AI基础：机器学习的损失函数

机器学习初学者 今天

以下文章来源于AI有道，作者红色石头

0.导语

无论在机器学习还是深度领域中,损失函数都是一个非常重要的知识点。损失函数（Loss Function）是用来估量模型的预测值 f(x) 与真实值 y 的不一致程度。我们的目标就是最小化损失函数，让 f(x) 与 y 尽量接近。通常可以使用梯度下降算法寻找函数最小值。

损失函数有许多不同的类型，没有哪种损失函数适合所有的问题，需根据具体模型和问题进行选择。一般来说，损失函数大致可以分成两类：回归（Regression）和分类（Classification）。

目前已经发布：

AI 基础：简易数学入门

AI 基础：Python开发环境设置和小技巧

AI 基础：Python 简易入门

AI 基础：Numpy 简易入门

AI 基础：Pandas 简易入门

AI 基础：Scipy(科学计算库) 简易入门

AI基础：数据可视化简易入门（matplotlib和seaborn）

AI基础：机器学习库Scikit-learn的使用

AI基础：机器学习简易入门

AI基础：特征工程-类别特征

AI基础：特征工程-数字特征处理

AI基础：特征工程-文本特征处理

AI基础：词嵌入基础和Word2Vec

AI基础：图解Transformer

AI基础：一文看懂BERT

后续持续更新

> 本文作者：红色石头
>
> 出处：AI有道

1.回归损失函数

回归模型中的三种损失函数包括：均方误差（Mean Square Error）、平均绝对误差（Mean Absolute Error，MAE）、Huber Loss。

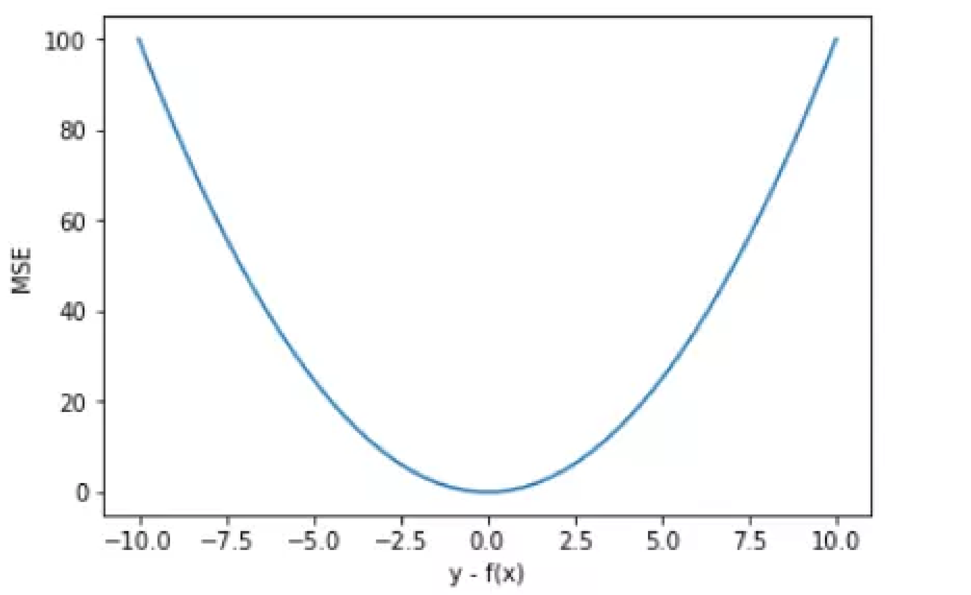
1. 均方误差（Mean Square Error，MSE）

均方误差指的就是模型预测值 f(x) 与样本真实值 y 之间距离平方的平均值。其公式如下所示：

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

其中，yi 和 f(xi) 分别表示第 i 个样本的真实值和预测值，m 为样本个数。

为了简化讨论，忽略下标 i，m = 1，以 y-f(x) 为横坐标，MSE 为纵坐标，绘制其损失函数的图形：



MSE 曲线的特点是光滑连续、可导，便于使用梯度下降算法，是比较常用的一种损失函数。而且，MSE 随着误差的减小，梯度也在减小，这有利于函数的收敛，即使固定学习因子，函数也能较快取得最小值。

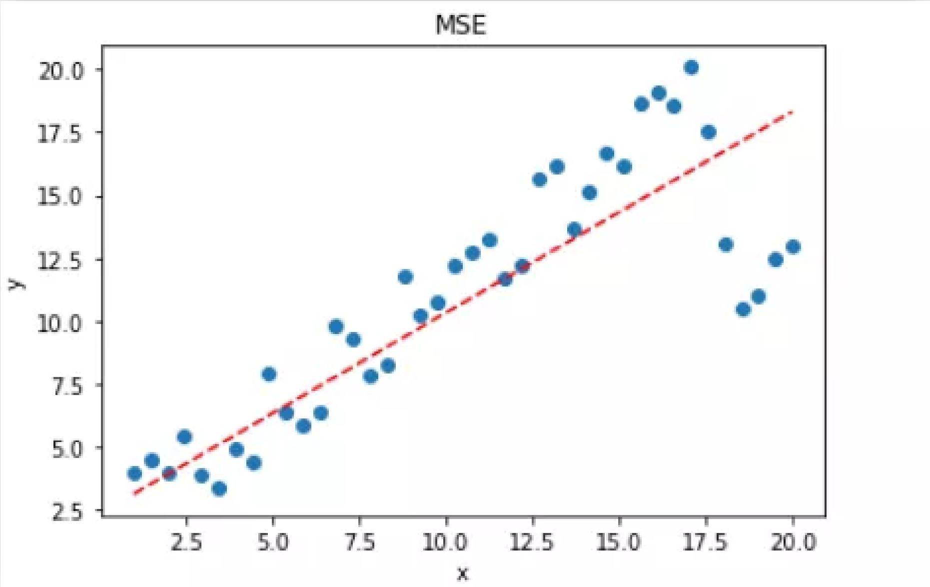
平方误差有个特性，就是当 yi 与 f(xi) 的差值大于 1 时，会增大其误差；当 yi 与 f(xi) 的差值小于 1 时，会减小其误差。这是由平方的特性决定的。也就是说， MSE 会对误差较大（>1）的情况给予更大的惩罚，对误差较小（<1）的情况给予更小的惩罚。从训练的角度来看，模型会更加偏向于惩罚较大的点，赋予其更大的权重。

如果样本中存在离群点，MSE 会给离群点赋予更高的权重，但是却是以牺牲其他正常数据点的预测效果为代价，这最终会降低模型的整体性能。我们来看一下使用 MSE 解决含有离群点的回归模型。

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(1, 20, 40)
y = x + [np.random.choice(4) for _ in range(40)]
y[-5:] -= 8
X = np.vstack((np.ones_like(x),x))       # 引入常数项 1
m = X.shape[1]
# 参数初始化
W = np.zeros((1,2))

# 迭代训练
num_iter = 20
lr = 0.01
J = []
for i in range(num_iter):
```

```
    y_pred  = W.dot(X)
    loss =  1/(2*m) * np.sum((y-y_pred)**2)
    J.append(loss)
    W = W + lr *  1/m * (y-y_pred).dot(X.T)

# 作图
y1 = W[0,0] + W[0,1]*1
y2 = W[0,0] + W[0,1]*20
plt.scatter(x, y)
plt.plot([1,20],[y1,y2])
plt.show()
```

拟合结果如下图所示：



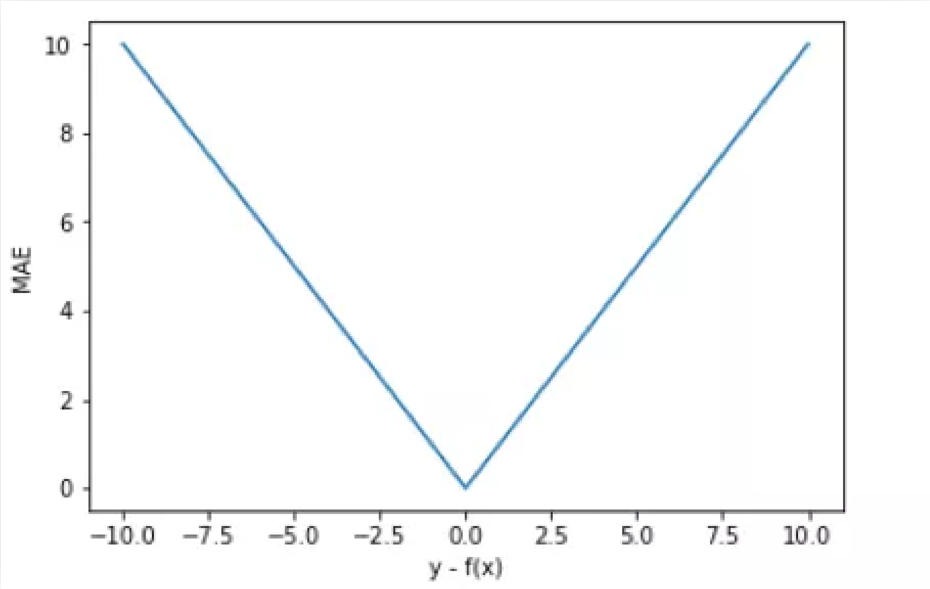
可见，使用 MSE 损失函数，受离群点的影响较大，虽然样本中只有 5 个离群点，但是拟合的直线还是比较偏向于离群点。这往往是我们不希望看到的。

2. 平均绝对误差（Mean Absolute Error，MAE）

平均绝对误差指的就是模型预测值 f(x) 与样本真实值 y 之间距离的平均值。其公式如下所示：

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

为了简化讨论，忽略下标 i，m = 1，以 y-f(x) 为横坐标，MAE 为纵坐标，绘制其损失函数的图形：



直观上来看，MAE 的曲线呈 V 字型，连续但在 y-f(x)=0 处不可导，计算机求解导数比较困难。而且 MAE 大部分情况下梯度都是相等的，这意味着即使对于小的损失值，其梯度也是大的。这不利于函数的收敛和模型的学习。

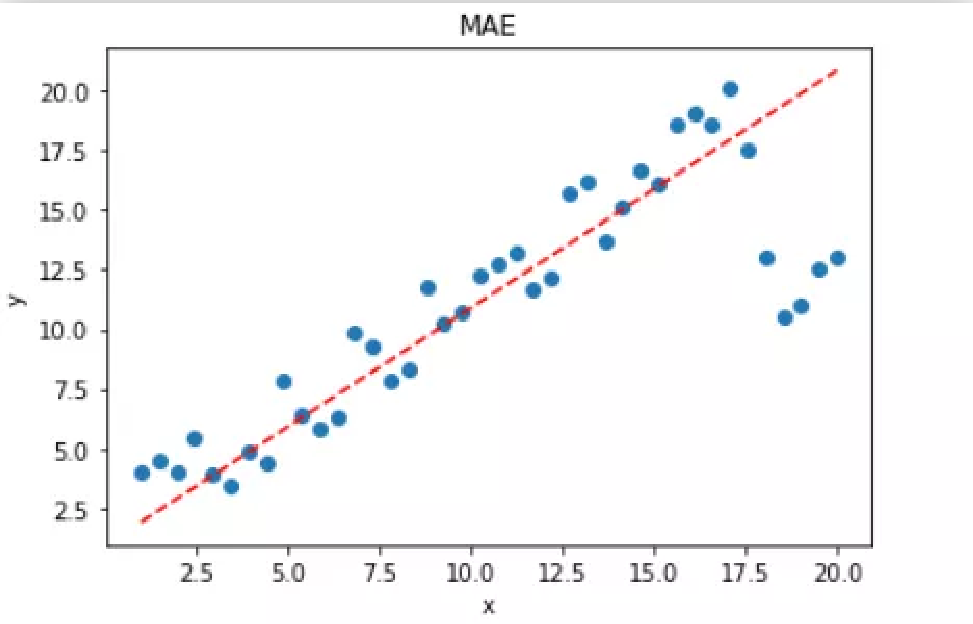
值得一提的是，MAE 相比 MSE 有个优点就是 MAE 对离群点不那么敏感，更有包容性。因为 MAE 计算的是误差 y-f(x) 的绝对值，无论是 y-f(x)>1 还是 y-f(x)<1，没有平方项的作用，惩罚力度都是一样的，所占权重一样。针对 MSE 中的例子，我们来使用 MAE 进行求解，看下拟合直线有什么不同。

```
X  = np.vstack((np.ones_like(x),x))      # 引入常数项 1
m  =  X.shape[1]
# 参数初始化
W = np.zeros((1,2))

# 迭代训练
num_iter =  20
lr  =  0.01
J = []
for  i in  range(num_iter):
     y_pred = W.dot(X)
     loss =  1/m  * np.sum(np.abs(y-y_pred))
     J.append(loss)
     mask = (y-y_pred).copy()
     mask[y-y_pred >  0] =  1
     mask[mask <=  0] = -1
     W = W +  lr  *  1/m  * mask.dot(X.T)

# 作图
```

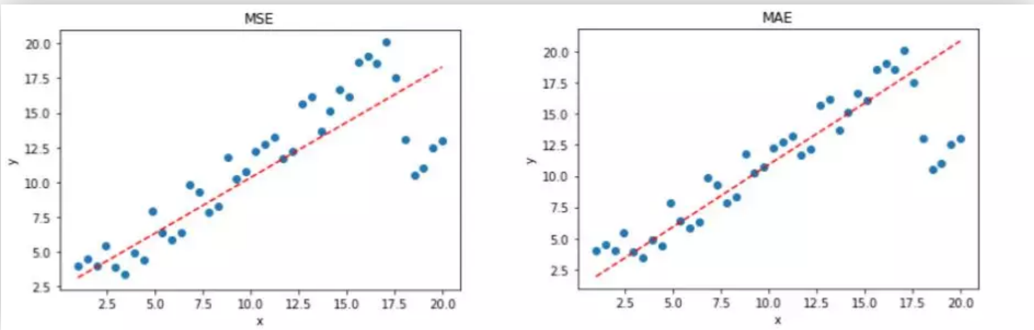
```
y1 = W[0,0] + W[0,1]*1
y2 = W[0,0] + W[0,1]*20
plt.scatter(x,  y)
plt.plot([1,20],[y1,y2],'r--')
plt.xlabel('x')
plt.ylabel('y')
plt.title('MAE')
plt.show()
```

注意上述代码中对 MAE 计算梯度的部分。

拟合结果如下图所示：



显然，使用 MAE 损失函数，受离群点的影响较小，拟合直线能够较好地表征正常数据的分布情况。这一点，MAE 要优于 MSE。二者的对比图如下：



选择 MSE 还是 MAE 呢？

实际应用中，我们应该选择 MSE 还是 MAE 呢？从计算机求解梯度的复杂度来说，MSE 要优于 MAE，而且梯度也是动态变化的，能较快准确达到收敛。但是从离群点角度来看，如果离群点是实际数据或重要数据，而且是应该被检测到的异常值，那么我们应该使用MSE。另一方面，离群点仅仅代表数据损坏或者错误采样，无须给予过多关注，那么我们应该选择MAE作为损失。

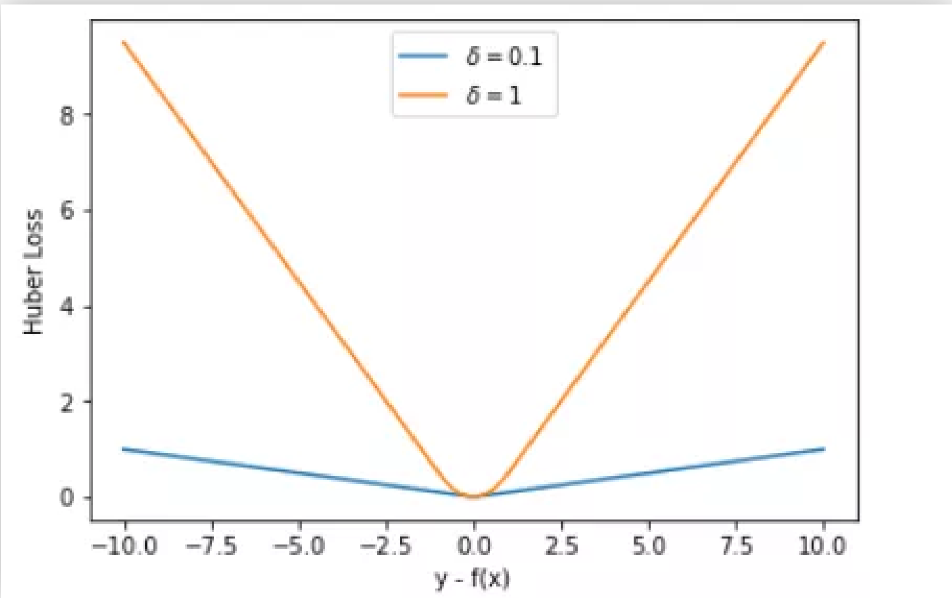
3. Huber Loss

既然 MSE 和 MAE 各有优点和缺点，那么有没有一种激活函数能同时消除二者的缺点，集合二者的优点呢？答案是有的。Huber Loss 就具备这样的优点，其公式如下：

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2, & |y - f(x)| \le \delta \\ \delta|y - f(x)| - \frac{1}{2}\delta^2, & |y - f(x)| > \delta \end{cases}$$

Huber Loss 是对二者的综合，包含了一个超参数 δ。δ 值的大小决定了 Huber Loss 对 MSE 和 MAE 的侧重性，当 |y−f(x)| ≤ δ 时，变为 MSE；当 |y−f(x)| > δ 时，则变成类似于 MAE，因此 Huber Loss 同时具备了 MSE 和 MAE 的优点，减小了对离群点的敏感度问题，实现了处处可导的功能。

通常来说，超参数 δ 可以通过交叉验证选取最佳值。下面，分别取 δ = 0.1、δ = 10，绘制相应的 Huber Loss，如下图所示：



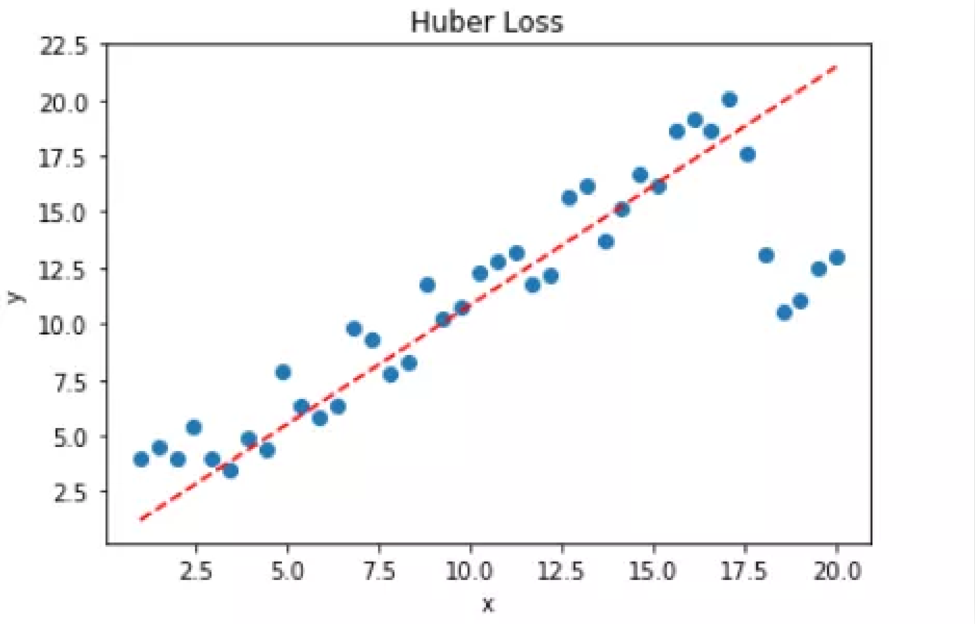
Huber Loss 在 |y−f(x)| > δ 时，梯度一直近似为 δ，能够保证模型以一个较快的速度更新参数。当 |y−f(x)| ≤ δ 时，梯度逐渐减小，能够保证模型更精确地得到全局最优值。因此，Huber Loss 同时具备了前两种损失函数的优点。

下面，我们用 Huber Loss 来解决同样的例子。

```
X  = np.vstack((np.ones_like(x),x))      # 引入常数项  1
m  =  X.shape[1]
# 参数初始化
W = np.zeros((1,2))

# 迭代训练
num_iter =  20
lr  =  0.01
delta =  2
J = []
for  i in  range(num_iter):
     y_pred = W.dot(X)
     loss =  1/m  * np.sum(np.abs(y-y_pred))
     J.append(loss)
     mask = (y-y_pred).copy()
     mask[y-y_pred > delta] = delta
     mask[mask < -delta] = -delta
     W = W +  lr  *  1/m  * mask.dot(X.T)

# 作图
y1 = W[0,0] + W[0,1]*1
y2 = W[0,0] + W[0,1]*20
plt.scatter(x,  y)
plt.plot([1,20],[y1,y2],'r--')
plt.xlabel('x')
plt.ylabel('y')
plt.title('MAE')
plt.show()
```

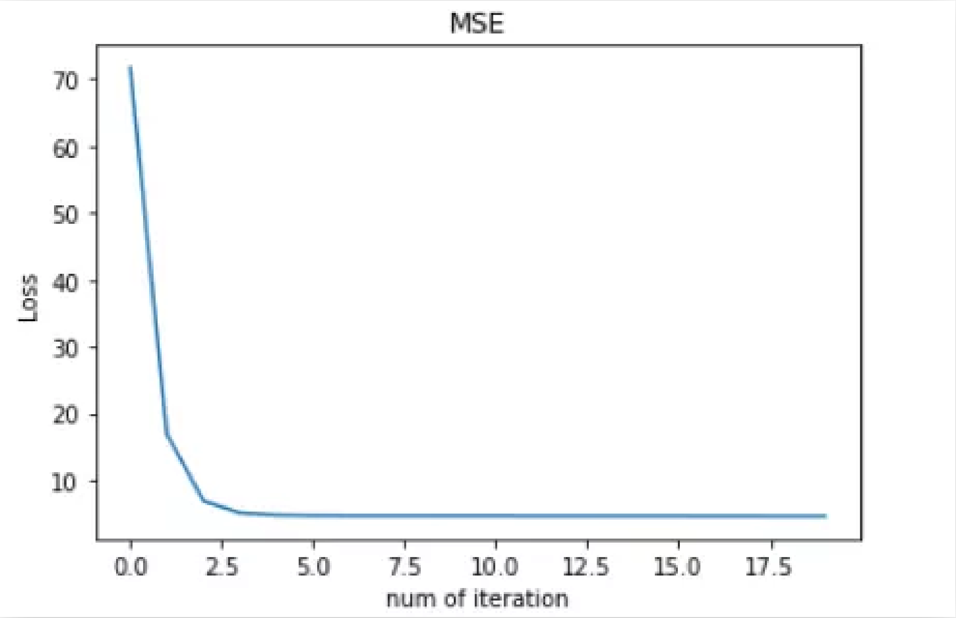
注意上述代码中对 Huber Loss 计算梯度的部分。

拟合结果如下图所示：

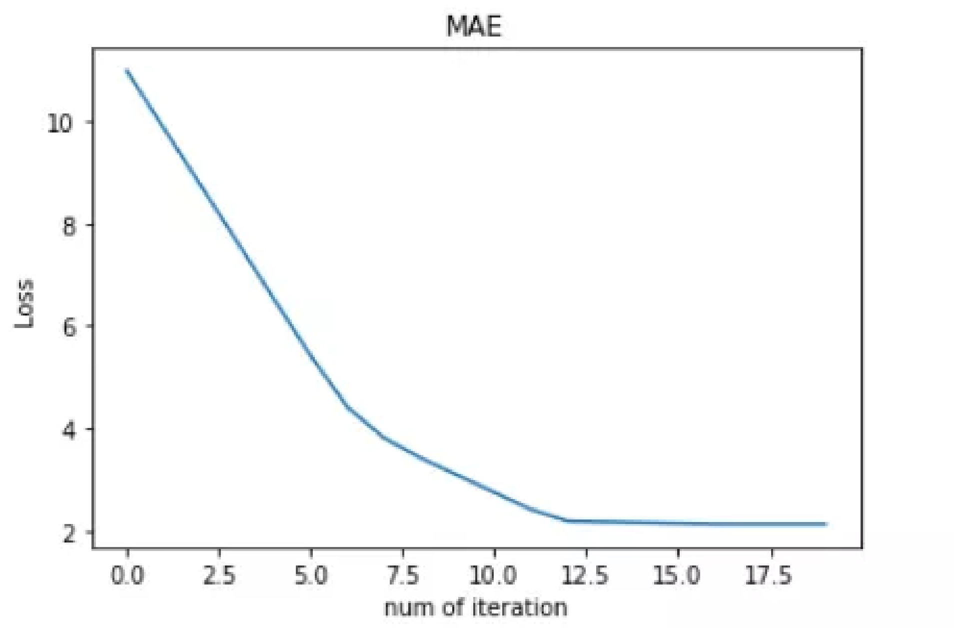


可见，使用 Huber Loss 作为激活函数，对离群点仍然有很好的抗干扰性，这一点比 MSE 强。另外，我们把这三种损失函数对应的 Loss 随着迭代次数变化的趋势绘制出来：

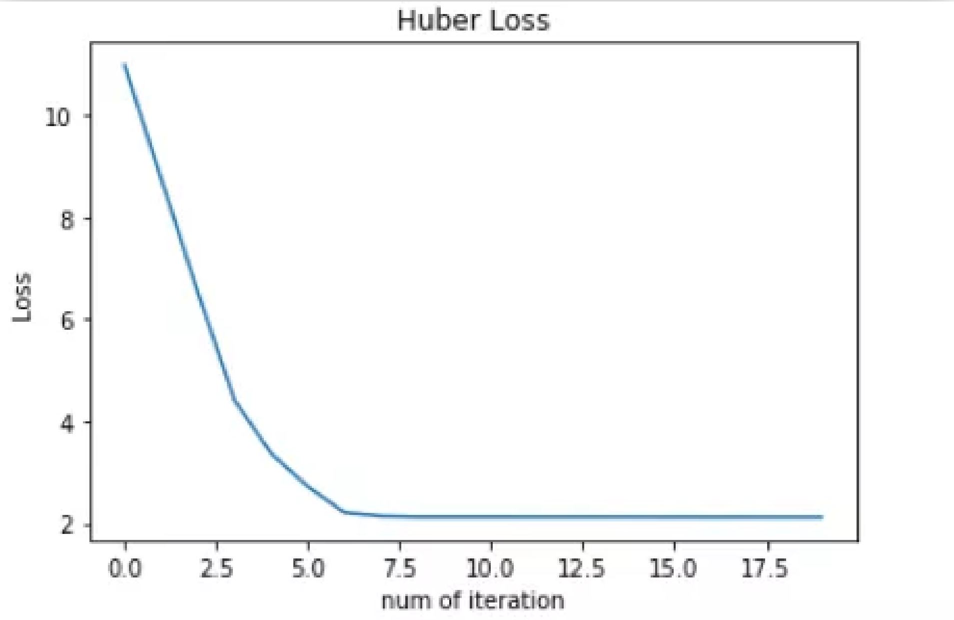
MSE：



MAE：

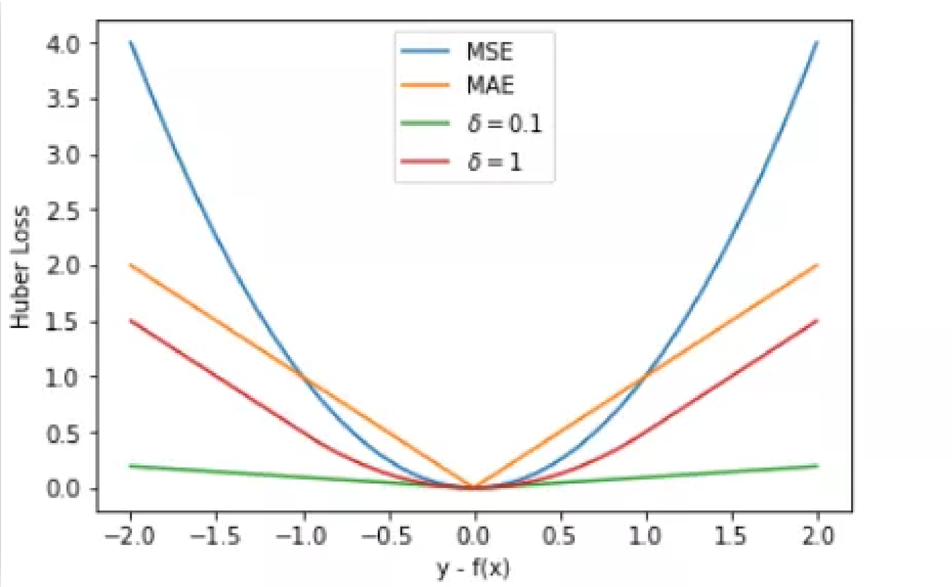


Huber Loss：



对比发现，MSE 的 Loss 下降得最快，MAE 的 Loss 下降得最慢，Huber Loss 下降速度介于 MSE 和 MAE 之间。也就是说，Huber Loss 弥补了此例中 MAE 的 Loss 下降速度慢的问题，使得优化速度接近 MSE。

最后，我们把以上介绍的回归问题中的三种损失函数全部绘制在一张图上。



好了，以上就是回归问题 3 种常用的损失函数包括：MSE、MAE、Huber Loss 的简单介绍和详细对比。

2.分类损失函数

0.模型输出

在讨论分类问题的损失函数之前，我想先说一下模型的输出 g(s)。一般来说，二分类机器学习模型包含两个部分：线性输出 s 和非线性输出 g(s)。其中，线性输出一般是模型输入 x 与 参数 w 的乘积，简写成：s = wx；非线性输出一般是 Sigmoid 函数，其表达式如下：

$$g(s)=\frac{1}{1+e^{-s}}$$

经过 Sigmoid 函数，g(s) 值被限定在 [0,1] 之间，若 s ≥ 0，g(s) ≥ 0.5，则预测为正类；若 s < 0，g(s) < 0.5，则预测为负类。

关于正类和负类的表示，通常有两种方式：一种是用 {+1, -1} 表示正负类；另一种是用 {1, 0} 表示正负类。下文默认使用 {+1, -1}，因为这种表示方法有种好处。如果使用 {+1, -1} 表示正负类，我们来看预测类别与真实类别的四种情况：

- s ≥ 0, y = +1: 预测正确
- s ≥ 0, y = -1: 预测错误
- s < 0, y = +1: 预测错误

- s < 0, y = -1: 预测正确

发现了吗？显然，上面四个式子可以整合成直接看 ys 的符号即可：

- 若 ys ≥ 0，则预测正确
- 若 ys < 0，则预测错误

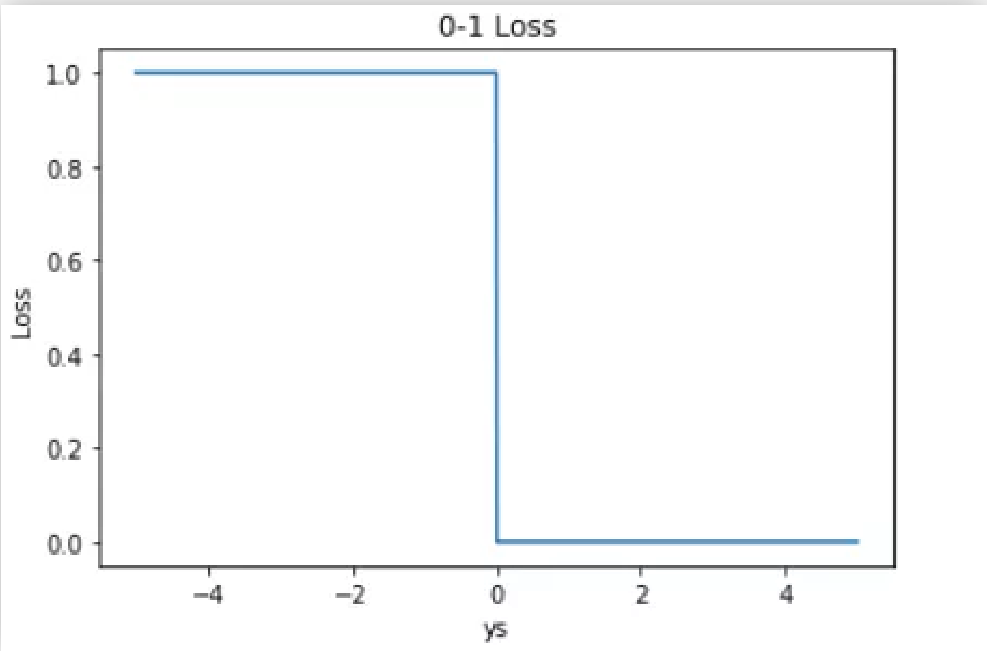
这里的 ys 类似与回归模型中的残差 s - y。因此，在比较分类问题的各个损失函数的时候，我们就可以把 ys 当作自变量 x 轴即可，这样更加方便。

1.0-1 Loss

0-1 Loss 是最简单也是最容易直观理解的一种损失函数。对于二分类问题，如果预测类别 y_hat 与真实类别 y 不同，则 L=1；如果预测类别 y_hat 与 真实类别 y 相同，则 L=0（L 表示损失函数）。0-1 Loss 的表达式为：

$$L(y, s) = \begin{cases} 0, & ys \geq 0 \\ 1, & ys < 0 \end{cases}$$

0-1 Loss 的曲线如下图所示：



0-1 Loss 的特点就是非常直观容易理解。但是它存在两个缺点：

- 0-1 Loss 对每个错分类点都施以相同的惩罚（损失为 1），这样对犯错比较大的点（ys 远小于 0）无法进行较大的惩罚，所有犯错点都同等看待，这不符合常理，不太合适。
- 0-1 Loss 不连续、非凸、不可导，难以使用梯度优化算法。

因此，实际应用中，0-1 Loss 很少使用。

2.Cross Entropy Loss

Cross Entropy Loss 是非常重要的损失函数，也是应用最多的损失函数之一。二分类问题的交叉熵 Loss 主要有两种形式，下面分别详细介绍。

第一种形式是基于输出标签 label 的表示方式为 {0,1}，也最为常见。它的 Loss 表达式为：

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

这个公式是如何推导的呢？很简单，从极大似然性的角度出发，预测类别的概率可以写成：

$$P(y|x)=\hat{y}^y\cdot(1-\hat{y})^{(1-y)}$$

我们可以这么来看，当真实样本标签 y = 1 时，上面式子第二项就为 1，概率等式转化为：

$$P(y=1|x)=\hat{y}$$

当真实样本标签 y = 0 时，上面式子第一项就为 1，概率等式转化为：

$$P(y=0|x)=1-\hat{y}$$

我们希望的是概率 P(y|x) 越大越好。首先，我们对 P(y|x) 引入 log 函数，因为 log 运算并不会影响函数本身的单调性。则有：

$$log\ P(y|x)=log(\hat{y}^y\cdot(1-\hat{y})^{(1-y)})=ylog\ \hat{y}+(1-y)log\ (1-\hat{y})$$

我们希望 log P(y|x) 越大越好，反过来，只要 log P(y|x) 的负值 -log P(y|x) 越小就行了。那我们就可以引入损失函数，且令 Loss = -log P(y|x) 即可。则得到损失函数为：

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

我们来看，当 y = 1 时：

$$L = -log\ \hat{y}$$

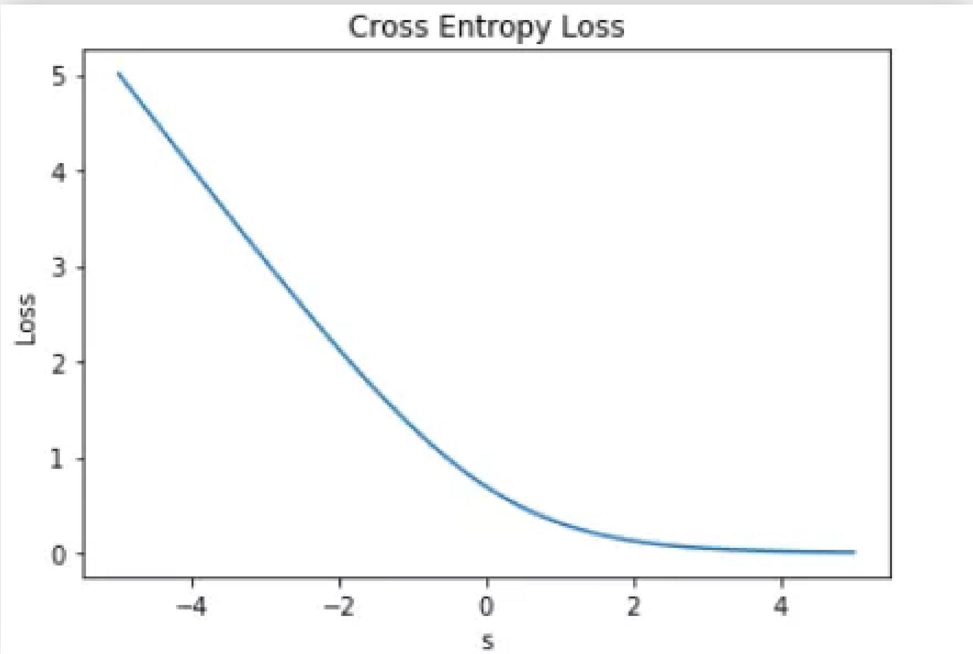
因为，

$$\hat{y} = \frac{1}{1+e^{-s}}$$

所以，代入到 L 中，得：

$$L = log(1+e^{-s})$$

这时候，Loss 的曲线如下图所示：



从图中明显能够看出，s 越大于零，L 越小，函数的变化趋势也完全符合实际需要的情况。

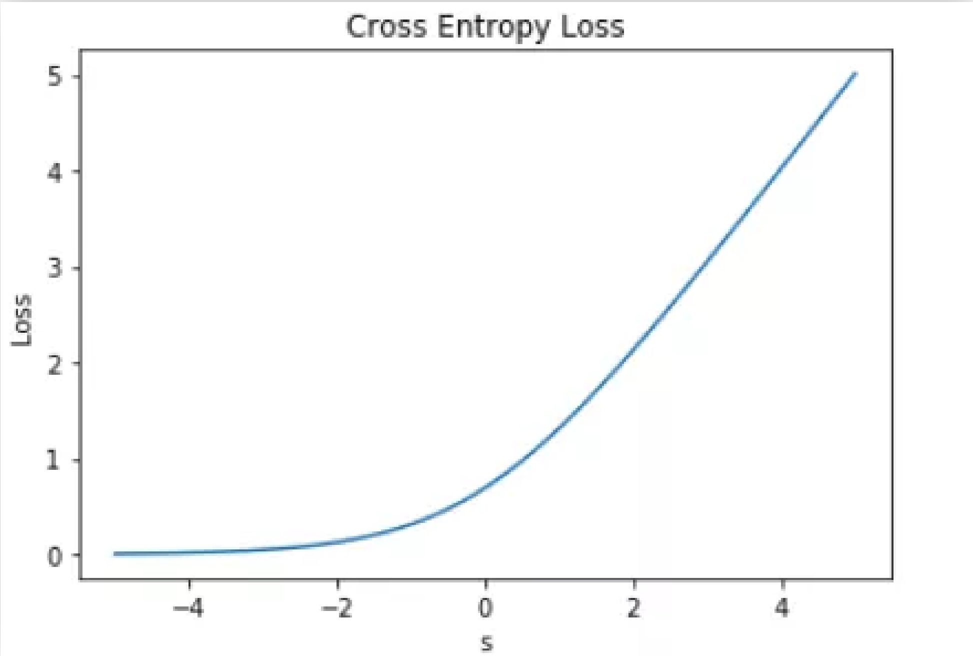
当 y = 0 时：

$$L = -log\left(1 - \hat{y}\right)$$

对上式进行整理，同样能得到：

$$L = log(1 + e^{s})$$

这时候，Loss 的曲线如下图所示：



从图中明显能够看出，s 越小于零，L 越小，函数的变化趋势也完全符合实际需要的情况。

第二种形式是基于输出标签 label 的表示方式为 {-1,+1}，也比较常见。它的 Loss 表达式为：

$$L = log(1 + e^{-ys})$$

下面对上式做个简单的推导，我们在 0 小节说过，ys 的符号反映了预测的准确性。除此之外，ys 的数值大小也反映了预测的置信程度。所以，从概率角度来看，预测类别的概率可以写成：

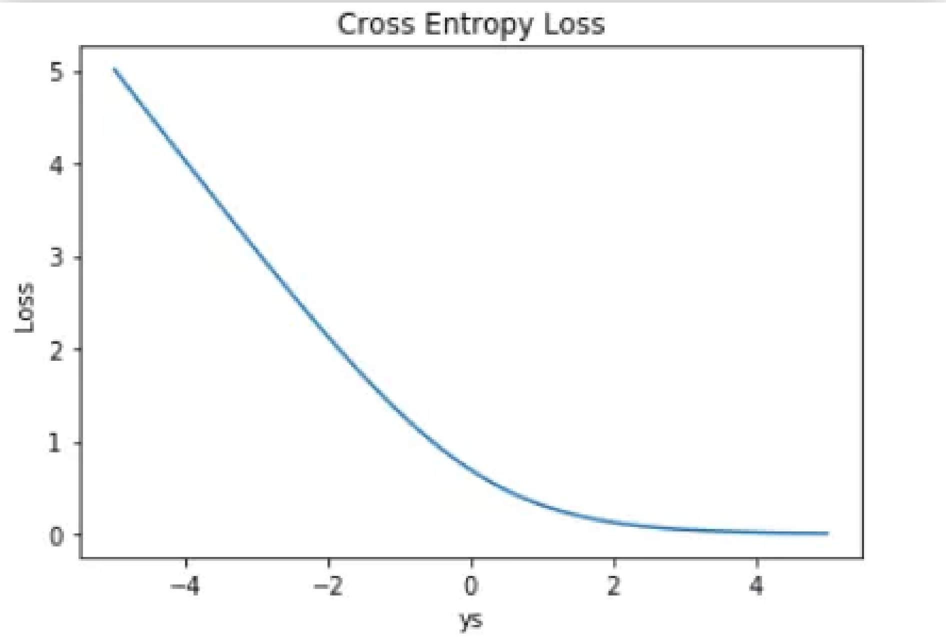
$$P(y|x) = g(ys)$$

分别令 y = +1 和 y = -1 就能很容易理解上面的式子。

接下来，同样引入 log 函数，要让概率最大，反过来，只要其负数最小即可。那么就可以定义相应的损失函数为：

$$L = -log\,g(ys) = -log\frac{1}{1+e^{-ys}} = log(1+e^{-ys})$$

这时候，我们以 ys 为横坐标，可以绘制 Loss 的曲线如下图所示：



其实上面介绍的两种形式的交叉熵 Loss 是一样的，只不过由于标签 label 的表示方式不同，公式稍有变化。标签用 {-1,+1} 表示的好处是可以把 ys 整合在一起，作为横坐标，容易作图且具有实际的物理意义。

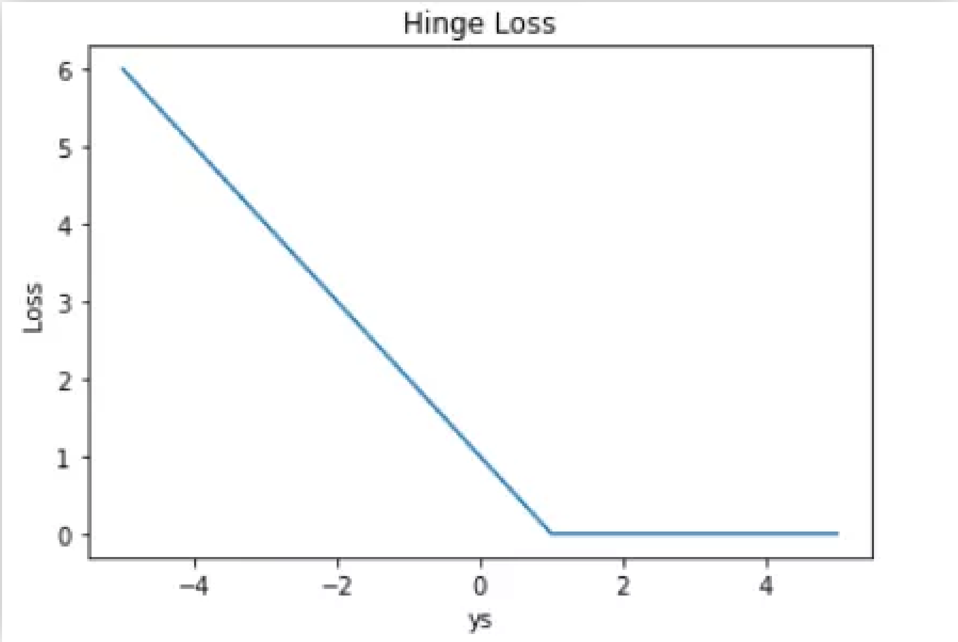
总结一下，交叉熵 Loss 的优点是在整个实数域内，Loss 近似线性变化。尤其是当 ys << 0 的时候，Loss 更近似线性。这样，模型受异常点的干扰就较小。而且，交叉熵 Loss 连续可导，便于求导计算，是使用最广泛的损失函数之一。

3.Hinge Loss

Hinge Loss，又称合页损失，其表达式如下：

$$L = max(0, 1 - ys)$$

Hinge Loss 的曲线如下图所示：



Hinge Loss 的形状就像一本要合上的书，故称为合页损失。显然，只有当 ys < 1 时，Loss 才大于零；对于 ys > 1 的情况，Loss 始终为零。

Hinge Loss 一般多用于支持向量机（SVM）中，体现了 SVM 距离最大化的思想。而且，当 Loss 大于零时，是线性函数，便于梯度下降算法求导。

Hinge Loss 的另一个优点是使得 ys > 0 的样本损失皆为 0，由此带来了稀疏解，使得 SVM 仅通过少量的支持向量就能确定最终超平面。

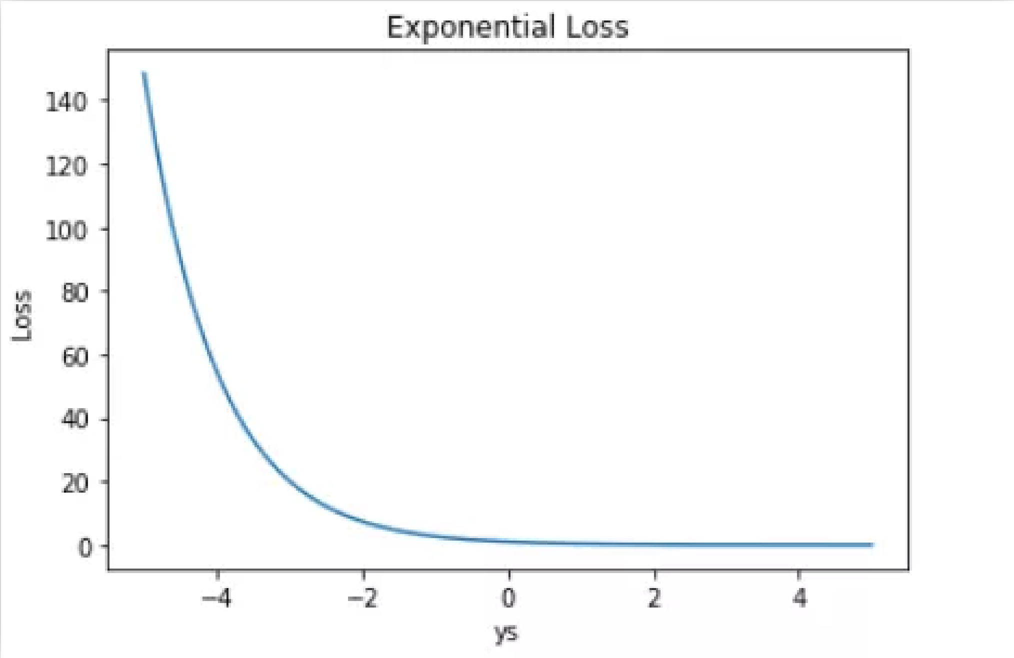
4.Exponential Loss

Exponential Loss，又称指数损失，其表达式如下：

$$L = e^{-ys}$$

可以对上式进行一个直观的理解，类似于上文提到的第二种形式的交叉熵 Loss，去掉 log 和 log 中的常数 1，并不影响 Loss 的单调性。因此，推导得出了 Exponential Loss 的表达式。

Exponential Loss 的曲线如下图所示：



Exponential Loss 与交叉熵 Loss 类似，但它是指数下降的，因此梯度较其它 Loss 来说，更大一些。

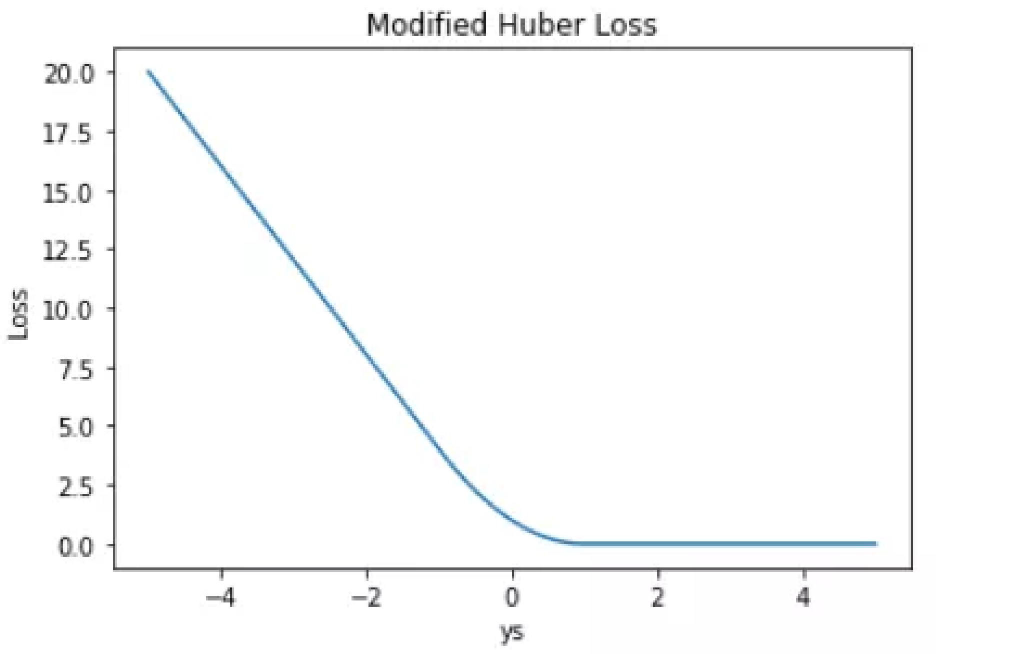
Exponential Loss 一般多用于AdaBoost 中。因为使用 Exponential Loss 能比较方便地利用加法模型推导出 AdaBoost算法。

5.Modified Huber Loss

在之前介绍回归损失函数，我们介绍过 Huber Loss，它集合了 MSE 和 MAE 的优点。Huber Loss 也能应用于分类问题中，称为 Modified Huber Loss，其表达是如下：

$$L(y,s)=\begin{cases} max(0,1-ys)^2, & ys\geq -1 \\ -4ys, & ys< -1 \end{cases}$$

Modified Huber Loss 的曲线如下图所示：



从表达式和 Loss 图形上看，Modified Huber Loss 结合了 Hinge Loss 和 交叉熵 Loss 的优点。一方面能在 ys > 1 时产生稀疏解提高训练效率；另一方面对于 ys < −1 样本的惩罚以线性增加，这意味着受异常点的干扰较少。scikit-learn 中的 SGDClassifier 就使用了 Modified Huber Loss。

6.Softmax Loss

对于多分类问题，也可以使用 Softmax Loss。

机器学习模型的 Softmax 层，正确类别对于的输出是：

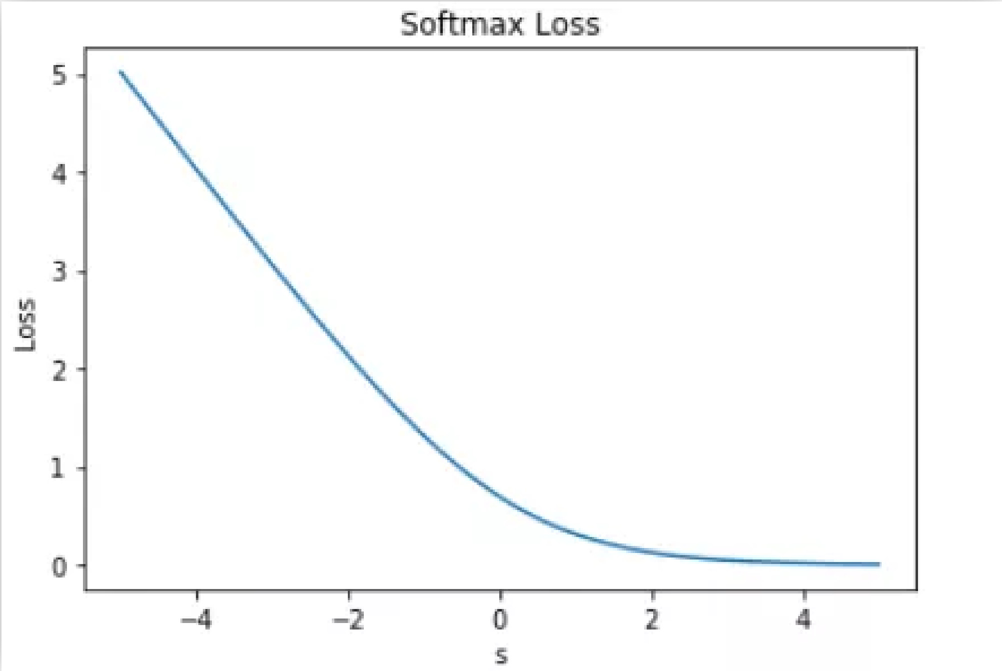
$$S=\frac{e^s}{\sum_{j=1}^{C}e^{s_j}}$$

其中，C 为类别个数，小写字母 s 是正确类别对应的 Softmax 输入，大写字母 S 是正确类别对应的 Softmax 输出。

由于 log 运算符不会影响函数的单调性，我们对 S 进行 log 操作。另外，我们希望 log(S) 越大越好，即正确类别对应的相对概率越大越好，那么就可以对 log(S) 前面加个负号，来表示损失函数：

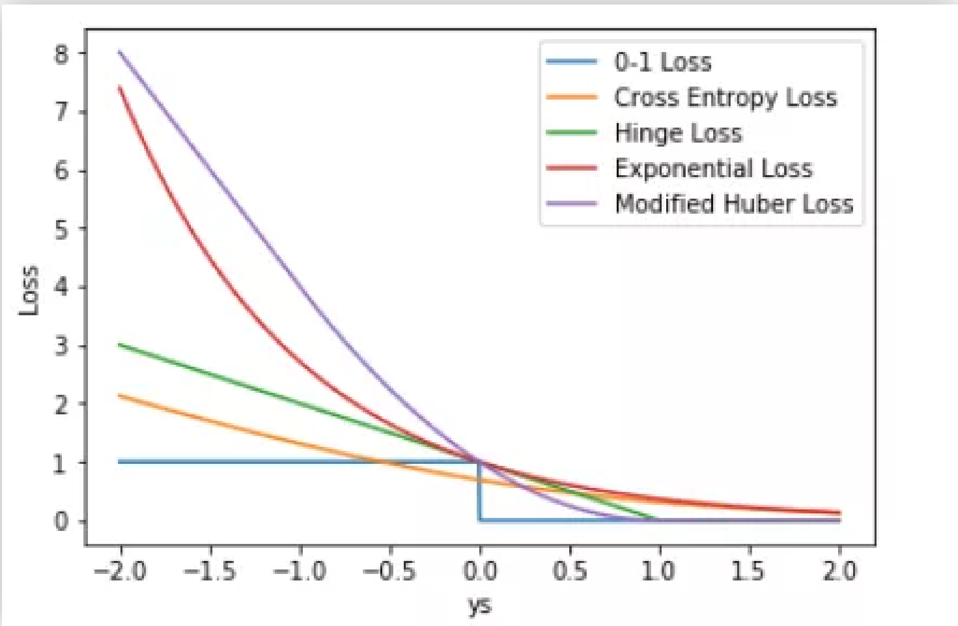
$$L=-log\frac{e^s}{\sum_{j=1}^{C}e^{s_j}}=-s+log\sum_{j=1}^{C}e^{s_j}$$

Softmax Loss 的曲线如下图所示：

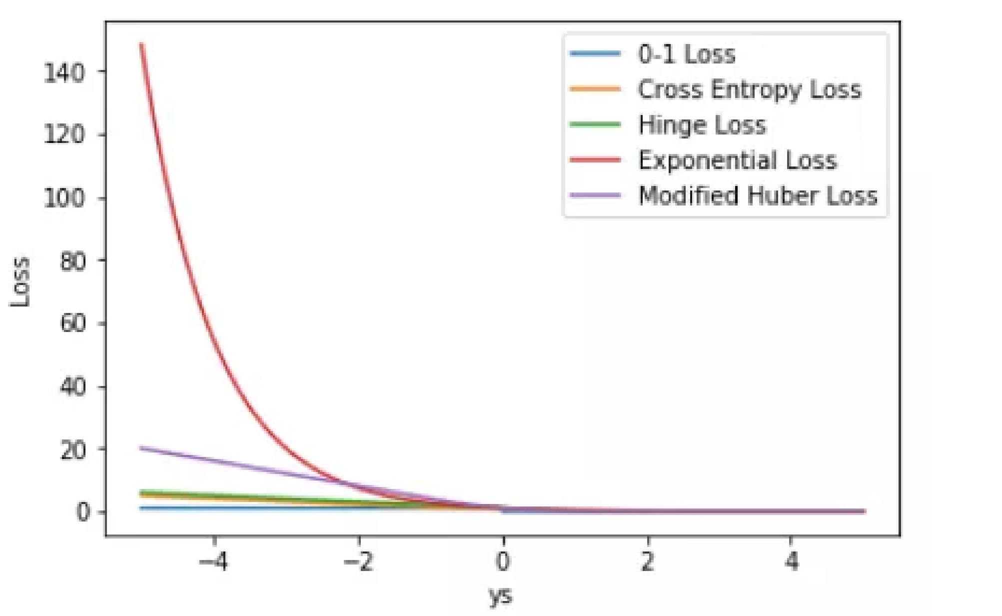


上图中，当 s << 0 时，Softmax 近似线性；当 s>>0 时，Softmax 趋向于零。Softmax 同样受异常点的干扰较小，多用于神经网络多分类问题中。

最后，我们将 0-1 Loss、Cross Entropy Loss、Hinge Loss、Exponential Loss、Modified Huber Loss 画在一张图中：



上图 ys 的取值范围是 [-2,+2]，若我们把 ys 的坐标范围取得更大一些，上面 5 种 Loss 的差别会更大一些，见下图：



显然，这时候 Exponential Loss 会远远大于其它 Loss。从训练的角度来看，模型会更加偏向于惩罚较大的点，赋予其更大的权重。如果样本中存在离群点，Exponential Loss 会给离群点赋予更高的权重，但却可能是以牺牲其他正常数据点的预测效果为代价，可能会降低模型的整体性能，使得模型不够健壮（robust）。

相比 Exponential Loss，其它四个 Loss，包括 Softmax Loss，都对离群点有较好的“容忍性”，受异常点的干扰较小，模型较为健壮。

好了，以上就是总结的几个常用的损失函数，知道这些细节和原理，对我们是很有帮助的。希望本文对你有所帮助 ~

参考文献：

http://www.10tiao.com/html/782/201806/2247495489/1.html
https://www.cnblogs.com/massquantity/p/8964029.html

备注：公众号菜单包含了整理了一本AI小抄，非常适合在通勤路上用学习。

往期精彩回顾

- 那些年做的学术公益-你不是一个人在战斗
- 适合初学者入门人工智能的路线及资料下载
- 机器学习在线手册
- 深度学习在线手册

备注：加入本站微信群或者qq群，请回复“加群”

加入知识星球（4500+用户，ID：92416895），请回复“知识星球”

喜欢文章，点个在看