《计算机视觉》实验报告

姓名：严昕宇 学号：20121802

实验 6

一. 任务 1

a) 核心代码：

实验6 行人检测

实验目标

1. 在数据集INRIADATA上，使用Hog+SVM实现行人检测
2. SVM参数调优，提高分类准确率
3. 画出ROC曲线

实验步骤

1. 建立包含行人的一个图像数据库。这将作为我们的正数据样本
2. 建立不包含行人的一个图像数据库。这将作为我们的负数据样本
3. 在数据集上训练一个SVM
4. 将SVM应用于每个可能的测试图像块，以确定整个图像是否包含一个行人
5. SVM参数调整，优化模型，提高分类准确率

1. 数据集准备

本实验采用的INRIA 数据集是一组有标记的站立或行走的人的图像，是 Navneet Dalal在图像和视频中检测直立的行人的研究工作中收集的。该研究详见Dalal的论文"Histograms of Oriented Gradients for Human Detection"，也是在这篇论文中提出了 HOG+SVM行人检测算法。

INRIA 数据集中训练集有正样本614张（包含 1237个行人），负样本 1218张；测试集有正样本288张（包含589个行人），负样本453张。

[1]:
```
import os
import random
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

```
PosPath = './INRIADATA/normalized_images/train/pos'
NegPath = './INRIADATA/normalized_images/train/neg'
TestPosPath = './INRIADATA/normalized_images/test/pos'
TestNegPath = './INRIADATA/normalized_images/test/neg'

PosImgList = os.listdir(PosPath)
NegImgList = os.listdir(NegPath)
TestPosList = os.listdir(TestPosPath)
TestNegList = os.listdir(TestNegPath)
```

2. 设置Hog描述符参数

Hog中的Window、Block和Cell

1. Window
   在Hog特征提取时，一个窗口是最小的特征提取单元,类似于深度学习中的卷积.其运算公式一样。以示例图像为例输入图像(128, 64, 3),win大小为32x32,步长为(8,8),在输入两端各添加P个0,那么滑动次数为：列上滑动：(128-32)/8 = 12次,行上滑动 (64-32)/8 = 4，共滑动得到12x4 = 48 个窗口
2. Block
   假设block 设置为 8x8, 步长为(8,8), 此时一个窗口内,有 (32-8)/8 =3,行同样如此,共3x3 = 9个Block 这张图片共 48x9 = 432个Block
3. Cell
   这里的Cell,可以理解为不可再分的基本单元,Cell类似于将Block直接分块,若Cell大小为(4x4)则一个block 64/16 = 4个cell ,这张图共 48x9x4 = 1728个cell

[2]:
```
win_size = (48, 96)       # 检测窗口大小（检测对象的最小尺寸48 * 96）
block_size = (16, 16)     #（每个块最大为16 * 16）
block_stride = (8, 8)     # 单元格尺寸
cell_size = (8, 8)        #（从一个单元格移动8*8像素到另外一个单元格）
num_bins = 9              # 对于每一个单元格，统计9个方向的梯度直方图
# 设置Hog描述符参数
Hog = cv2.HOGDescriptor(win_size, block_size, block_stride, cell_size, num_bins)
```

3. 构建正样本

[3]:
```
x_index = []

for file in PosImgList:
    filename = f"./INRIADATA/normalized_images/train/pos/{file}"
    img = cv2.imread(filename)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # 减少计算量
    x_index.append(Hog.compute(img,(64,64))) # 利用HOG进行计算winStride

x_positive_hog = np.array(x_index,dtype=np.float32) # 数据类型转换，兼容OpenCV
y_positive = np.ones(x_positive_hog.shape[0],dtype=np.int32)

# (2416, 1980, 1) (2416,)，即2416个训练样本，1980特征值
print (x_positive_hog.shape, y_positive.shape)
```

```
(2416, 1980) (2416,)
```

4. 构建负样本 图像尺寸与正样本一样

负样本的图片已预先处理过，随机切割成128x64大小

[4]:
```
x_neg = []

for file in NegImgList:
    filename = f"./INRIADATA/normalized_images/train/neg/{file}"
    img = cv2.imread(filename)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # 减少计算量
    x_neg.append(Hog.compute(img,(64,64))) # 利用HOG进行计算winStride
x_negative = np.array(x_neg)
y_negative = np.zeros(x_negative.shape[0],dtype=np.int32)
# 计算非人的HOG
x_neg_hog = np.array(x_negative,dtype=np.float32) # 数据类型转换，兼容OpenCV

# (1218, 1980, 1) 即1218个训练样本，1980特征值
print(x_neg_hog.shape)
```

```
(1218, 1980)
```

5. 将X和Y（正样本和负样本）合并

[5]:
```
# 将X和Y(正样本和负样本）合并
feature = np.concatenate((x_positive_hog,x_neg_hog))
target = np.concatenate((y_positive,y_negative))
```

6. 数据集拆分

[6]:
```
from sklearn.model_selection import train_test_split
# 70%训练，30%测试
x_train,x_test,y_train,y_test = train_test_split(feature,target,test_size=0.3,
                                                 random_state=None)
```

7. 实现SVM支持向量机

[7]:
```
# 实现支持向量机
from sklearn.svm import SVC
clf = SVC(kernel = 'linear', C = 1, probability = True).fit(x_train,y_train)

# 计算训练集和测试集准上的确率
train_score = clf.score(x_train, y_train)
test_score = clf.score(x_test, y_test)
print("训练集上的准确率:", train_score)
print("测试集上的准确率:", test_score)
```

b) 实验结果截图

```
训练集上的准确率: 1.0
测试集上的准确率: 0.9523373052245646
```

图 1 Hog+SVM 行人检测初步结果

c) 实验小结

INRIA Person 图像数据集 Inria 数据集是最常使用的行人检测数据集。其中正样本（行人）为 png 格式，负样本为 jpg 格式。里面的图片分为只有车，只有人，有车有人，无车无人四个类别。需要注意的是，png 格式图片存在一定问题，需要搭配 PIL 库修复后才能正常读取。

二. 任务 2

a) 核心代码：

8. 网格搜索优化SVM

网格搜索适用于三四个（或者更少）的超参数（当超参数的数量增长时，网格搜索的计算复杂度会呈现指数增长，这时要换用随机搜索），用户列出一个较小的超参数值域，这些超参数值域的笛卡尔集（排列组合）为一组组超参数。
网格搜索算法使用每组超参数训练模型并挑选验证集误差最小的超参数组合。以SVM为例，挑选SVM的超参数C值、kernel类型和gamma值。
下面的配置表示要搜索两种网格：一种是linear kernel和不同C值；一种是RBF kernel以及不同的C和gamma值。Grid Search会挑选最适合的超参数值。

[8]:
```
from sklearn import model_selection as ms
# 要实验的超参数
params = [{'kernel': ['linear'], 'C': [1, 10, 100]},
          {'kernel': ['rbf'], 'C': [1, 10, 100], 'gamma': [1, 0.1, 0.01]}]
# 创建一个网格搜索: 9 组参数组合，5 折交叉验证
grid_search  = ms.GridSearchCV(SVC(probability=True),params,refit=True,
                               return_train_score=True, cv=2)
clf.fit(x_train, y_train)

# 开始搜索
grid_search.fit(x_train,y_train)
grid_search.score(x_train,y_train) # 输出精度
```

```
# grid_search.best_params_ 获得最优参数
# grid_search.best_score_ 保存的是交叉验证的平均精度，是在训练集上进行交叉验证得到的
print(grid_search.best_params_,grid_search.best_score_)  # 最优参数

# 获取最优模型
optimal_SVM = grid_search.best_estimator_

# 预测
predicted = optimal_SVM.predict(x_test)
print('预测值：', predicted[0:21])  # 输出结果是对应数据的标签
print('实际值:', y_test[0:21])

# 获取验证集的准确率
test_score = optimal_SVM.score(x_test, y_test)
print("测试集上的准确率：", test_score)
```

b) 实验结果截图

```
{'C': 10, 'gamma': 0.1, 'kernel': 'rbf'} 0.9638228701215801
预测值： [1 1 1 0 1 1 1 0 0 1 1 0 1 1 1 0 0 0 0 1 1]
实际值: [1 1 1 0 1 1 1 0 0 1 1 0 1 1 1 0 0 0 0 1 1]
测试集上的准确率： 0.9688359303391384
```

图 2 Hog+SVM 行人检测改进结果

c) 实验小结

sklearn 的 svm 模型主要有三种核：linear、rbf 和 poly，由于我们面对的是二分类问题，所以我们在这里仅分析 linear 和 rbf。对于 linear svm，主要参数为惩罚系数 C；对于 rbf svm，主要参数为 C 和 gamma。在这里我们使用网格搜索的方式进行机器调优。

三. 任务 3

a) 核心代码：

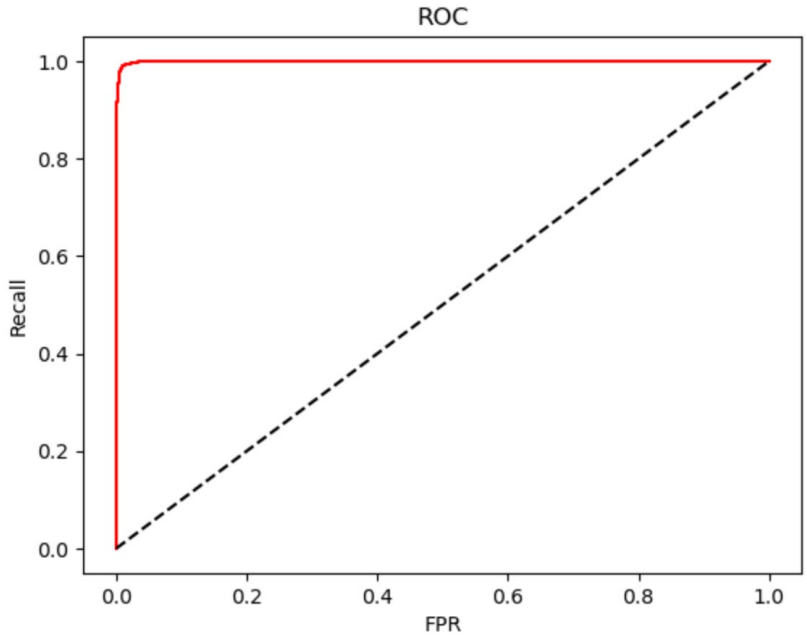
9. 绘制ROC曲线

[9]:
```python
# 导入要用的库
from sklearn.metrics import roc_curve

# 利用roc_curve函数获得假阳性率FPR，和真阳性率recall，都是一系列值
FPR, recall, thresholds = roc_curve(target,optimal_SVM.decision_function(feature),
                                    pos_label=1)
# 画图
plt.figure()
# 设置标题
plt.plot(FPR, recall, c='red', label='ROC curve')
# ROC 曲线
plt.title('ROC')
plt.plot([0, 1], [0, 1], color='black', linestyle='--')
plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('FPR')
plt.ylabel('Recall')
plt.show()

# 计算 AUC 面积
from sklearn.metrics import roc_auc_score
# 置信度，也可以是概率值
auc = roc_auc_score(y_test,optimal_SVM.decision_function(x_test))
print("AUC值:", auc)
```

b) 实验结果截图



AUC值：0.9936336412584854

图 3 ROC 曲线

图 4 随机样本预测

c) 实验小结

梯度直方图特征(HOG) 是一种对图像局部重叠区域的密集型描述符, 它通过计算局部区域的梯度方向直方图来构成特征。Hog 特征结合 SVM 分类器已经被广泛应用于图像识别中，尤其在行人检测中获得了极大的成功。需要提醒的是，HOG+SVM 进行行人检测的方法是法国研究人员 Dalal 在 2005 的 CVPR 上提出的，而如今虽然有很多行人检测算法不断提出，但基本都是以 HOG+SVM 的思路为主。