机器学习常用损失函数总览：基本形式、原理、特点

数据派THU 2020-07-16

> 来源：七月在线实验室

本文约**4300字**，建议阅读**9分钟。**
本文将介绍机器学习、深度学习中分类与回归常用的几种损失函数。

机器学习中的监督学习本质上是给定一系列训练样本 (x_i, y_i) ，尝试学习 $x \to y$的映射关系，使得给定一个 x ，即便这个 x 不在训练样本中，也能够得到尽量接近真实 y 的输出 $\hat{y}$ 。

而损失函数（Loss Function）则是这个过程中关键的一个组成部分，用来**衡量模型的输出 $\hat{y}$ 与真实的 y 之间的差距**，给模型的优化指明方向。

本文将介绍机器学习、深度学习中分类与回归常用的几种损失函数，包括均方差损失 Mean Squared Loss、平均绝对误差损失 Mean Absolute Error Loss、Huber Loss、分位数损失 Quantile Loss、交叉熵损失函数 Cross Entropy Loss、Hinge 损失 Hinge Loss。

主要介绍各种损失函数的基本形式、原理、特点等方面。

目录：

01 前言

在正文开始之前，先说下关于 Loss Function、Cost Function 和 Objective Function 的区别和联系。在机器学习的语境下这三个术语经常被交叉使用。

- 损失函数 Loss Function 通常是**针对单个训练样本而言**，给定一个模型输出 $\hat{y}$ 和一个真实 y ，损失函数输出一个实值损失
- 代价函数 Cost Function 通常是**针对整个训练集**（或者在使用 mini-batch gradient descent 时一个 mini-batch）的总损失
- 目标函数 Objective Function 是一个更通用的术语，表示任意希望被优化的函数，用于机器学习领域和非机器学习领域（比如运筹优化）

一句话总结三者的关系就是：A loss function is a part of a cost function which is a type of an objective function.

由于损失函数和代价函数只是在针对样本集上有区别，因此在本文中统一使用了损失函数这个术语，但下文的相关公式实际上采用的是代价函数 Cost Function 的形式，请读者自行留意。

02 均方差损失Mean Squared Error Loss

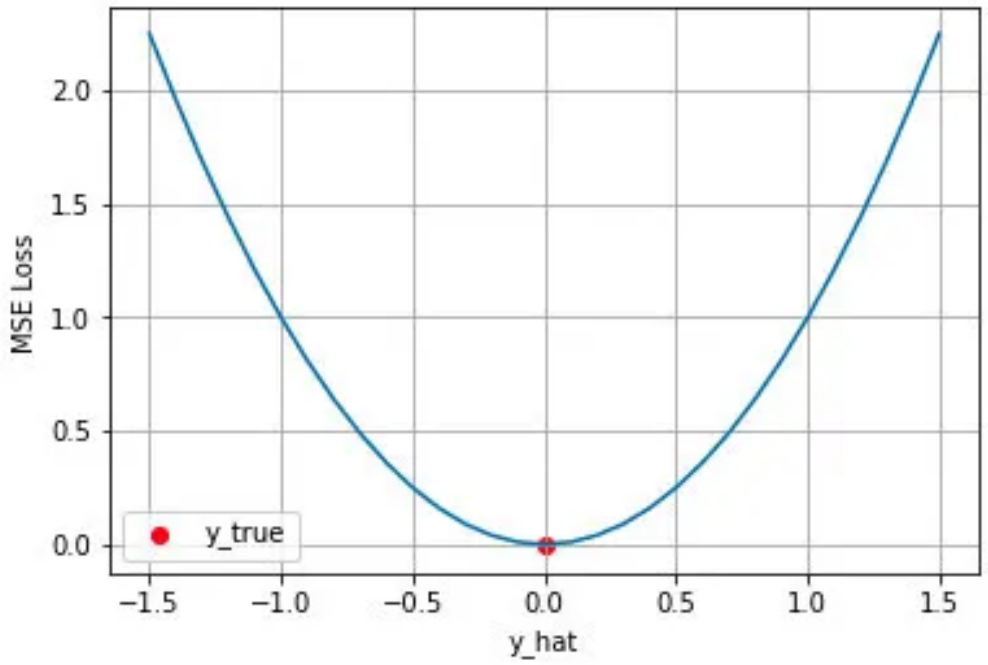
基本形式与原理：

均方差 Mean Squared Error (MSE) 损失是机器学习、深度学习回归任务中最常用的一种损失函数，也称为 L2 Loss。其基本形式如下

$$J_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

从直觉上理解均方差损失，这个损失函数的最小值为 0（当预测等于真实值时），最大值为无穷大。下图是对于真实值 $y = 0$ ，不同的预测值 $[-1.5, 1.5]$ 的均方差损失的变化图。

横轴是不同的预测值，纵轴是均方差损失，可以看到随着预测与真实值绝对误差 $|y - \hat{y}|$ 的增加，均方差损失呈二次方地增加。



背后的假设：

实际上在一定的假设下，我们可以使用最大化似然得到均方差损失的形式。假设**模型预测与真实值之间的误差服从标准高斯分布**（ $\mu = 0, \sigma = 1$ ），则给定一个 x_i模型输出真实值 y_i 的概率为

$$p(y_i|x_i) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(y_i - \hat{y}_i)^2}{2}\right)$$

进一步我们假设数据集中 N 个样本点之间相互独立，则给定所有 x 输出所有真实值 y 的概率，即似然 Likelihood，为所有 $p(y_i|x_i)$ 的累乘

$$L(x, y) = \prod_{i=1}^{N}\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(y_i - \hat{y}_i)^2}{2}\right)$$

通常为了计算方便，我们通常最大化对数似然 Log-Likelihood

$$LL(x, y) = \log(L(x, y)) = -\frac{N}{2}\log 2\pi - \frac{1}{2}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

去掉与 $\hat{y}_i$ 无关的第一项，然后转化为最小化负对数似然 Negative Log-Likelihood

$$NLL(x, y) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

可以看到这个实际上就是均方差损失的形式。也就是说**在模型输出与真实值的误差服从高斯分布的假设下，最小化均方差损失函数与极大似然估计本质上是一致的。**

因此在这个假设能被满足的场景中（比如回归），均方差损失是一个很好的损失函数选择；当这个假设没能被满足的场景中（比如分类），均方差损失不是一个好的选择。

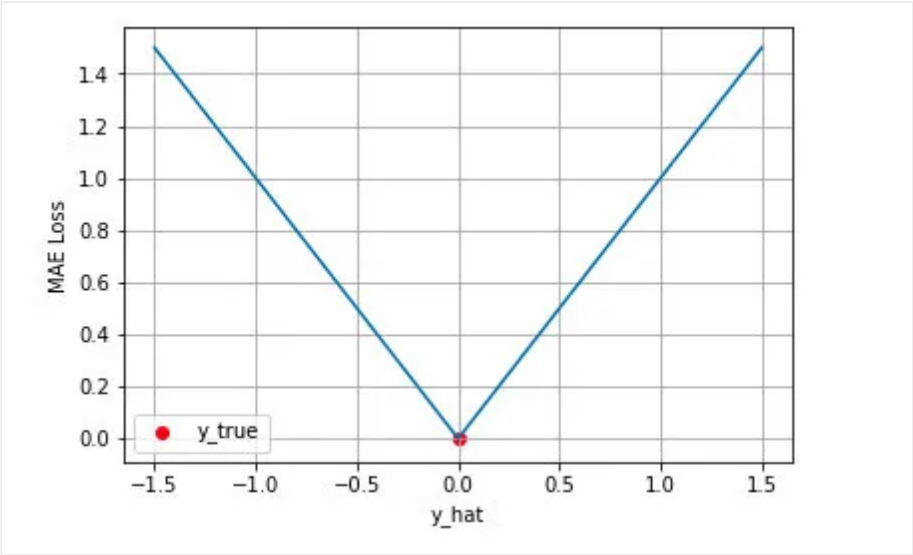
03 平均绝对误差损失

基本形式与原理：

平均绝对误差 Mean Absolute Error (MAE) 是另一类常用的损失函数，也称为 L1 Loss。其基本形式如下

$$J_{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

同样的我们可以对这个损失函数进行可视化如下图，MAE 损失的最小值为 0（当预测等于真实值时），最大值为无穷大。可以看到随着预测与真实值绝对误差 $|y - \hat{y}|$ 的增加，MAE 损失呈线性增长



背后的假设：

同样的我们可以在一定的假设下通过最大化似然得到 MAE 损失的形式，假设**模型预测与真实值之间的误差服从拉普拉斯分布 Laplace distribution**（$\mu = 0, b = 1$），则给定一个 x_i 模型输出真实值 y_i 的概率为

$$p(y_i|x_i) = \frac{1}{2}\exp(-|y_i - \hat{y}_i|)$$

与上面推导 MSE 时类似，我们可以得到的负对数似然实际上就是 MAE 损失的形式

$$L(x,y)=\prod_{i=1}^{N}\frac{1}{2}\exp(-|y_i-\hat{y}_i|)$$

$$LL(x,y)=-\frac{N}{2}-\sum_{i=1}^{N}|y_i-\hat{y}_i|$$

$$NLL(x,y)=\sum_{i=1}^{N}|y_i-\hat{y}_i|$$

MAE 与 MSE 区别：

MAE 和 MSE 作为损失函数的主要区别是：MSE 损失相比 MAE 通常可以更快地收敛，但 MAE 损失对于 outlier 更加健壮，即更加不易受到 outlier 影响。

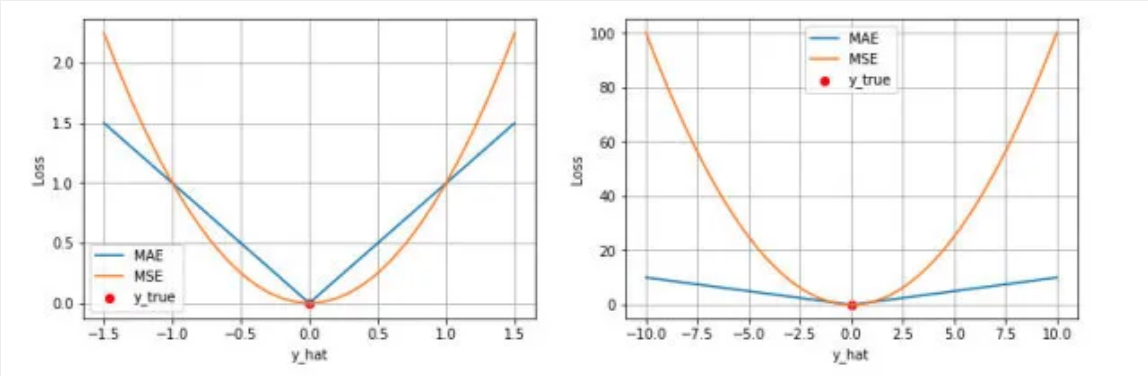
MSE 通常比 MAE 可以更快地收敛。当使用梯度下降算法时，MSE 损失的梯度为 $-\hat{y_i}$ ，而 MAE 损失的梯度为 ± 1 ，即 MSE 的梯度的 scale 会随误差大小变化，而 MAE 的梯度的 scale 则一直保持为 1，即便在绝对误差 $|y_i-\hat{y_i}|$ 很小的时候 MAE 的梯度 scale 也同样为 1，这实际上是非常不利于模型的训练的。

当然你可以通过在训练过程中动态调整学习率缓解这个问题，但是总的来说，损失函数梯度之间的差异导致了 MSE 在大部分时候比 MAE 收敛地更快。

这个也是 MSE 更为流行的原因。

MAE 对于 outlier 更加 robust。我们可以从两个角度来理解这一点：

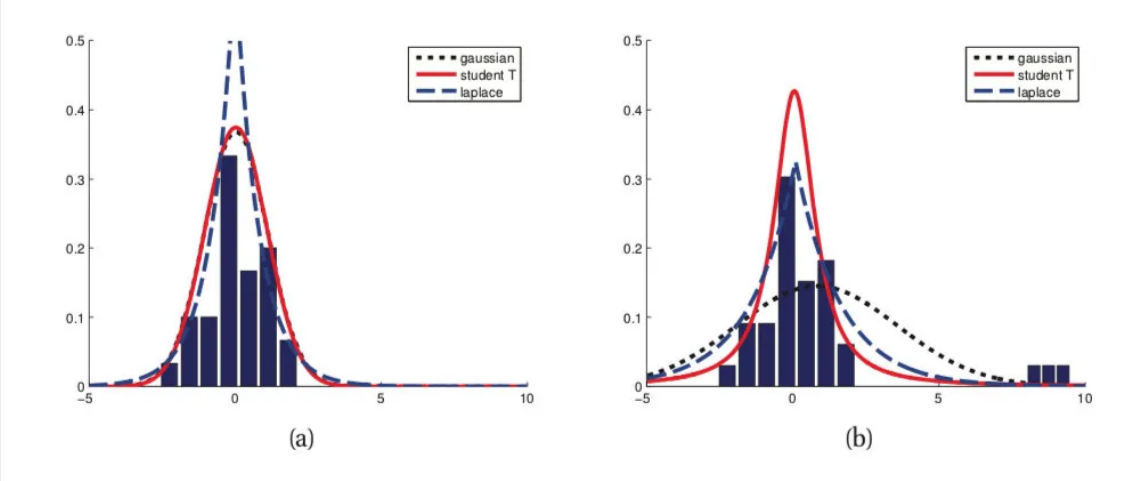
- 第一个角度是直观地理解，下图是 MAE 和 MSE 损失画到同一张图里面，由于MAE 损失与绝对误差之间是线性关系，MSE 损失与误差是平方关系，当误差非常大的时候，MSE 损失会远远大于 MAE 损失。因此当数据中出现一个误差非常大的 outlier 时，MSE 会产生一个非常大的损失，对模型的训练会产生较大的影响。



第二个角度是从两个损失函数的假设出发，MSE 假设了误差服从高斯分布，MAE 假设了误差服从拉普拉斯分布。拉普拉斯分布本身对于 outlier 更加 robust。

参考下图（来源：Machine Learning: A Probabilistic Perspective 2.4.3 The Laplace distribution Figure 2.8），当右图右侧出现了 outliers 时，拉普拉斯分布相比高斯分布受到的影响要小很多。

因此以拉普拉斯分布为假设的 MAE 对 outlier 比高斯分布为假设的 MSE 更加 robust。



04 Huber Loss

上文我们分别介绍了 MSE 和 MAE 损失以及各自的优缺点，MSE 损失收敛快但容易受 outlier 影响，MAE 对 outlier 更加健壮但是收敛慢，Huber Loss 则是一种将 MSE 与 MAE 结合起来，取两者优点的损失函数，也被称作 Smooth Mean Absolute Error Loss 。

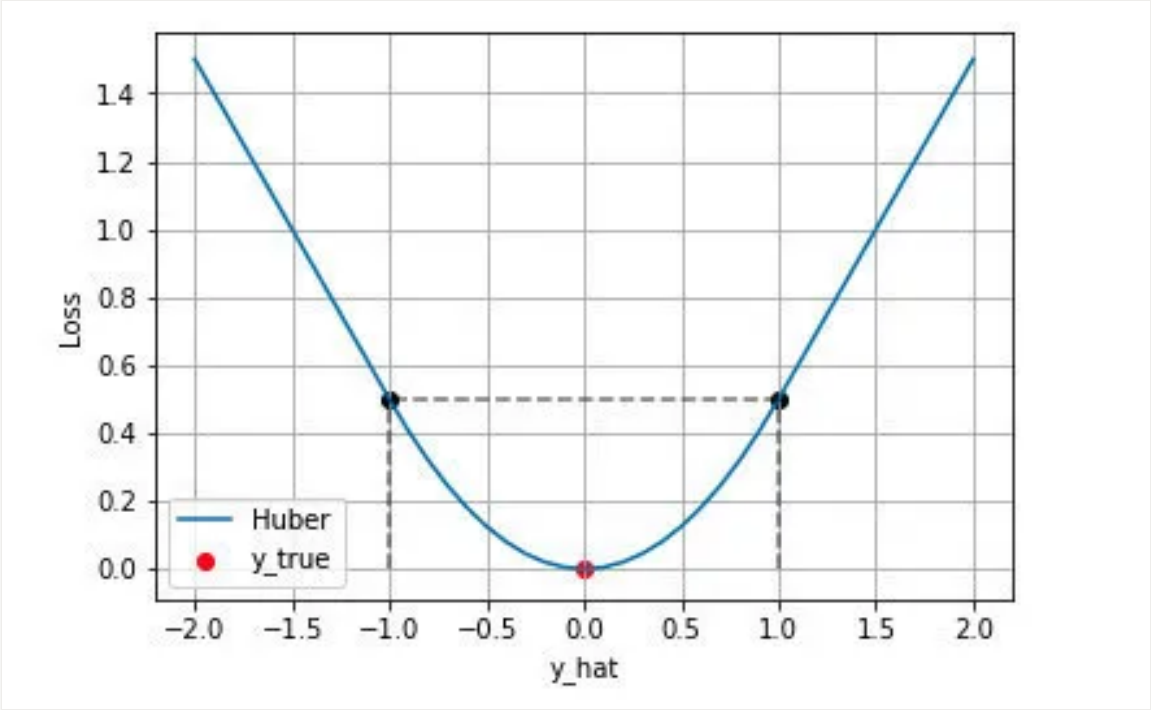
其原理很简单，就是在误差接近 0 时使用 MSE，误差较大时使用 MAE，公式为

$$J_{huber} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}_{|y_i-\hat{y}_i|\leq\delta}\frac{(y_i-\hat{y}_i)^2}{2} + \mathbb{I}_{|y_i-\hat{y}_i|>\delta}(\delta|y_i-\hat{y}_i| - \frac{1}{2}\delta^2)$$

上式中 δ 是 Huber Loss 的一个超参数，δ 的值是 MSE 和 MAE 两个损失连接的位置。

上式等号右边第一项是 MSE 的部分，第二项是 MAE 部分，在 MAE 的部分公式为 $\delta|y_i-\hat{y}_i| - \frac{1}{2}\delta^2$ 是为了保证误差 $|y-\hat{y}| = \pm\delta$ 时 MAE 和 MSE 的取值一致，进而保证 Huber Loss 损失连续可导。

下图是 $\delta = 1.0$ 时的 Huber Loss，可以看到在 $[-\delta, \delta]$ 的区间内实际上就是 MSE 损失，在 $(-\infty, \delta)$ 和 (δ, ∞) 区间内为 MAE损失。



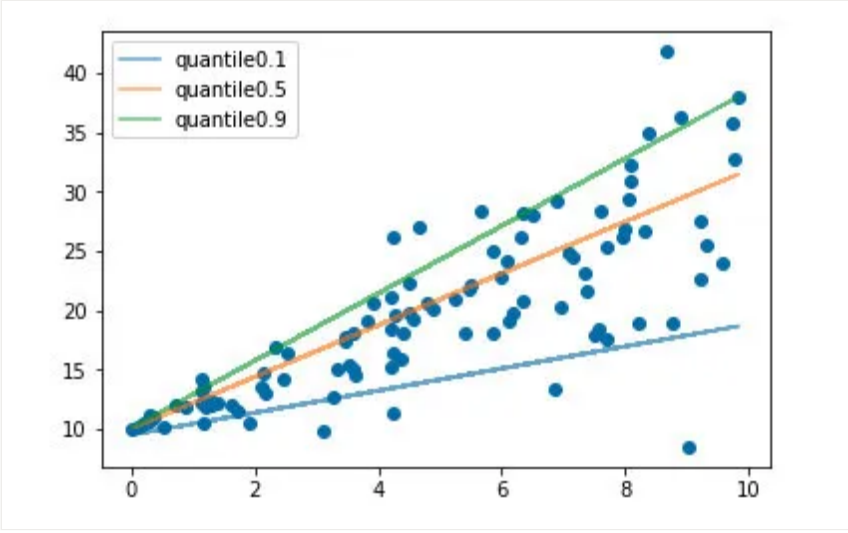
Huber Loss 的特点：

Huber Loss 结合了 MSE 和 MAE 损失，在误差接近 0 时使用 MSE，使损失函数可导并且梯度更加稳定；在误差较大时使用 MAE 可以降低 outlier 的影响，使训练对 outlier 更加健壮。缺点是需要额外地设置一个 δ 超参数。

05 分位数损失 Quantile Loss

分位数回归 Quantile Regression 是一类在实际应用中非常有用的回归算法，通常的回归算法是拟合目标值的期望或者中位数，而分位数回归可以通过给定不同的分位点，拟合目标值的不同分位数。

例如我们可以分别拟合出多个分位点，得到一个置信区间，如下图所示（图片来自笔者的一个分位数回归代码 demo Quantile Regression Demo）



分位数回归是通过使用分位数损失 Quantile Loss 来实现这一点的，分位数损失形式如下，式中的 r 分位数系数。

$$J_{quant} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}_{\hat{y}_i \geq y_i}(1-r)|y_i - \hat{y}_i| + \mathbb{I}_{\hat{y}_i < y_i} r|y_i - \hat{y}_i|$$

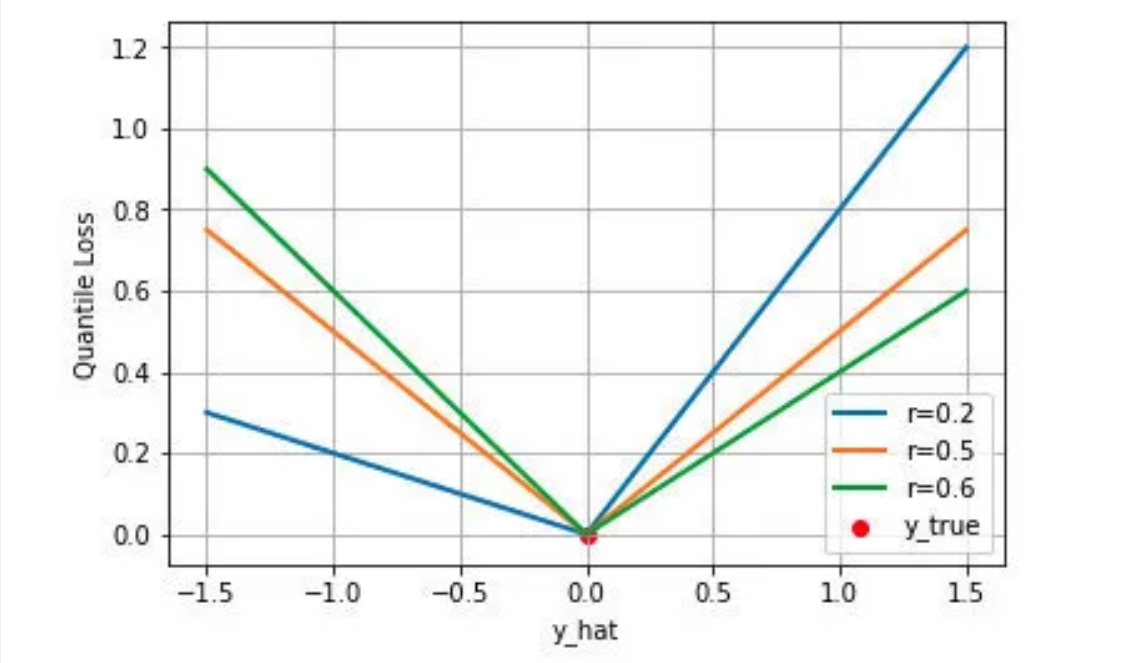
我们如何理解这个损失函数呢？这个损失函数是一个分段的函数，将 $\hat{y}_i \geq y_i$ （高估） 和 $\hat{y}_i < y_i$ （低估） 两种情况分开来，并分别给予不同的系数。

当 $r > 0.5$时，低估的损失要比高估的损失更大，反过来当 $r < 0.5$ 时，高估的损失比低估的损失大；分位数损失实现了**分别用不同的系数控制高估和低估的损失，进而实现分位数回归。**

特别地，当 $r = 0.5$ 时，分位数损失退化为 MAE 损失，从这里可以看出 MAE 损失实际上是分位数损失的一个特例 — 中位数回归（这也可以解释为什么 MAE 损失对 outlier 更鲁棒：MSE 回归期望值，MAE 回归中位数，通常 outlier 对中位数的影响比对期望值的影响小）。

$$J_{quant}^{r=0.5} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

下图是取不同的分位点 0.2、0.5、0.6 得到的三个不同的分位损失函数的可视化，可以看到 0.2 和 0.6 在高估和低估两种情况下损失是不同的，而 0.5 实际上就是 MAE。



06 交叉熵损失 Cross Entropy Loss

上文介绍的几种损失函数都是适用于回归问题损失函数，对于分类问题，最常用的损失函数是交叉熵损失函数 Cross Entropy Loss。

二分类：

考虑二分类，在二分类中我们通常使用 Sigmoid 函数将模型的输出压缩到 (0, 1) 区间内 $\hat{y}_i \in (0,1)$，用来代表给定输入 x_i，模型判断为正类的概率。由于只有正负两类，因此同时也得到了负类的概率。

$$p(y_i = 1|x_i) = \hat{y}_i$$
$$p(y_i = 0|x_i) = 1 - \hat{y}_i$$

将两条式子合并成一条

$$p(y_i|x_i) = (\hat{y}_i)^{y_i}(1-\hat{y}_i)^{1-y_i}$$

假设数据点之间独立同分布，则似然可以表示为

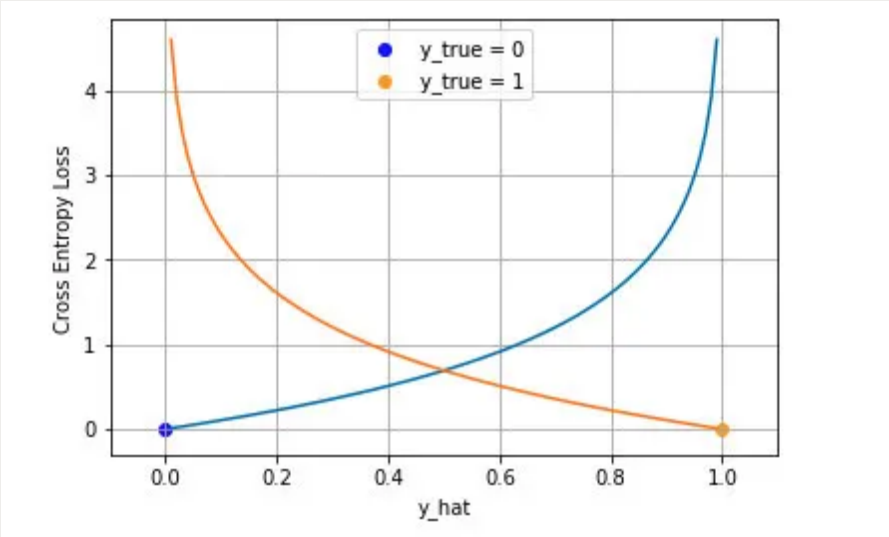
$$L(x,y) = \prod_{i=1}^{N}(\hat{y}_i)^{y_i}(1-\hat{y}_i)^{1-y_i}$$

对似然取对数，然后加负号变成最小化负对数似然，即为交叉熵损失函数的形式

$$NLL(x,y) = J_{CE} = -\sum_{i=1}^{N}(y_i log(\hat{y}_i) + (1-y_i)log(1-\hat{y}_i))$$

下图是对二分类的交叉熵损失函数的可视化，蓝线是目标值为 0 时输出不同输出的损失，黄线是目标值为 1 时的损失。

可以看到约接近目标值损失越小，随着误差变差，损失呈指数增长。



多分类：

在多分类的任务中，交叉熵损失函数的推导思路和二分类是一样的，变化的地方是真实值 y_i 现在是一个 One-hot 向量，同时模型输出的压缩由原来的 Sigmoid 函数换成 Softmax 函数。

Softmax 函数将每个维度的输出范围都限定在 $(0,1)$ 之间，同时所有维度的输出和为 1，用于表示一个概率分布。

$$p(y_i|x_i) = \prod_{k=1}^{K}(\hat{y_i^k})^{y_i^k}$$

其中 $k \in K$ 表示 K 个类别中的一类，同样的假设数据点之间独立同分布，可得到负对数似然为

$$NLL(x,y) = J_{CE} = -\sum_{i=1}^{N}\sum_{k=1}^{K} y_i^k \log(\hat{y_i^k})$$

由于 y_i 是一个 one-hot 向量，除了目标类为 1 之外其他类别上的输出都为 0，因此上式也可以写为

$$J_{CE} = -\sum_{i=1}^{N} y_i^{c_i} \log(\hat{y_i^{c_i}})$$

其中 c_i 是样本 x_i 的目标类。通常这个应用于多分类的交叉熵损失函数也被称为 Softmax Loss 或者 Categorical Cross Entropy Loss。

Cross Entropy is good. But WHY?

分类中为什么不用均方差损失？上文在介绍均方差损失的时候讲到实际上均方差损失假设了误差服从高斯分布，在分类任务下这个假设没办法被满足，因此效果会很差。

为什么是交叉熵损失呢？有两个角度可以解释这个事情，一个角度从最大似然的角度，也就是我们上面的推导；另一个角度是可以用信息论来解释交叉熵损失：

假设对于样本 x_i 存在一个最优分布 $y_i^\star$ 真实地表明了这个样本属于各个类别的概率，那么我们希望模型的输出 $\hat{y_i}$ 尽可能地逼近这个最优分布，在信息论中，我们可以使用 KL 散度 Kullback–Leibler Divergence 来衡量两个分布的相似性。给定分布 p 和分布 q， 两者的 KL 散度公式如下

$$KL(p,q) = \sum_{k=1}^{K} p^k \log(p^k) - \sum_{k=1}^{K} p^k \log(q^k)$$

其中第一项为分布 p 的信息熵，第二项为分布 p 和 q 的交叉熵。将最优分布 $y_i^\star$ 和输出分布 $\hat{y}_i$ 带入 p 和 q 得到

$$KL(y_i^\star, \hat{y}_i) = \sum_{k=1}^{K} y_i^{\star k} \log(y_i^{\star k}) - \sum_{k=1}^{K} y_i^{\star k} \log(\hat{y_i^k})$$

由于我们希望两个分布尽量相近，因此我们最小化 KL 散度。同时由于上式第一项信息熵仅与最优分布本身相关，因此我们在最小化的过程中可以忽略掉，变成最小化

$$-\sum_{k=1}^{K} y_i^{\star k} \log(\hat{y_i^k})$$

我们并不知道最优分布 $y_i^\star$ ，但训练数据里面的目标值 y_i 可以看做是 $y_i^\star$ 的一个近似分布

$$-\sum_{k=1}^{K} y_i^{k} \log(\hat{y_i^k})$$

这个是针对单个训练样本的损失函数，如果考虑整个数据集，则

$$J_{KL} = -\sum_{i=1}^{N} \sum_{k=1}^{K} y_i^{k} \log(\hat{y_i^k}) = -\sum_{i=1}^{N} y_i^{c_i} \log(\hat{y_i^{c_i}})$$

可以看到通过最小化交叉熵的角度推导出来的结果和使用最大化似然得到的结果是一致的。

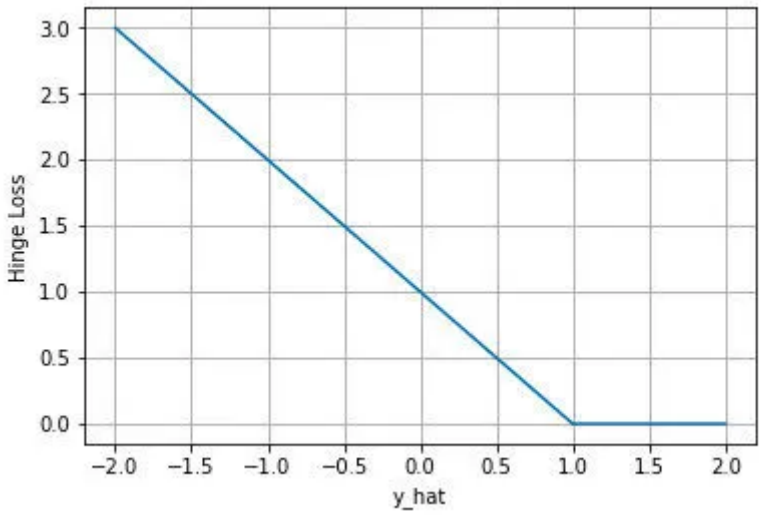
07 合页损失 Hinge Loss

合页损失 Hinge Loss 是另外一种二分类损失函数，适用于 maximum-margin 的分类，支持向量机 Support Vector Machine (SVM) 模型的损失函数本质上就是 Hinge Loss + L2 正则化。

合页损失的公式如下

$$J_{hinge} = \sum_{i=1}^{N} \max(0, 1 - \mathrm{sgn}(y_i)\hat{y}_i)$$

下图是 y 为正类， 即 $\mathbf{sgn}(y) = 1$ 时，不同输出的合页损失示意图



可以看到当 y 为正类时，模型输出负值会有较大的惩罚，当模型输出为正值且在 $(0,1)$ 区间时还会有一个较小的惩罚。

即合页损失不仅惩罚预测错的，并且对于预测对了但是置信度不高的也会给一个惩罚，只有置信度高的才会有零损失。

使用合页损失直觉上理解是要找到一个决策边界，使得所有数据点被这个边界正确地、高置信地被分类。

08 总结

本文针对机器学习中最常用的几种损失函数进行相关介绍，首先是适用于回归的均方差损失 Mean Squared Loss、平均绝对误差损失 Mean Absolute Error Loss，两者的区别以及两者相结合得到的 Huber Loss，接着是应用于分位数回归的分位数损失 Quantile Loss，表明了平均绝对误差损失实际上是分位数损失的一种特例。

在分类场景下，本文讨论了最常用的交叉熵损失函数 Cross Entropy Loss，包括二分类和多分类下的形式，并从信息论的角度解释了交叉熵损失函数，最后简单介绍了应用于 SVM 中的 Hinge 损失 Hinge Loss。

受限于时间，本文还有其他许多损失函数没有提及，比如应用于 Adaboost 模型中的指数损失 Exponential Loss，0-1 损失函数等。

另外通常在损失函数中还会有正则项（L1/L2 正则），这些正则项作为损失函数的一部分，通过约束参数的绝对值大小以及增加参数稀疏性来降低模型的复杂度，防止模型过拟合，这部分内容在本文中也没有详细展开。

读者有兴趣可以查阅相关的资料进一步了解。That’s all. Thanks for reading.