right)\log\left(1-h_\theta\left(x^{(i)}\right)\right)$$

之所以有人认为逻辑回归是平方损失，是因为在使用梯度下降来求最优解的时候，它的迭代式子与平方损失求导后的式子非常相似，从而给人一种直观上的错觉。

二、平方损失函数（最小二乘法，Ordinary Least Squares）

最小二乘法是线性回归的一种，OLS将问题转化成了一个凸优化问题。在线性回归中，它假设样本和噪声都服从高斯分布（为什么假设成高斯分布呢？其实这里隐藏了一个小知识点，就是中心极限定理，可以参考【central limit theorem】），最后通过极大似然估计（MLE）可以推导出最小二乘式子。最小二乘的基本原则是：最优拟合直线应该是使各点到回归直线的距离和最小的直线，即平方和最小。换言之，OLS是基于距离的，而这个距离就是我们用的最多的欧几里得距离。为什么它会选择使用欧式距离作为误差度量呢（即Mean squared error， MSE），主要有以下几个原因：

- 简单，计算方便；

- 欧氏距离是一种很好的相似性度量标准；
- 在不同的表示域变换后特征性质不变。

平方损失（Square loss）的标准形式如下：

$$L(Y,f(X)) = (Y - f(x))^2$$

当样本个数为n时，此时的损失函数变为：

$$L(Y,f(X)) = \sum_{i=1}^{n}(Y - f(X))^2$$

$Y - f(X)$表示的是残差，整个式子表示的是残差的平方和，而我们的目的就是最小化这个目标函数值（注：该式子未加入正则项），也就是最小化残差的平方和（residual sum of squares，RSS）。

而在实际应用中，通常会使用均方差（MSE）作为一项衡量指标，公式如下：

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\tilde{Y_i} - Y_i)^2$$

上面提到了线性回归，这里额外补充一句，我们通常说的线性有两种情况，一种是因变量y是自变量x的线性函数，一种是因变量y是参数αα的线性函数。在机器学习中，通常指的都是后一种情况。

三、指数损失函数（Adaboost）

学过Adaboost算法的人都知道，它是前向分步加法算法的特例，是一个加和模型，损失函数就是指数函数。在Adaboost中，经过m此迭代之后，可以得到$f_m(x)$:

$$f_m(x) = f_{m-1}(x) + a_m G_m(x)$$

Adaboost每次迭代时的目的是为了找到最小化下列式子时的参数a和G：

$$argmina,G = \sum_{i=1}^{N} exp[-y_i(f_{m-1}(x_i) + aG(x_i))]$$

而指数损失函数(exp-loss）的标准形式如下：

$$L(y,f(x)) = exp[-yf(x)]$$

可以看出，Adaboost的目标式子就是指数损失，在给定N个样本的情况下，Adaboost的损失函数为：

$$L(y,f(x)) = \frac{1}{N}\sum_{i=1}^{n} exp[-y_i f(x_i)]$$

四、Hinge损失函数（SVM）

4.1 Hinge损失函数（SVM）

线性支持向量机学习除了原始最优化问题，还有另外一种解释，就是最优化以下目标函数：

$$\sum_{i}^{N}[1-y_i(w\cdot x_i+b)]_+ + \lambda||w||^2$$

目标函数的第一项是经验损失或经验风险，函数

$$L(y\cdot(w\cdot x+b)) = [1-y(w\cdot x+b)]_+$$

称为合页损失函数（hinge loss function）。下标"+"表示以下取正值的函数：

$$[z]_+ = \begin{cases} z, & z>0 \\ 0, & z\le 0 \end{cases}$$

这就是说，当样本点(x_i,y_i)被正确分类且函数间隔（确信度）$y_i(w\cdot x_i+b)$大于1时，损失是0，否则损失是$1-y_i(w\cdot x_i+b)$。目标函数的第二项是系数为λ的w的L_2范数，是正则化项。

接下来证明线性支持向量机原始最优化问题：

$$\min_{w,b,\xi}\frac{1}{2}||w||^2 + C\sum_{i=1}^{N}\xi_i$$

$$s.t.\ \ y_i\left(w\cdot x_i+b\right)\geqslant 1-\xi_i,\ i=1,2,\cdots,N$$

$$\xi_i\geqslant 0,\ i=1,2,\cdots,\ N$$

等价于最优化问题

$$min_{w,b}\sum_{i}^{N}[1-y_i(w\cdot x_i+b)]_+ + \lambda||w||^2$$

先令$[1-y_i(w\cdot x_i+b)]_+ = \xi_i$，则$\xi_i\ge 0$，第二个约束条件成立；由$[1-y_i(w\cdot x_i+b)]_+ = \xi_i$，当$1-y_i(w\cdot x_i+b)>0$时，有$y_i(w\cdot x_i+b) = 1-\xi_i$;当$1-y_i(w\cdot x_i+b)\le 0$时，$\xi_i = 0$，有

$y_i(w \cdot x_i + b) \geq 1 - \xi_i$，所以第一个约束条件成立。所以两个约束条件都满足，最优化问题可以写作

$$min_{w,b} \sum_{i=1}^{N} \xi_i + \lambda ||w||^2$$

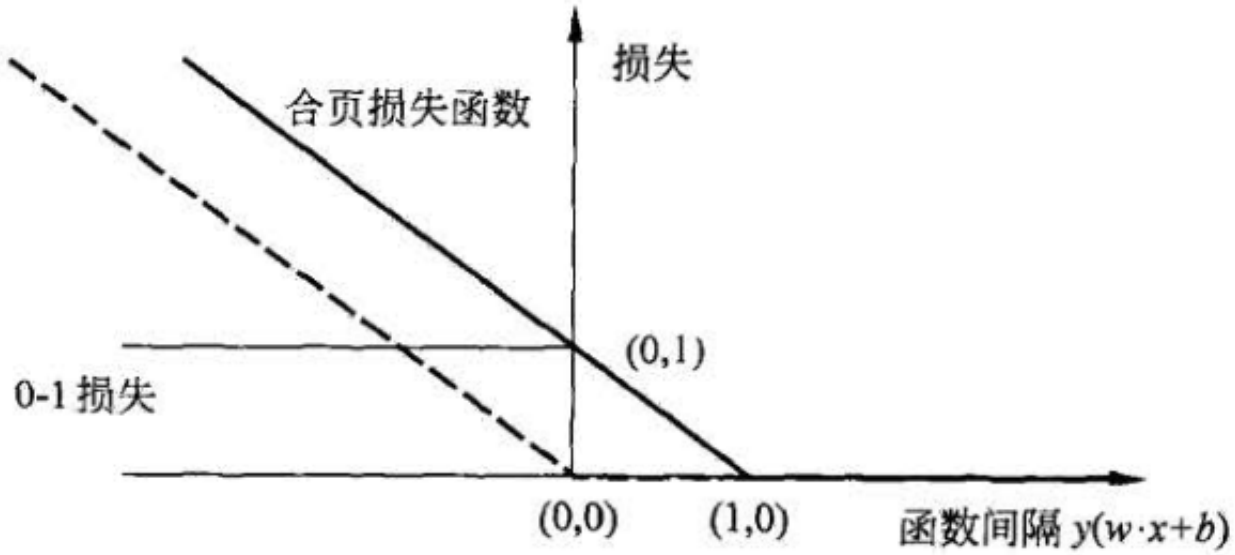
若取$\lambda = \frac{1}{2C}$则

$$min_{w,b} \frac{1}{C}(\frac{1}{2}||w||^2 + C\sum_{i=1}^{N} \xi_i)$$

与原始最优化问题等价。

合页损失函数图像如图所示，横轴是函数间隔$y(w \cdot x + b)$，纵轴是损失。由于函数形状像一个合页，故名合页损失函数。

图中还画出了0-1损失函数，可以认为它是一个二类分类问题的真正的损失函数，而合页损失函数是0-1损失函数的上界。由于0-1损失函数不是连续可导的，直接优化其构成的目标函数比较困难，可以认为线性支持向量机是优化由0-1损失函数的上界（合页损失函数）构成的目标函数。这时的上界损失函数又称为代理损失函数（surrogate function）。



图中虚线显示的是感知机的损失函数$[-y_i(w \cdot x_i + b)]_+$。这时当样本点$(x_i, y_i)$被正确分类时，损失是0，否则损失是$-y_i(w \cdot x_i + b)$，相比之下，合页损失函数不仅要分类正确，而且确信度足够高时损失才是0，也就是说，合页损失函数对学习有更高的要求

4.2 逻辑斯谛回归和SVM的损失函数对比

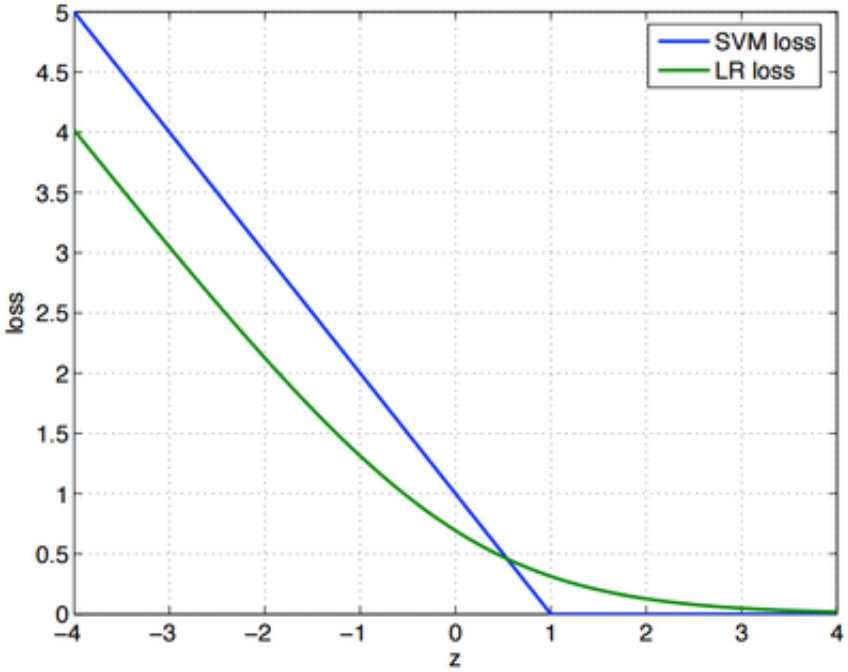
我们先来看一下带松弛变量的 SVM 和正则化的逻辑回归它们的损失函数：

$$\text{SVM:} \frac{1}{n}\sum_{i=1}^{n}(1-y_i[w_0+\mathbf{x}_i^T\mathbf{w}_1])^+ + \lambda\|\mathbf{w}_1\|/2 \quad (1)$$

$$\text{Logistic:} \frac{1}{n}\sum_{i=1}^{n}\overbrace{-\log g(y_i[w_0+\mathbf{x}_i^T\mathbf{w}_1])}^{-\log P(y_i|\mathbf{x},\mathbf{W})} + \lambda\|\mathbf{w}_1\|/2 \quad (2)$$

其中 $g(z) = (1 + exp(-z))^{-1}$
可以将两者统一起来:

$$\text{Both:} \frac{1}{n}\sum_{i=1}^{n} Loss(\overbrace{y_i[w_0+\mathbf{x}_i^T\mathbf{w}_1]}^{z}) + \lambda\|\mathbf{w}_1\|/2$$



这两个损失函数的目的都是增加对分类影响较大的数据点的权重，减少与分类关系较小的数据点的权重。SVM的处理方法是只考虑support vectors，也就是和分类最相关的少数点，去学习分类器。而逻辑回归通过非线性映射，大大减小了离分类平面较远的点的权重，相对提升了与分类最相关的数据点的权重,两者的根本目的都是一样的。

svm考虑局部（支持向量），而logistic回归考虑全局，就像大学里的辅导员和教师间的区别。

辅导员关心的是挂科边缘的人，常常找他们谈话，告诫他们一定得好好学习，不要浪费大好青春，挂科了会拿不到毕业证、学位证等等，相反，对于那些相对优秀或者良好的学生，他们却很少去问，因为辅导员相信他们一定会按部就班的做好分内的事；而大学里的教师却不是这样的，他们关心的是班里的整体情况，大家是不是基本都理解了，平均分怎么样，至于某个人的分数是59还是61，他们倒不是很在意。

总结：

1. LR采用log损失，SVM采用合页损失。
2. LR对异常值敏感，SVM对异常值不敏感。
3. 在训练集较小时，SVM较适用，而LR需要较多的样本。
4. LR模型找到的那个超平面，是尽量让所有点都远离他，而SVM寻找的那个超平面，是只让最靠近中间分割线的那些点尽量远离，即只用到那些支持向量的样本。
5. 对非线性问题的处理方式不同，LR主要靠特征构造，必须组合交叉特征，特征离散化。SVM也可以这样，还可以通过kernel。
6. svm 更多的属于非参数模型，而logistic regression 是参数模型，本质不同。其区别就可以参考参数模型和非参模型的区别

那怎么根据特征数量和样本量来选择SVM和LR模型呢？Andrew NG的课程中给出了以下建议：

1. 如果Feature的数量很大，跟样本数量差不多，这时候选用LR或者是Linear Kernel的SVM
2. 如果Feature的数量比较小，样本数量一般，不算大也不算小，选用SVM+Gaussian Kernel
3. 如果Feature的数量比较小，而样本数量很多，需要手工添加一些feature变成第一种情况。(LR和不带核函数的SVM比较类似。)