# pairwise loss

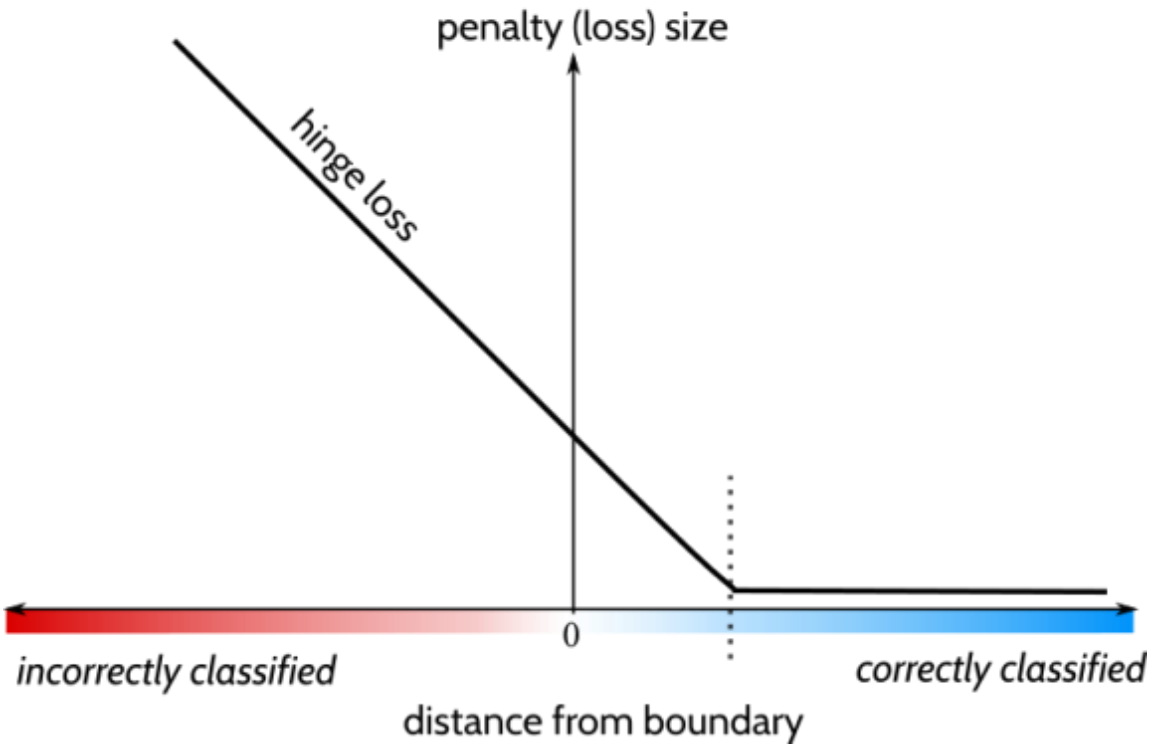
## 1 max_margin_loss(hinge loss)

**二分类下的基本形式**:

$$L(\hat{y}) = max(0, 1 - y * \hat{y}) \tag{1}$$

一般用于二分类。如果真实标签$y$取值是1的话，公式退化为

$$\begin{aligned} L(\hat{y}) &= max(0, 1 - \hat{y}) \\ &= \begin{cases} 1 - \hat{y} & \hat{y} < 1, \\ 0 & \hat{y} >= 1 \end{cases} \end{aligned} \tag{2}$$




即把1当作一个阈值。如果预测值超过这个阈值，认为该样本已经足够好，就不需要再去学习了。而是专注于学习那些预测值还没有达到该阈值的样本。尽量让预测值去逼近这个阈值，学习该预测值和阈值之间的差值。因为该损失像一本打开的书，所以称为合页损失函数。

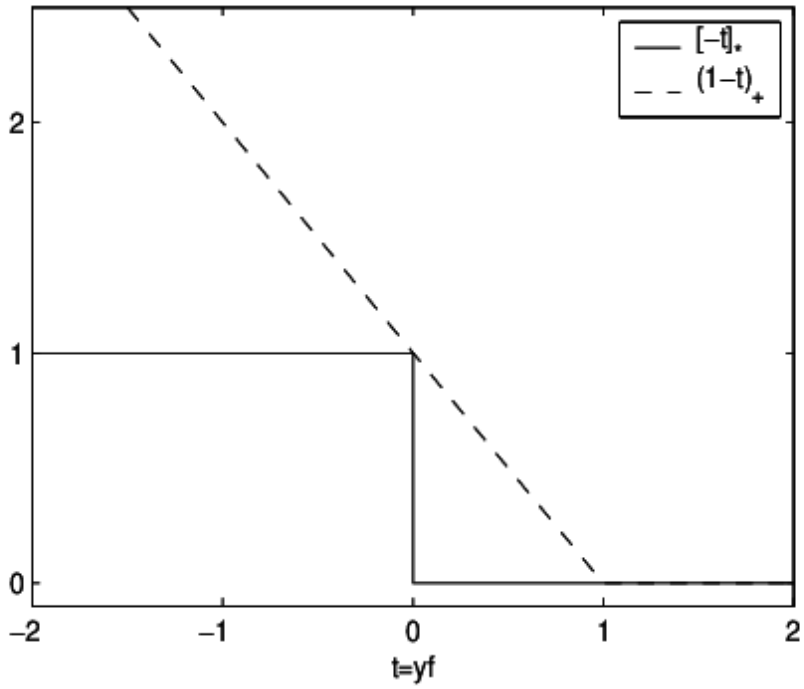
负样本场景同理 。都是只要达到这个阈值了就不去再学习。只有预测值不达标的样本才会产生损失。本质都是让预测值从一侧接近我们设定的阈值。

**更通用的形式**

$$L(\hat{y}) = max(0, 1 - z) \tag{3}$$

本质上，hinge loss学习的是一个分数与我们指定的阈值之间的差距。如果已经超过了我们设定的这个"足够好"的标准，就不需要再去学习了。否则需要去学习不满足的那部分：阈值 -当前分数。

优点1： 相比0/1损失, 会对未满足阈值的样本进一步学习。




缺点1：只会对未满足阈值的样本学习。对满足阈值或者超过阈值的样本，没有足够的限制。因此需要很好的定义分数和阈值。

可以通过灵活调整阈值和分数z的定义来满足不同的使用场景。

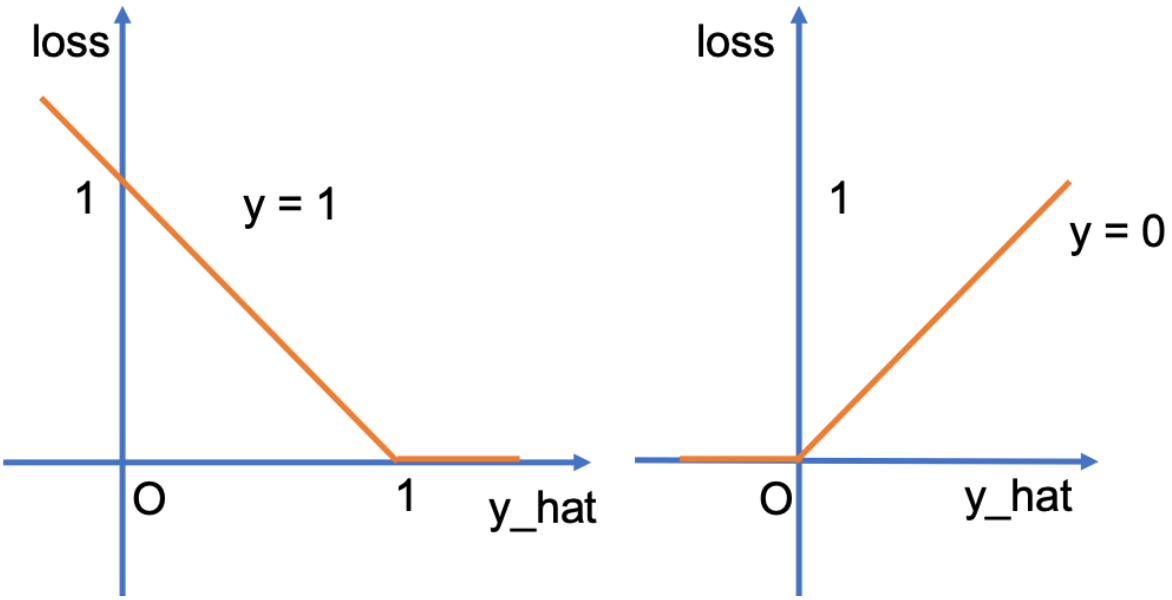
ex: 认为分数大于0就足够好，可以把阈值设置成0。

ex: 二分类场景标签取+1/-1的场景，可以定义分数是$z = y * \hat{y}$, 反应预测值和真值的相似程度。

ex:二分类标签取1/0的场景，可以定义阈值z是0。当y=1时，定义预测值大于y就足够好了。y=0时，定义预测值小于0就足够好了。即分别定义了2种条件下的z。可统一写做：

$$z = \begin{cases} \hat{y} - y & y = 1, \\ y - \hat{y} & y = 0 \end{cases} = sign(y) * (\hat{y} - y) \tag{4}$$

按阈值取后，损失函数如图：
$L(\hat{y}) = max(0, 0 - z) = max(0, -z)$。相比标签为+-1的场景，预测值的阈值变成了0。




**pair-wise形式**

在推荐等rank场景下，常常更关注2个样本的相对大小而非绝对大小。因此在仍然用hinge loss作为损失函数的情况下，一般用正负样本之间的差距作为我们学习的目标，希望正样本的分值比负样本的分值高。因此定义分值$z = s_+ - s_-$。对于选定的阈值margin，hinge loss可以写做：

$$\begin{aligned} L(\hat{y}) &= max(0, margin - z) \\ &= max(0, margin - (s_+ - s_-)) \\ &= max(0, margin + s_- - s_+)) \end{aligned} \tag{5}$$

当正样本的分值比负样本的分值，超过设定的阈值margin后，模型会认为该pair已经能够较好的区分正负样本了。因此不再对这个pair进行学习了。 因为该损失函数是在最大化正负样本的分数间隔（直到满足阈值为止），因此也被叫做max margin loss。在pair-wise的学习中比较常见。较著名的比如TransE、Pinsage等算法都采用了该损失函数，可以使正负样本有较好的区分度。

## 2 BPR loss

另一个很类似的算法是BRP loss,相当于交叉熵的pair-wise版本。同样是基于正负样本之间的分数差距来计算损失

二分类场景下，交叉熵损失函数是

$$L(z) = -\sum_i y_i log(p_i(z)) \tag{6}$$

对每个正样本而言，$p_i(z)$是该样本取值为1的概率。从极大似然的角度看，希望最大化样本取值是1的概率。其中$z$是模型直接输出的样本的分数。通过sigmoid函数转为概率：

$$p(y_i = 1) = sigmoid(z) = \frac{1}{1 + e^{-z}} \tag{7}$$

所以在CTR场景下，我们优化的是单样本的直接输出$z$，比如wx等。

而在rank等场景下。我们更关注正负样本输出分值间的差距，而不是单样本绝对分值本身。因此对一对正负样本$(x_+, x_-)$而言，我们重新通过正负样本间的分差，重新定义概率为正样本分值大于负样本分值的概率：

$$p(s_+ > s_-) = sigmoid(z) = \frac{1}{1 + e^{-z}} = \frac{1}{1 + e^{-(s_+ - s_-)}} \tag{8}$$

其中$z$是同一模型对正负样本打分之差：$z = s_+ - s_-$。因此对一个样本对，最终的pair-wise loss(即BPR loss)可以写作：

$$L(z) = -log(p(z)) \tag{9}$$

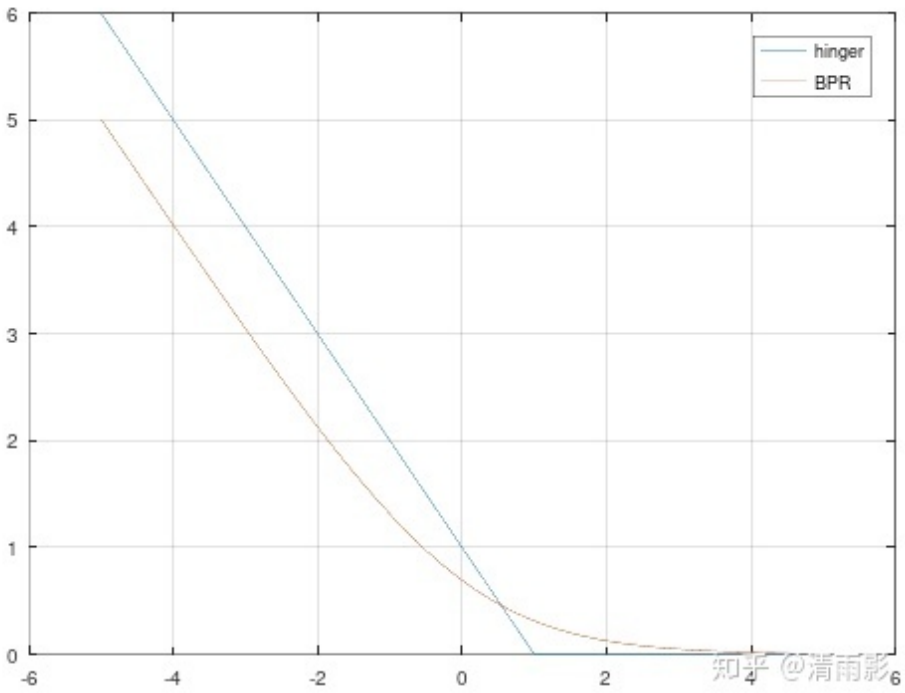
$$= -log(sigmoid(s_+ - s_-))$$

由于在推荐场景中，未点过的不一定是明确的0，直接把其学习未学习较强的负信号不一定合适。因此可以将样本组织成正负样本对的形式。看过的相对没看过的，分数会高一点。比较符合负样本不太明确的场景。因此BPR Loss被广泛应用于业界的推荐系统

## 3 对比

对比公式（5），(9)可以发现，这两个pair-wise loss分别是hinge_loss和交叉熵损失的pair-wise版本。由于这2个loss本身比较接近,因此效果上不会差特别多。

```
z = [-5:0.01:5];
plot(z,max(0,1-z),z,log(1+exp(-z)))
legend('hinger','BPR')
```




相对来说，hinger loss可以通过设置不同阈值来控制想要的正负样本分差，可控性更强，而且计算简单。

而bpr-loss则鼓励正负样本间差距越大越好，在分差较大时。loss也不完全为0，对超过阈值的部分仍有约束。整体上会使正负样本间分差尽量拉大。

选择哪个，可以根据原理，结合实验和AB进行验证。