计算机视觉作业 -6：目标检测

王天锐　20120318

2021 年 1 月 19 日

摘要

本次实验作业主要是利用机器学习方法（分类）实现静态场景下的测视车辆检测。Data 文件夹中包含 train_34x94（训练集）和 test（测试集）两个文件夹。其中，train_34x94 文件夹中的数据用于训练模型，包含 pos 文件夹（内有 550 个正例样本）和 neg 文件夹（内有 500 个负例样本）；Test 文件夹中的数据用于测试。在 Test 测试集中的总体检测性能的评价指标为 Recall、Precision 和 F-measure。

1　方法原理及实现

1.1　HOG 特征

方向梯度直方图（Histogram of Oriented Gradient, HOG）特征是一种在计算机视觉和图像处理中用来进行物体检测的特征描述子。HOG 特征通过计算和统计图像局部区域的梯度方向直方图来构成特征。在一副图像中，局部目标的表象和形状能够被梯度或边缘的方向密度分布很好地描述。其本质为：梯度的统计信息，而梯度主要存在于边缘的地方。

本实验 HOG 特征用 opencv 库中的 HOGDescriptor 函数实现。整体代码如下:

```
winSize = (94, 34)
blockSize = (8, 8)
blockStride = (2, 2)
cellSize = (4, 4)
nbins = 11
hog = cv2.HOGDescriptor(winSize, blockSize, blockStride, cellSize, nbins)
train_data = []
winStride = (0, 0)
for path in train_path:
        img = cv2.imread(path)
        fea = hog.compute(img, winStride)
```

这里有几个重要的参数要研究下：winSize(64,128), blockSize(16,16), blockStride(8,8), cellSize(8,8), nbins(9)。上面这些都是 HOGDescriptor 的成员变量，括号里的数值是它们的默认值，它们反应了 HOG 描述子的参数。nBins 表示在一个胞元 (cell) 中统计梯度的方向数目，例如 nBins=9 时，在一个胞元内统计 9 个方向的梯度直方图，每个方向为 180/9=20 度。这里做了几个示意图来解释另几个参数的含义。

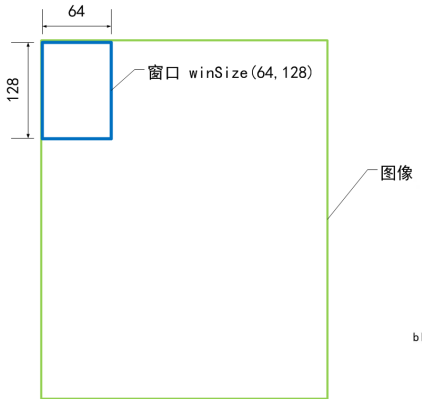

图 1: WinSize 参数

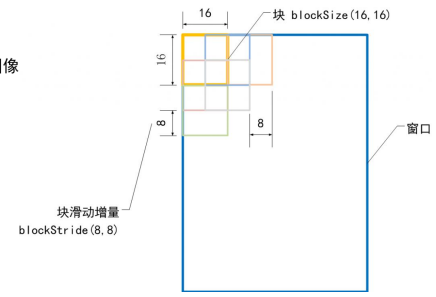

图 2: BlockSize、BlockStride 参数

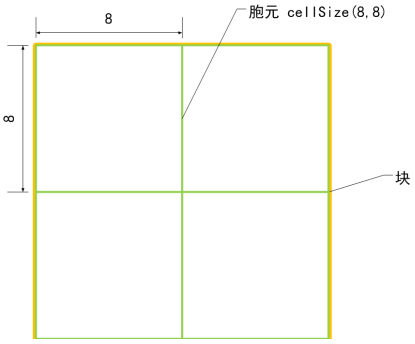

图 3: CellSize 参数

1.2 主成成分分析 (PCA)

主成成分分析 (Principal Component Analysis) 的主要思想是将 n 维特征映射到 k 维上，k 维是全新的正交特征，并且 k 可以远小于 n。本实验使用的是基于特征值分解协方差矩阵实现的 PCA 算法。对于输入的数据集 $X_{m\times n}$ 我们需要将其降到 k 维得到 $X^*_{m\times k}$。第一步处理是中心化操作:

$$X_{ij} = X_{ij} - \overline{X_j}$$

然后计算出中心化后 A 的协方差矩阵

$$\begin{aligned} C &= E[(X - E[X])(X - E[X])^T] \\ &= \begin{bmatrix} conv(X_{i1}, X_{i1}) & conv(X_{i1}, X_{i2}) & \cdots & conv(X_{i1}, X_{in}) \\ conv(X_{i2}, X_{i1}) & conv(X_{i2}, X_{i2}) & \cdots & conv(X_{i2}, X_{in}) \\ \vdots & \vdots & \ddots & \vdots \\ conv(X_{in}, X_{i1}) & conv(X_{in}, X_{i2}) & \cdots & conv(X_{in}, X_{in}) \end{bmatrix} \end{aligned}$$

然后计算协方差矩阵的特征值:

$$det(\lambda E - X) = 0$$

$$\begin{vmatrix} \lambda - conv(X_{i1}, X_{i1}) & conv(X_{i1}, X_{i2}) & \cdots & conv(X_{i1}, X_{in}) \\ conv(X_{i2}, X_{i1}) & \lambda - conv(X_{i2}, X_{i2}) & \cdots & conv(X_{i2}, X_{in}) \\ \vdots & \vdots & \ddots & \vdots \\ conv(X_{in}, X_{i1}) & conv(X_{in}, X_{i2}) & \cdots & \lambda - conv(X_{in}, X_{in}) \end{vmatrix} = 0$$

通过上式解出特征值 λ, 然后将特征值带入:

$$(C - \lambda I)\alpha = 0$$

解出特征向量 α, 然后将特征向量和特征值按特征值大小进行排序，选取最大的 k 个特征向量 α_1、α_2、...、α_k。

最后利用选出的特征向量构成转换矩阵 U 作用于原数据矩阵, 得出结果 Z:

$$Z = X^T U$$

整个 PCA 的代码实现为:

```
pca = PCA(n_components=300)
pca.fit(train_data)
pca.transform(fd)
```

1.3 Logistic 回归分类器

事件的几率 (odds)，是指该事件发生的概率与该事件不发生的概率的比值。如果事件发生的概率是 p，那么该事件的几率是 p/(1-p)。取该事件发生几率的对数，定义为该事件的对数几率（log odds）或 logit 函数：

$$logit(p) = log\frac{p}{1-p}$$

事件发生的概率 p 的取值范围为 [0,1]，对于这样的输入，计算出来的几率只能是非负的（大家可以自己验证），而通过取对数，便可以将输出转换到整个实数范围内:

$$logit(p(y=1|x)) = w_0x_0 + w_1x_1 + \cdots + w_nx_n = \sum_{i=0}^{n} w_ix_i = w^Tx$$

其中，$p(y=1|x)$ 是条件概率分布，表示当输入为 x 时，实例被分为 1 类的概率，依据此概率我们能得到事件发生的对数几率。但是，我们的初衷是做分类器，简单点说就是通过输入特征来判定该实例属于哪一类别或者属于某一类别的概率。所以我们取 logit 函数的反函数，令 w^Tx 的线性组合为输入，p 为输出，经如下推导:

$$log\frac{p}{1-p} = w^Tx$$

令 $w^Tx = z$，对上述公式取反得到:

$$\theta(z) = p = \frac{1}{1+e^{-z}}$$

即可得到 logistic 函数。也成为 sigmoid 函数。

令优化目标函数为:

$$J(w) = log(L(w)) = \sum_{i=1}^{n} -y^{(i)}log(\theta(z^{(i)})) - (1-y^{(i)})log(1-\theta(z^{(i)}))$$

然后可以利用梯度下降法对参数进行求取。

本实验利用 Sklearn 库对该算法进行实现，代码如下:

```
# Logistic
clf = LogisticRegression()
clf.fit(train_data, train_label)
clf.predict_proba(pca.transform(fd))
```

1.4 非极大值抑制

非极大值抑制（Non-Maximum Suppression，NMS），顾名思义就是抑制不是极大值的元素，可以理解为局部最大搜索。这个局部代表的是一个邻域，邻域有两个参数可变，一是邻域的维数，二是邻域的大小。例如在行人检测中，滑动窗口经提取特征，经分类器分类识别后，每个窗口都会得到一个分数。但是滑动窗口会导致很多窗口与其他窗口存在包含或者大部分交叉的情况。这时就需要用到 NMS 来选取那些邻域里分数最高（是行人的概率最大），并且抑制那些分数低的窗口。NMS 在计算机视觉领域有着非常重要的应用，如视频目标跟踪、数据挖掘、3D 重建、目标识别以及纹理分析等。如下图所示。

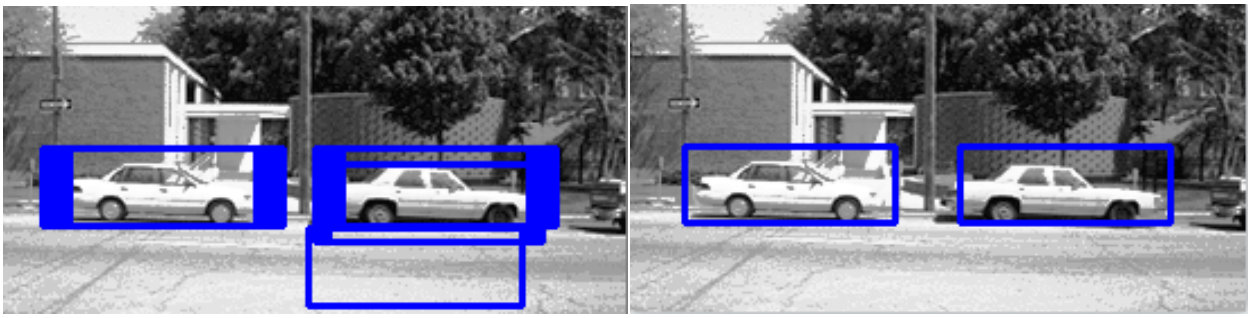

图 4: 非极大值抑制示意图

我们的目的就是要去除冗余的检测框, 保留最好的一个。
首先第一步是计算交并比，即两边界框相交部分面积与相并部分面积之比:

$$IOU = \frac{\text{相交面积}}{\text{相并面积}}$$

整体算法流程为:

(1) 将所有的框按类别划分，并剔除背景类，因为无需 NMS。

(2) 对每个物体类中的边界框 (BBOX)，按照分类置信度降序排列。

(3) 在某一类中，选择置信度最高的边界框 BBOX1，将 BBOX1 从输入列表中去除，并加入输出列表。

(4) 逐个计算 BBOX1 与其余 BBOX2 的交并比 IoU，若 IoU(BBOX1,BBOX2)> 阈值 TH，则在输入去除 BBOX2。

(5) 重复步骤 3 4，直到输入列表为空，完成一个物体类的遍历。

(6) 重复 2 5，直到所有物体类的 NMS 处理完成。

(7) 输出列表，算法结束。

整体代码实现如下:

```
def overlapping_area(result_1, result_2):
        """
        IOU
        """
        x_1, y_1 = result_1[0], result_1[1]
        x_2, y_2 = result_2[0], result_2[1]
        x1_br = result_1[0] + result_1[3]
        x2_br = result_2[0] + result_2[3]
        y1_br = result_1[1] + result_1[4]
        y2_br = result_2[1] + result_2[4]
        x_overlap =  max(0,  min(x1_br, x2_br) -  max(x_1, x_2))
        y_overlap =  max(0,  min(y1_br, y2_br) -  max(y_1, y_2))
        overlap_area = x_overlap * y_overlap
        area_1 = result_1[3] * result_2[4]
        area_2 = result_2[3] * result_2[4]
        total_area = area_1 + area_2 - overlap_area
        return overlap_area /  float(total_area)

def nms(detections, threshold=0.5):
        """
        非极大值抑制
        """
        if  len(detections) == 0:
                return []
        detections =  sorted(detections, key=lambda detections: detections[2],
        reverse=True)
        new_detections = [detections[0]]
        del detections[0]
        for index, detection in  enumerate(detections):
                for new_detection in new_detections:
                        if overlapping_area(detection, new_detection) > threshold:
                                del detections[index]
                                break
                else:
                        new_detections.append(detection)
                        del detections[index]
        return new_detections
```

1.5 动态步长滑窗

滑窗步长如果过大则容易错过部分待测目标。当时如果滑窗步长过小，则会导致每张图片的计算时间过长。本实验采用动态步长的方法。当检测到当前窗口车的置信度大于设置阈值时，减小步长；若小于阈值，则用较大步长进行滑窗。

1.6 评测指标

要求利用 Recall、Precision 和 F-measure 指标对模型结果进行评测。在介绍这三个指标前需要明确四个概念：TP、FP、TN、FN。

TP(True Positive)	在判定为 positive 的样本中，判断正确的数目
FP(False Positive)	在判定为 positive 的样本中，判断错误的数目
TN(True Negative)	在判定为 negative 的样本中，判断正确的数目
FN(False Negative)	在判定为 negative 的样本中，判断错误的数目

Recall、Precision 和 F-measure 相应的计算方法如下:

$$召回率(recall) = \frac{TP}{TP + FN}$$

$$精确率(precision) = \frac{TP}{TP + FP}$$

可见，精确率和召回率是相互影响的，理想情况下两者都高，但是一般情况下准确率高，召回率就低；召回率高，准确率就低；如果两者都低，证明算法有问题。

在两者都要求高的情况下，综合衡量精确率和召回率就用 F1 值。

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

2 实验结果及分析

2.1 分类器训练

2.1.1 数据集处理

本实验一共提供了 549 张训练集图片 (车) 和 499 张测试集图片 (非车)。如下为部分展示图:

图 5: 数据集展示

本实验利用 sklearn 中的 train_test_split 方法对数据集进行训练集和测试集的划分。对原始数据进行 1:4 的训练集测试集划分。划分结果如下所示:

```
数据划分结果，训练集： (838, 27104) 测试集： (210, 27104)
```

图 6: 数据集划分结果

2.1.2 PCA

划分好数据集后，利用训练集数据对 PCA 降维模型进行训练。本次实验选择将 27104 维数据降维至 300 维。如下是训练后转换结果。

```
pca结果，训练集： (838, 300) 测试集： (210, 300)
```

图 7: pca 降维后的训练集和测试集

2.1.3 分类器训练

PCA 降维模型训练好后，就用降维后的结果对分类器进行训练。最终分类器的训练测试结果为:

```
分类器训练结果:
              precision    recall  f1-score   support

      no car       1.00      0.97      0.99       106
         car       0.97      1.00      0.99       104

    accuracy                           0.99       210
```

图 8: 分类器测试结果

从图中可以看出，分类模型对于汽车分类的准确率为 0.97，召回率为 1，F1-score 为 0.99，总共参与测试的汽车图片样本数为 104 张。整体 210 张汽车和非汽车的分类准确率为 0.99。可以看到从当前的结果来看效果是非常理想的。

2.2 滑窗检测

降维模型和分类模型都训练好后，我们利用与训练图片等大的窗口大小对待检测图片进行滑窗。然后对每个窗进行分类，判断其是否为汽车。如果是汽车则将其保存为候选，最后利用所有候选带入 NMS 算法计算得到最终的检测框。如下为部分完全检测正确的结果示意图。

图 9: 检测正确展示

可以看到，模型不但能很好地分别出完整的汽车，也能分辨出被部分遮挡的汽车。分析原因是因为 HOG 算子和 PCA 降维方法让分类模型能够注意到汽车抽象的特征，从而达到遮挡分类正确的需求。

相对的也会有检测错误或者未被检测的情况。如下所示：

图 10: 检测错误展示

从错误的结果可以看到，真实世界的情况是千变万化的。第一张图中左侧车子未被检测，右侧成功检测，这里分析原因是因为左侧车子为白色，和背景颜色特别像，导致模型不能很好提取到车子的轮廓，导致分类错误。第二张图车子未被检测，分析原因是因为图中车子是斜着的，然后我们的滑窗是水平横滑，再加上滑窗使用的动态步长，导致车子那一块没有被减小步长，进一步导致车子检测失败。第三张图分析原因是因为车子在图中过小，由于我们用的是固定窗大小进行滑窗，导致没有检测到车子。第四张图可以看到模型将房檐当作车子了，这里分析原因可以看到被误分的房檐有一个较长的线条，模

型可能将该线条当作车的轮廓了，所以误分类了。

2.3 实时性分析

如下是采用动态步长滑窗、HOG 算子、300 维 PCA、Logistic 分类器的模型实时性分析结果。

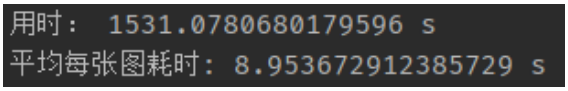

图 11: 耗时分析

从结果可以看到，处理 170 张图片总共耗时 1531 秒，平均每张图 8.95 秒，可见该算法实时性较差，分析原因是因为滑窗带来的大量冗余循环。

2.4 最终检测结果

由于测试集未给出标签。所以只能通过人为统计。经过统计发现该模型一共检测出了 183 辆车。结果如下：

指标	数量/(个)
TP(True Positive)	179
FP(False Positive)	4
FN(False Negative)	22

可以得到三个指标结果为：

$$recall = \frac{179}{179+22} = 0.890$$

$$precision = \frac{179}{179+4} = 0.978$$

$$F1 = \frac{2 \times 0.890 \times 0.978}{0.890 + 0.978} = 0.932$$

3 总结

经过这次实验作业，我更加深刻地理解了目标检测的相关知识。对相关算法进行了复现和实验。通过实验将上课所学的知识应用到了实践中去。通过实验加深了一些课堂上理论的理解。还让我发现了传统目标检测方法的一些问题所在。也开发了我的自主探索的意识。非常感谢朱老师给我这次实验的机会。