问题

看到一句话：NMS都不懂，还做什么Detection！ 虎躯一震.....懂是大概懂，但代码能写出来吗？？？

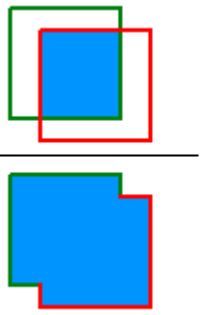
在目标检测网络中，产生 proposal 后使用分类分支给出每个框的每类置信度，使用回归分支修正框的位置，最终会使用 NMS 方法去除**同个类别**当中 IOU 重叠度较高且 scores 即置信度较低的那些检测框。

下图就是在目标检测中 NMS 的使用效果：emmm大概就是能让你更无遮挡地看到美女的脸吧hhhh

背景知识

NMS (Non-maximum suppression) 非极大值抑制，即抑制不是极大值的检测框，根据什么去抑制？在目标检测领域，当然是根据 IOU (Intersection over Union) 去抑制。下图是绿色检测框与红色检测框的 IOU 计算方法：

$$IOU = \frac{\text{area of overlap}}{\text{area of union}} =$$



NMS 原理及示例

注意 NMS 是针对一个特定的类别进行操作的。例如假设一张图中有要检测的目标有“人脸”和“猫”，没做NMS之前检测到10个目标框，每个目标框变量表示为: $[x_1, y_1, x_2, y_2, score_1, score_2]$ ，其中 (x_1, y_1) 表示该框左上角坐标，(x_2, y_2) 表示该框右下角坐标，$score_1$ 表示"人脸"类别的置信度，$score_2$ 表示"猫"类别的置信度。当 $score_1$ 比 $score_2$ 大时，将该框归为“人脸”类别，反之归为“猫”类别。最后我们假设10个目标框中有6个被归类为“人脸”类别。

接下来演示如何对“人脸”类别的目标框进行 NMS 。

首先对6个目标框按照 $score_1$ 即置信度降序排序：

目标框	score_1
A	0.9
B	0.85
C	0.7
D	0.6
E	0.4
F	0.1

(1) 取出最大置信度的那个目标框 A 保存下来
(2) 分别判断 B-F 这 5 个目标框与 A 的重叠度 IOU ，如果 IOU 大于我们预设的阈值（一般为 0.5），则将该目标框丢弃。假设此时丢弃的是 C和 F 两个目标框，这时候该序列中只剩下 B D E 这三个。
(3) 重复以上流程，直至排序序列为空。

代码实现

```
# bboxees维度为 [N, 4]，scores维度为 [N, 1]，均为np.array()
def single_nms(self, bboxes, scores, thresh = 0.5):
    # x1、y1、x2、y2以及scores赋值
    x1 = bboxes[:, 0]
    y1 = bboxes[:, 1]
    x2 = bboxes[:, 2]
    y2 = bboxes[:, 3]

    # 计算每个检测框的面积
    areas = (x2 - x1 + 1) * (y2 - y1 + 1)

    # 按照 scores 置信度降序排序，order 为排序的索引
    order = scores.argsort() # argsort为python中的排序函数，默认升序排序
    order = order[::-1] # 将升序结果翻转为降序

    # 保留的结果框索引
    keep = []

    # torch.numel() 返回张量元素个数
    while order.size > 0:
        if order.size == 1:
            i = order[0]
            keep.append(i)
            break
        else:
            i = order[0]  # 在pytorch中使用item()来取出元素的实值，即若只是 i = order[0]，此时的 i 还是一个 tensor，因此不能赋值给 keep
            keep.append(i)

        # 计算相交区域的左上坐标及右下坐标
        xx1 = np.maximum(x1[i], x1[order[1:]])
        yy1 = np.maximum(y1[i], y1[order[1:]])
        xx2 = np.minimum(x2[i], x2[order[1:]])
        yy2 = np.minimum(y2[i], y2[order[1:]])

```

```
        # 计算相交的面积，不重叠时为0
        w = np.maximum(0.0, xx2 - xx1 + 1)
        h = np.maximum(0.0, yy2 - yy1 + 1)
        inter = w * h

        # 计算 IOU = 重叠面积 / (面积1 + 面积2 - 重叠面积)
        iou = inter / (areas[i] + areas[order[1:]] - inter)

        # 保留 IOU 小于阈值的 bboxes
        inds = np.where(iou <= thresh)[0]
        if inds.size == 0:
            break
        order = order[inds + 1] # 因为我们上面求iou的时候得到的结果索引与order相比偏移了一位，因此这里要补回来
    return keep  # 这里返回的是bboxes中的索引，根据这个索引就可以从bboxes中得到最终的检测框结果
```

参考资料

[NMS算法详解（附Pytorch实现代码）]
[非极大值抑制（Non-Maximum Suppression，NMS）]

By Yee

2020.05.16