机器学习损失函数总结

2013747 张怡桢

软件学院，南开大学

摘要

任务：整理典型机器学习算法的损失函数，包括线性回归、*Logistic* 回归、带正则项的 *Logistic* 回归（岭回归）、神经网络、线性可分 *SVM*、带松弛变量的 *SVM*、*AdaBoost*。

损失函数或者代价函数的目的是：衡量模型的预测能力的好坏。
损失函数 *(Loss function)*：是定义在单个训练样本上的，也就是就算一个样本的误差，比如我们想要分类，就是预测的类别和实际类别的区别，是一个样本的哦，用 L 表示。
代价函数 *(Cost function)*：是定义在整个训练集上面的，也就是所有样本的误差的总和的平均，也就是损失函数的总和的平均，有没有这个平均其实不会影响最后的参数的求解结果。

1 损失函数（Loss function）

损失函数分为：经验风险损失函数; 结构风险损失函数

- 经验风险损失函数：预测结果和实际结果之间的差别
- 结构风险损失函数：经验风险损失函数 + 正则项 (L0、L1（Lasso)、L2（Ridge）)。

损失函数的选择依据：

- 采用的机器学习算法；
- 是否易于求导；
- 数据集中异常值所占的比例。

回归损失函数：MAE，MSE，RMSE，Huber Loss，Log-Cosh，Quantile Loss
分类损失函数：log loss，CE，KL divergence，logistic loss，Focal loss，Hinge loss，Exponential loss

1.1 回归损失函数

- 平均绝对误差 MAE(Mean Absolute Error, L1 loss)

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - f(x_i)| \tag{1}$$

  - MAE 只计算误差的平均模长，而不考虑方向。
  - 优点：所有样本对平均值的残差的权重都相同，因此对异常点更鲁棒
  - 缺点：MAE 的导数恒定，即使一个很小的误差，梯度也很大，不利于网络收敛。

- 平方误差（Square loss）

$$L(Y, f(x)) = (Y - f(x))^2 \tag{2}$$

  当样本个数为 n 时，此时的损失函数变为：

$$L(Y, f(x)) = \sum_{i=1}^{n}(Y - f(x))^2 \tag{3}$$

  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。

- 均方误差 MSE(Mean Squared Error, L2 loss)

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2 \tag{4}$$

  - MSE 只考虑误差的平均大小，而不考虑其方向。
  - 优点：各点都连续光滑，梯度容易计算, 具有较为稳定的解
  - 缺点： 若误差 >1，在 MSE 中经过平方之后会进一步增大误差。如果样本中存在离群点，那么 MSE 的值受离群点的影响比 MAE 大，也就是鲁棒性不好。 当函数的输入值距离中心值较远的时候，使用梯度下降法求解的时候梯度很大，可能导致梯度爆炸。

- 均方根误差 RMSE（Root Mean Squared Error）

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2} \tag{5}$$

  - 优点：相对于 MSE，RMSE 的量纲和输入数据一样，比较起来更直观

  - 缺点：受异常值影响大；梯度爆炸

- Smooth L1

  - smooth L1 说的是光滑之后的 L1，前面说过了 L1 损失的缺点就是有折点，不光滑，导致不稳定，那如何让其变得光滑呢？smooth L1 损失函数为：

$$\text{smooth}_{L_1}(Y \mid f(x)) = \begin{cases} \frac{1}{2}(Y - f(x))^2 & |\text{Y} - \text{f(x)}| < 1 \\ |Y - f(x)| - \frac{1}{2} & |\text{Y} - \text{f(x)}| >= 1 \end{cases} \tag{6}$$

  - 优点：

    * 相比于 L1 损失函数，函数更平滑稳定，可以收敛地更快。
    * 相比于 L2 损失函数，对离群点、异常值不敏感，梯度变化相对更小，训练时不容易跑飞。

- Huber Loss

$$\text{HuberLoss}_L(Y \mid f(x)) = \begin{cases} \frac{1}{2}(Y - f(x))^2 & |\text{Y} - \text{f(x)}| <= \delta \\ \delta|Y - f(x)| - \frac{1}{2}\delta^2 & |\text{Y} - \text{f(x)}| > \delta \end{cases} \tag{7}$$

  - huber 损失是平方损失和绝对损失的综合，它克服了平方损失和绝对损失的缺点，不仅使损失函数具有连续的导数，而且利用 MSE 梯度随误差减小的特性，可取得更精确的最小值。
  - 尽管 huber 损失对异常点具有更好的鲁棒性，但是，它不仅引入了额外的参数，而且选择合适的参数比较困难，这也增加了训练和调试的工作量。

1.2 分类损失函数

对于二分类问题,$y \in \{-1, +1\}$，损失函数常表示为关于 $yf(x)$ 的单调递减形式。如图 1：

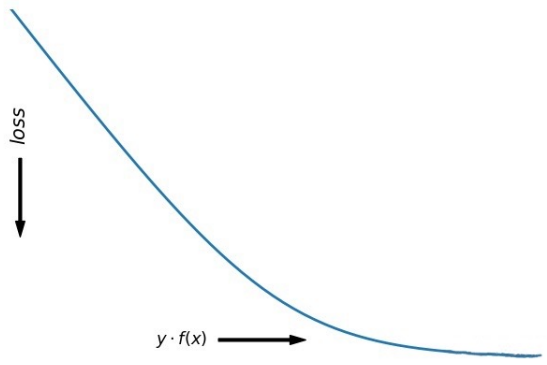


图 1. 二分类问题的损失函数经典表示 $yf(x)$

$yf(x)$ 被称为 margin，其作用类似于回归问题中的残差 $y - f(x)$。
二分类问题中的分类规则通常为

$$\text{sgn}(f(x)) = \begin{cases} +1 & \text{if} \quad yf(x) \geq 0 \\ -1 & \text{if} \quad yf(x) < 0 \end{cases} \tag{8}$$

可以看到如果 $yf(x) > 0$，则样本分类正确，$yf(x) < 0$ 则分类错误，而相应的分类决策边界即为 $f(x) = 0$。所以最小化损失函数也可以看作是最大化 margin 的过程，任何合格的分类损失函数都应该对 margin<0 的样本施以较大的惩罚。

- 0-1 损失函数

$$L(y, f(x)) = \begin{cases} 0 & \text{if} \quad yf(x) \geq 0 \\ 1 & \text{if} \quad yf(x) < 0 \end{cases} \tag{9}$$

  - 0-1 损失对每个错分类点都施以相同的惩罚，这样那些“错的离谱“(即 margin$\to\pm\infty$) 的点并不会收到大的关注，这在直觉上不是很合适。感知机就是用的这种损失函数，但是由于相等这个条件太过严格，我们可以放宽条件，即满足 $|Y-f(x)| < T$ 时认为相等。

$$L(Y, f(x)) = \begin{cases} 1 & \text{if} \quad |Y - f(x)| \geq T \\ 0 & \text{if} \quad |Y = f(x)| < T \end{cases} \tag{10}$$

  - 但是 0-1 损失不连续、非凸，优化困难，因而常使用其他的代理损失函数进行优化。

- Cross-Entropy Loss（交叉熵 Loss ）

  二分类问题的交叉熵 Loss 主要有两种形式（标签 y 的定义不同）：

  - 基于输出标签 label 的表示方式为 {0,1}，这种的就是下页图 2 吴恩达老师分析的形式

$$Loss(h_\theta(x_{(i)}), y_{(i)}) = -y_{(i)} \log h_\theta(x_{(i)}) - (1 - y_{(i)}) \log(1 - h_\theta(x_{(i)})) \tag{11}$$

    另有常见表达式如

$$-\sum_i \hat{y} \log (y_i) \tag{12}$$

$$L = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})] \tag{13}$$

  - 基于输出标签 label 的表示方式为 {-1,+1} （Logistics Loss），也比较常见。它的 Loss 表达式为：

$$L(y, f(x)) = \log \left(1 + e^{-yf(x)}\right) \tag{14}$$

    $yf(x)$ 的符号反映了预测的准确性。

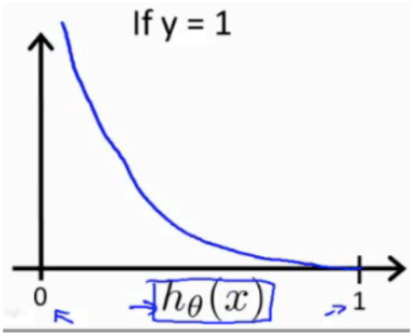
    由下页图 3 可知，$yf(x)$ 同号则 Loss 值很小，异号则 Loss 很大，可以很好得表达偏差值

We cannot use the same cost function that we use for linear regression because the Logistic Function will cause the output to be wavy, causing many local optima. In other words, it will not be a convex function.

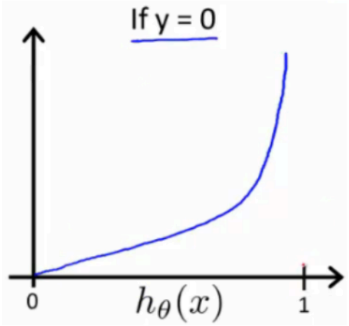
Instead, our cost function for logistic regression looks like:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$
$$\text{Cost}(h_\theta(x), y) = -\log(h_\theta(x)) \quad \text{if y} = 1$$
$$\text{Cost}(h_\theta(x), y) = -\log(1 - h_\theta(x)) \quad \text{if y} = 0$$

When y = 1, we get the following plot for $J(\theta)$ vs $h_\theta(x)$:



Similarly, when y = 0, we get the following plot for $J(\theta)$ vs $h_\theta(x)$:



$$\text{Cost}(h_\theta(x), y) = 0 \text{ if } h_\theta(x) = y$$
$$\text{Cost}(h_\theta(x), y) \to \infty \text{ if } y = 0 \text{ and } h_\theta(x) \to 1$$
$$\text{Cost}(h_\theta(x), y) \to \infty \text{ if } y = 1 \text{ and } h_\theta(x) \to 0$$

图 2. 基于吴恩达老师讲解的的交叉熵 Loss

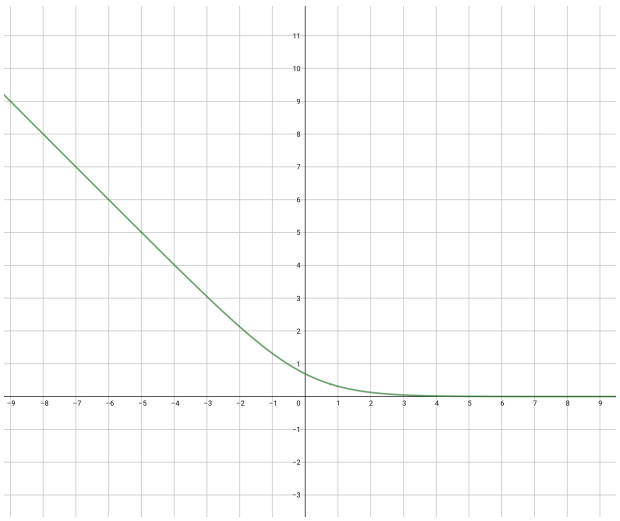


图 3. 绘制的 $\ln(1 + e^{-x})$ 图

- Log Loss(对数损失)

$$L = -\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log\left(p_{ij}\right) \tag{15}$$

其中 N 代表样本数量，C 代表类别数，y_{ij} 代表第 i 个样本属于第 j 类的真实标签，p_{ij} 表示第 i 个样本属于第 j 类的预测概率。

log 函数在趋近于 0 的时候值很大，一般要对其做 clip 处理。

运用 log loss 的典型分类器是 Logistic 回归算法，且均可用于 multi lable 和 multi class 分类。

- Focal loss

α 控制类别不平衡，γ 调节困难样本权重

$$\text{FocalLoss } = -\alpha\left(1-p_t\right)^{\gamma}\log p_t \tag{16}$$

- Hinge loss

$$L(y, f(x)) = \max(0, 1 - y \times f(x)) \tag{17}$$

Hinge loss 通常被用于最大间隔算法（maximum-margin），而最大间隔算法又是 SVM 用到的重要算法

（注意：SVM 的学习算法有两种解释：1. 间隔最大化与拉格朗日对偶；2. Hinge Loss）。

Hinge loss 专用于二分类问题，标签值 y= ±1，预测值 $\hat{y} \in R$。

  - 当 $\hat{y} \geq +1$ 或者 $\hat{y} \leq -1$ 时，都是分类器确定的分类结果，此时的损失函数 loss 为 0；
  - 而当预测值 $\hat{y} \in (-1,1)$ 时，分类器对分类结果不确定，loss 不为 0。显然，当 $\hat{y}$= 0 时，loss 达到最大值。

- Exponential Loss（指数损失）

$$L(y, f(x)) = e^{-yf(x)} \tag{18}$$

exponential loss 为 AdaBoost 中使用的损失函数，使用 exponential loss 能比较方便地利用加法模型推导出 AdaBoost 算法。然而其和 squared loss 一样，对异常点敏感，不够 robust。

2 代价函数

2.1 线性回归 Linearal Regression

- 损失函数
  - 利用的是平方误差的 $\frac{1}{2}$ 倍
  - 这里出现 $\frac{1}{2}$ 是因为后面会有求导的步骤

$$Loss(h_\theta(x_{(i)}), y_{(i)}) = \frac{1}{2}(h_\theta(x_{(i)}) - y_{(i)})^2 \tag{19}$$

- 代价函数
  - 将上面式子的损失函数均方误差

$$\begin{aligned} \mathcal{J}(\theta) &= \frac{1}{m}\sum_{i=1}^{m} Loss(h_\theta(x_{(i)}), y_{(i)}) \\ &= \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x_{(i)}) - y_{(i)})^2 \end{aligned} \tag{20}$$

- 带正则项的代价函数

$$\mathcal{J}(\theta) = \frac{1}{2m}[\sum_{i=1}^{m}(h_\theta(x_{(i)}) - y_{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2] \tag{21}$$

2.2 Logistic 回归

- 损失函数
  - 这里利用的是 Cross-Entropy Loss（交叉熵 Loss ）的损失函数

$$Loss(h_\theta(x_{(i)}), y_{(i)}) = -y_{(i)}\log h_\theta(x_{(i)}) - (1 - y_{(i)})\log(1 - h_\theta(x_{(i)})) \tag{22}$$

- 代价函数
  - 对损失函数做均值就是 Logistic 回归的代价函数

$$\begin{aligned} \mathcal{J}(\theta) &= \frac{1}{m}\sum_{i=1}^{m} Loss(h_\theta(x_{(i)}), y_{(i)}) \\ &= -\frac{1}{m}[\sum_{i=1}^{m} y_{(i)}\log h_\theta(x_{(i)}) + (1 - y_{(i)})\log(1 - h_\theta(x_{(i)}))] \end{aligned} \tag{23}$$

- 带正则项的代价函数

$$\mathcal{J}(\theta) = [-\frac{1}{m}[\sum_{i=1}^{m} y_{(i)}\log h_\theta(x_{(i)}) + (1 - y_{(i)})\log(1 - h_\theta(x_{(i)}))]] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2 \tag{24}$$

2.3 神经网络

- 基于交叉熵的代价函数

$$\begin{aligned} J(\Theta) = &-\frac{1}{m}\sum_{t=1}^{m}\sum_{k=1}^{K}\left[y_k^{(t)}\log\left(h_\Theta\left(x^{(t)}\right)\right)_k + \left(1-y_k^{(t)}\right)\log\left(1-h_\Theta\left(x^{(t)}\right)_k\right)\right] \\ &+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}\left(\Theta_{j,i}^{(l)}\right)^2 \end{aligned} \tag{25}$$

- 基于平方误差的损失函数

$$E_k = \frac{1}{2}\sum_{k=1}^{m}\sum_{j=1}^{l}\left(\hat{y}_j^k - y_j^k\right)^2 \tag{26}$$

2.4 线性可分 SVM

- 代价函数 (拉格朗日乘子法)

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{m}\alpha_i\left(1 - y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right)\right) \tag{27}$$

  - 经验风险部分为 $\sum_{i=1}^{m}\alpha_i\left(1 - y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right)\right)$
    而 $\mathbf{w}^T\mathbf{x}_i + b$ 应该 ≥ 1 或 ≤ -1，最理想的是 y_i 和 $\mathbf{w}^T\mathbf{x}_i + b$ 输出相同，损失为 0 。
  - 结构风险部分为 $\frac{1}{2}\|\mathbf{w}\|^2$
    - 如何理解双竖线：$\frac{1}{2}\|\mathbf{w}\|^2 = \frac{1}{2}\sum_{j=1}^{n}\mathbf{w}^2$
    - 结构风险公式的来源：
      1. 点到直线距离: $\frac{\left|\mathbf{w}^T\mathbf{x}+b\right|}{\|\mathbf{w}\|}$ 所以, $\mathbf{w}^T\mathbf{x}+b=1$ 上的点到 $\mathbf{w}^T\mathbf{x}+b=0$ 是 $\frac{1}{\|\mathbf{w}\|}$。
      2. 所以 $\frac{2}{\|w\|}$ ： 两类样本间的 margin ——我们要最大化 margin 才可以取得较好的分类器，这里是未使用拉格朗日乘子法的代价函数

$$\begin{aligned} &\max_{\mathbf{w},b}\frac{2}{\|\mathbf{w}\|} \\ s.t.\quad &y_i\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b\right) \geqslant 1, \\ &i = 1, 2, 3..., m \end{aligned} \tag{28}$$

      3. 利用拉格朗日乘子法把最大化问题变成最小化的问题

$$\max_{\mathbf{w},b}\frac{2}{\|\mathbf{w}\|} \Longrightarrow \min_{\mathbf{w},b}\frac{1}{2}\|\mathbf{w}\|^2 \tag{29}$$

      结构风险与正则化形式类似，都是为了避免 overfitting，因为要保证泛化能力。知识向量机的最厉害的的地方：把结构风险与最大化间隔建立联系，也即做到最大化间隔就可以保证泛化能力。

- 判决函数

  使用训练集优化上述的代价函数, 得到模型参数 w，b 后有如下的判决函数，判决函数的界限可以不是 ±1，这取决于你对于准确的度的拿捏。

$$f\left(\mathbf{x}_i;\mathbf{w}\right)=\begin{cases}1, & \mathbf{w}^T\mathbf{x}_i+b\geq 1\\ -1, & \mathbf{w}^T\mathbf{x}_i+b\leq -1\end{cases} \tag{30}$$

  对代价函数进行优化的优化模型:

$$\max_{\alpha}\left\{\min_{\mathbf{w},b}L(\mathbf{w},b,\alpha)\right\} \tag{31}$$

  通过对 w，b 的求导以及对于 α 的 KKT 求解之后，判决函数变成

$$\begin{aligned}f(\mathbf{x})&=\operatorname{sign}\left(\mathbf{w}^T\mathbf{x}+b\right)\\&=\operatorname{sign}\left(\sum_{i=1}^{m}\alpha_i y_i\mathbf{x}_i\mathbf{x}+b\right)\end{aligned} \tag{32}$$

  这里的形式的有趣之处在于，对于新点 x 的预测，只需要计算它与训练数据点 x_i 的内积即可（表示向量内积），这一点至关重要，是之后使用 Kernel 进行非线性推广的基本前提。此外，所谓 Supporting Vector 也在这里显示出来——事实上，所有非 Supporting Vector 所对应的系数都是等于零的，因此对于新点的内积计算实际上只要针对少量的“支持向量”而不是所有的训练数据即可。为什么非支持向量对应的等于零呢？直观上来理解的话，就是这些“后方”的点——正如我们之前分析过的一样，对超平面是没有影响的，由于分类完全有超平面决定，所以这些无关的点并不会参与分类问题的计算，因而也就不会产生任何影响了。

  KKT(Karush-Kuhn-Tucker) 条件是对应二次规划问题的充分必要条件:

$$\begin{cases}\alpha_i\geqslant 0\\ y_i\left(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i+b\right)-1\geqslant 0\\ \alpha_i\left(y_i\left(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i+b\right)-1\right)=0\end{cases} \tag{33}$$

  若 $\boldsymbol{a}_i>0$, 则 $y_i\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}+b\right)=1$, 对应样本位于最大间隔边界上, 是一个支持向量。

  若 $y_i\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}+b\right)\neq 1$ ，则 $\boldsymbol{a}_i=0$。正如上文分析可见，后方的点对分类没有影响。

  则上述优化模型会变成:

$$\max_{\alpha_i\geq 0}\boldsymbol{\mathcal{L}}(w,b,\alpha)=\max_{\alpha_i\geq 0}\frac{1}{2}\|w\|^2-\sum_{i=1}^{n}\alpha_i\left(y_i\left(w^T x_i+b\right)-1\right) \tag{34}$$

  注意到如果是支持向量的话，上式中经验风险部分是等于 0 的（因为支持向量的 functional margin 等于 1 ），而对于非支持向量来说，functional margin 会大于 1 ，因此经验风险部分是大于零的，而又是非负的，为了满足最大化，必须等于 0 。这也就是这些非 Supporting Vector 的点的局限性。

2.5 Soft Margin SVM(线性不可分)

- 经验风险

  原来的 Loss 部分引入 Hinge Loss

$$HingeLoss = \max\left(0, 1 - y_i\left(\boldsymbol{w}^T\boldsymbol{x} + b\right)\right) \tag{35}$$

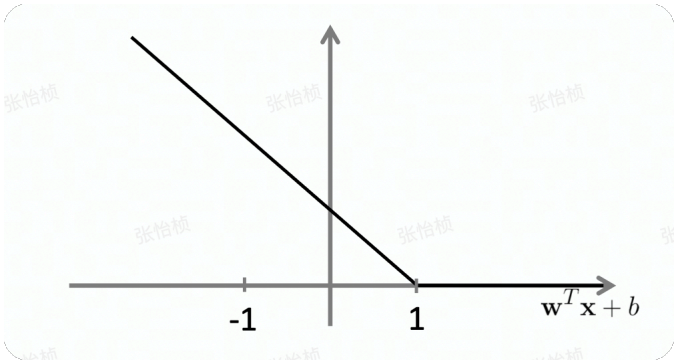


图 4. Hinge Loss

- 样本约束

  原来的样本严格的约束在 $y_i\left(\mathbf{w}^{\mathbf{T}_i} + b\right) \geqslant 1$ 现在的样本可以允许一些数据进入 margin，这将反映在对结构风险部分的调整

$$y_i\left(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b\right) \geqslant 1 - \xi_i, \quad \xi_i \geq 0, i = 1, 2, \cdots, m \tag{36}$$

- 结构风险（未使用拉格朗日乘子法的代价函数）

  该部分引入松弛变量，让一些数据样本可以进入 margin，但是对于松弛变量也要做严格的最小化处理

$$\begin{array}{ll} \min_{\mathbf{w},b,\xi_i} & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{s.t.} & y_i\left(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b\right) \geqslant 1 - \xi_i \\ & \xi_i \geqslant 0, i = 1, 2, \cdots, m \end{array} \tag{37}$$

- 代价函数（使用拉格朗日乘子法）

  对上述的结构风险与经验风险进行调整后:

$$\min_{\boldsymbol{w},b,\boldsymbol{\alpha},\boldsymbol{\xi},\boldsymbol{\mu}} \frac{1}{2}\|\boldsymbol{w}\|^2 + \sum_{i=1}^{m}\alpha_i\left(1 - \xi_i - y_i\left(\boldsymbol{w}^T\boldsymbol{x}_i + b\right)\right) + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\mu_i\xi_i \tag{38}$$

- 对代价函数进行优化的优化模型

$$\max_{\alpha}\{\min_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\mu}} L(\mathbf{w},b,\boldsymbol{\alpha},\boldsymbol{\xi},\boldsymbol{\mu})\} \tag{39}$$

求导后的优化结果：

$$\begin{aligned}&\max_{\alpha}\min_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\mu}} L(\mathbf{w},b,\boldsymbol{\alpha},\boldsymbol{\xi},\boldsymbol{\mu}) = \max_{\alpha}\sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j \\ &\text{s.t.}\ \sum_{i=1}^{m}\alpha_i y_i = 0, \quad 0 \leqslant \alpha_i \leqslant C, i = 1,2,\cdots,m\end{aligned} \tag{40}$$

2.6 AdaBoost

- 损失函数

  使用指数函数作为损失函数

$$L(y, f(x)) = e^{-yf(x)} \tag{41}$$

- 代价函数

  若将指数损失表示为期望值的形式：

$$E\left(e^{-yf(x)} \mid x\right) = P(y=1 \mid x)e^{-f(x)} + P(y=-1 \mid x)e^{f(x)} \tag{42}$$

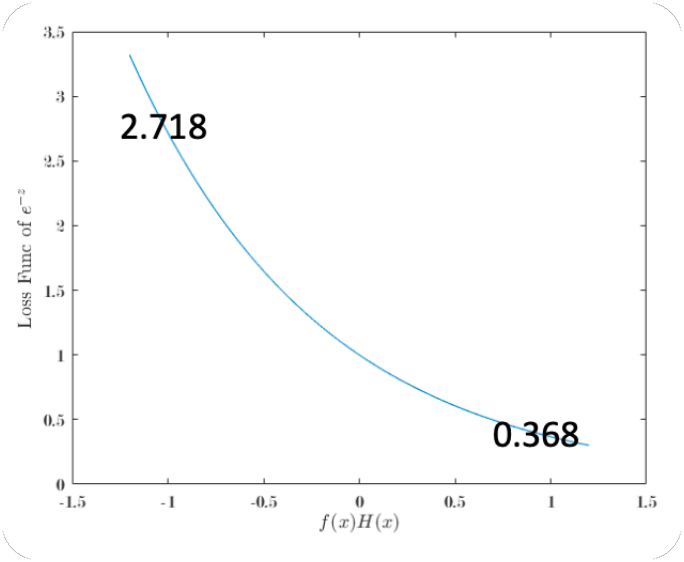


图 5. Exponential Loss