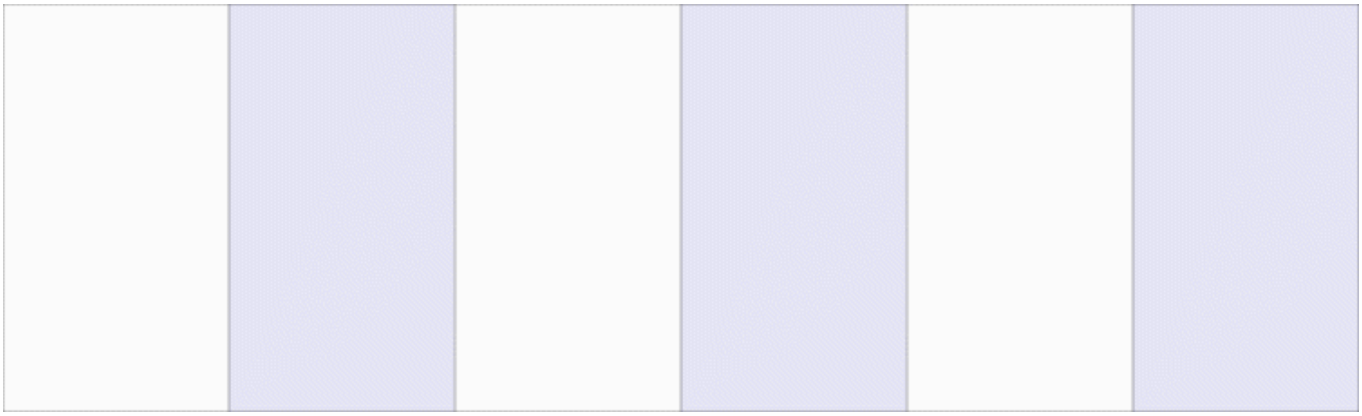
损失函数技术总结及Pytorch使用示例

极市平台 2023-01-02 22:00:36 发表于广东 手机阅读

以下文章来源于CV技术指南 ，作者仿佛若有光



作者丨仿佛若有光

来源丨CV技术指南

编辑丨极市平台

极市导读

本文对损失函数的类别和应用场景，常见的损失函数，常见损失函数的表达式，特性，应用场景和使用示例作了详细的总结。 >>加入极市CV技术交流群，走在计算机视觉的最前沿

前言

一直想写损失函数的技术总结，但网上已经有诸多关于损失函数综述的文章或博客，考虑到这点就一直拖着没写，直到有一天，我将一个二分类项目修改为多分类，简简单单地修改了损失函数，结果一直有问题，后来才发现是不同函数的标签的设置方式并不相同。

为了避免读者也出现这样的问题，本文中会给出每个损失函数的pytorch使用示例，这也是本文与其它相关综述文章或博客的区别所在。希望读者在阅读本文时，重点关注一下每个损失函数的使用示例中的target的设置问题。

本文对损失函数的类别和应用场景，常见的损失函数，常见损失函数的表达式，特性，应用场景和使用示例作了详细的总结。

主要涉及到L1 loss、L2 loss、Negative Log-Likelihood loss、Cross-Entropy loss、Hinge Embedding loss、Margin Ranking Loss、Triplet Margin loss、KL Divergence.

损失函数分类与应用场景

损失函数可以分为三类：回归损失函数(Regression loss)、分类损失函数(Classification loss)和排序损失函数(Ranking loss)。

应用场景：回归损失：用于预测连续的值。如预测房价、年龄等。分类损失：用于预测离散的值。如图像分类，语义分割等。排序损失：用于预测输入数据之间的相对距离。如行人重识别。

L1 loss

也称Mean Absolute Error，简称MAE，计算实际值和预测值之间的绝对差之和的平均值。

表达式如下：

$$\text{Loss}\left(\text{pred}, \text{y}\right) = \left|\text{y} - \text{pred}\right|$$

y表示标签，pred表示预测值。

应用场合：回归问题。

根据损失函数的表达式很容易了解它的特性：当目标变量的分布具有异常值时，即与平均值相差很大的值，它被认为对异常值具有很好的鲁棒行。

使用示例：

```
input = torch.randn(3, 5, requires_grad=True)
target = torch.randn(3, 5)

mae_loss = torch.nn.L1Loss()
output = mae_loss(input, target)
```

L2 loss

也称为Mean Squared Error，简称MSE，计算实际值和预测值之间的平方差的平均值。

表达式如下：

$$\text{Loss}(\text{ pred}, y) = \sum(\text{y} - \text{ pred })^2$$

应用场合：对大部分回归问题，pytorch默认使用L2，即MSE。

使用平方意味着当预测值离目标值更远时在平方后具有更大的惩罚，预测值离目标值更近时在平方后惩罚更小，因此，当异常值与样本平均值相差格外大时，模型会因为惩罚更大而开始偏离，相比之下，L1对异常值的鲁棒性更好。

使用示例：

```
input = torch.randn(3, 5, requires_grad=True)
target = torch.randn(3, 5)
mse_loss = torch.nn.MSELoss()
output = mse_loss(input, target)
```

Negative Log-Likelihood

简称NLL。表达式如下：

$$\text{loss}(\text{ pred}, \text{y}) = -(\log\ \text{pred }) $$

应用场景：多分类问题。

注：NLL要求网络最后一层使用softmax作为激活函数。通过softmax将输出值映射为每个类别的概率值。

根据表达式，它的特性是惩罚预测准确而预测概率不高的情况。

NLL 使用负号，因为概率（或似然）在 0 和 1 之间变化，并且此范围内的值的对数为负。最后，损失值变为正值。

在 NLL 中，最小化损失函数有助于获得更好的输出。从近似最大似然估计 (MLE) 中检索负对数似然。这意味着尝试最大化模型的对数似然，从而最小化 NLL。

使用示例

```
# size of input (N x C) is = 3 x 5
input = torch.randn(3, 5, requires_grad=True)
# every element in target should have 0 <= value < C
target = torch.tensor([1, 0, 4])

m = nn.LogSoftmax(dim=1)
```

```
nll_loss = torch.nn.NLLLoss()
output = nll_loss(m(input), target)
```

Cross-Entropy

此损失函数计算提供的一组出现次数或随机变量的两个概率分布之间的差异。它用于计算预测值与实际值之间的平均差异的分数。

表达式：

$$\text{loss}\left(\text{ pred}, y\right)=-\sum \text{y log pred}$$

应用场景：二分类及多分类。

特性：负对数似然损失不对预测置信度惩罚，与之不同的是，交叉熵惩罚不正确但可信的预测，以及正确但不太可信的预测。

交叉熵函数有很多种变体，其中最常见的类型是Binary Cross-Entropy (BCE)。BCE Loss 主要用于二分类模型；也就是说，模型只有 2 个类。

使用示例

```
input = torch.randn(3, 5, requires_grad=True)
target = torch.empty(3, dtype=torch.long).random_(5)

cross_entropy_loss = torch.nn.CrossEntropyLoss()
output = cross_entropy_loss(input, target)
```

Hinge Embedding

表达式：

$$\text{loss}(\text{ pred}, y)=\max(0,1-y*\text{ pred})$$

其中y为1或-1。

应用场景：

分类问题，特别是在确定两个输入是否不同或相似时。

学习非线性嵌入或半监督学习任务。

使用示例

```
input = torch.randn(3, 5, requires_grad=True)
target = torch.randn(3, 5)

hinge_loss = torch.nn.HingeEmbeddingLoss()
output = hinge_loss(input, target)
```

Margin Ranking Loss

Margin Ranking Loss 计算一个标准来预测输入之间的相对距离。这与其他损失函数（如 MSE 或交叉熵）不同，后者学习直接从给定的输入集进行预测。

表达式：

$$\text{loss}(\text{ pred, y}) = \max(0, -\text{y} * \text{ (pred1 - pred2)} + \text{ margin })$$

标签张量 y（包含 1 或 -1）。当 y == 1 时，第一个输入将被假定为更大的值。它将排名高于第二个输入。如果 y == -1，则第二个输入将排名更高。

应用场景：排名问题

使用示例

```
input_one = torch.randn(3, requires_grad=True)
input_two = torch.randn(3, requires_grad=True)
target = torch.randn(3).sign()

ranking_loss = torch.nn.MarginRankingLoss()
output = ranking_loss(input_one, input_two, target)
```

Triplet Margin Loss

计算三元组的损失。

表达式：

$$\text{Loss }(a, p, n) = \max\{0, d(ai, pi) - d(ai, ni) + \text{ margin }\}$$

三元组由a (anchor)，p (正样本) 和 n (负样本)组成.

应用场景：

确定样本之间的相对相似性

用于基于内容的检索问题

使用示例

```
anchor = torch.randn(100, 128, requires_grad=True)
positive = torch.randn(100, 128, requires_grad=True)
negative = torch.randn(100, 128, requires_grad=True)

triplet_margin_loss = torch.nn.TripletMarginLoss(margin=1.0, p=2)
output = triplet_margin_loss(anchor, positive, negative)
```

KL Divergence Loss

计算两个概率分布之间的差异。

表达式：

$$\text{loss}\left(\text{pred}, y\right) = y * \left(\log y - \text{ pred }\right)$$

输出表示两个概率分布的接近程度。如果预测的概率分布与真实的概率分布相差很远，就会导致很大的损失。如果 KL Divergence 的值为零，则表示概率分布相同。

KL Divergence 与交叉熵损失的关键区别在于它们如何处理预测概率和实际概率。交叉熵根据预测的置信度惩罚模型，而 KL Divergence 则没有。KL Divergence 仅评估概率分布预测与ground truth分布的不同之处。

应用场景：逼近复杂函数多类分类任务确保预测的分布与训练数据的分布相似

使用示例

```
input = torch.randn(2, 3, requires_grad=True)
target = torch.randn(2, 3)

kl_loss = torch.nn.KLDivLoss(reduction = 'batchmean')
output = kl_loss(input, target)
```

原文链接：https://neptune.ai/blog/pytorch-loss-functions
本文在此链接的基础上进行一部分而来修改。

极市干货

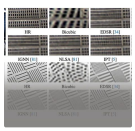
技术干货：数据可视化必须注意的30个小技巧总结 | 如何高效实现矩阵乘？万文长字带你从CUDA初学者的角度入门

实操教程：Nvidia Jetson TX2使用TensorRT部署yolov5s模型｜基于YOLOV5的数据集标注&训练，Windows/Linux/Jetson Nano多平台部署全流程

喜欢此内容的人还喜欢

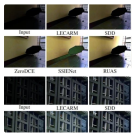
ICCV 2023｜南开程明明团队提出适用于SR任务的新颖注意力机制（已开源）

极市平台

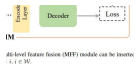


ICCV23｜将隐式神经表征用于低光增强，北大张健团队提出NeRCo

极市平台



ICCV 2023｜Pixel-based MIM: 简单高效的多级特征融合自监督方法



IM

dit-level feature fusion (MFF) module can be insertec