دانشگاه صنعتی خواجه نصیرالدین طوسی
K. N. TOOSI UNIVERSITY OF TECHNOLOGY

Pedestrian Detection using HoG Descriptors and SVM

Amirrad Kimiaei

MohammadReza Raei

Overview

In this project, you are going to implement a simple pedestrian detector using HOG feature descriptors and an SVM classifier.

Goals

1. Compute HoG features from images of a given data set, and extract image patches (both positive and negative examples).
2. Train an SVM classifier to perform classification and detection tasks.

Specifications

You need to train a pedestrian detector on the INRIA Person Dataset (http://pascal.inrialpes.fr/data/human). All details about the formats, positive examples' bounding boxes, etc. are described in the link. Just to clarify some obscurities:

1-The positive images are extracted from the original images and located (in two different sizes) in directories starting with the patch size (96X160H96, etc.).

2- According to the webpage, you should only use the 64x128 central part of these images. The reason why the pictures have been widened in both width and height is brought on the page.

3-Dividing the dataset to Train and Test are up to you. Although dataset providers seem to have fixed this, you are free to choose any portion of the dataset for training and the rest for testing.

4-Negative images are not provided in 64x128 patches. Read the documentation for details on how to build negative patches from negative images provided. **Building the negative data is all up to you**.

The next step is building a HoG Descriptor. The concept of HoG (Histogram of Oriented Gradients) was explained in the class. Look Here for a reminder.

OpenCV comes with a class called `HOGDescriptor`. This is the HoG feature extractor for your project. Find the official documentation here.

The class has few constructors; probably the most important one is the one taking the address of an xml file containing the HoG parameters. (The parameters can also be passed directly to the constructors but since the number of parameters are longer than you would expect, the first method is preferable.) Here is the format of the xml file where you can see each parameter as a tag:

```
<?xml version="1.0"?>
<opencv_storage>
<HOG type_id="opencv-object-detector-hog">
<winSize>35 35</winSize>
<blockSize>5 5</blockSize>
<blockStride>5 5</blockStride>
<cellSize>5 5</cellSize>
<nbins>9</nbins>
<derivAperture></derivAperture>
<winSigma>4.</winSigma>
<histogramNormType>L2</histogramNormType>
<L2HysThreshold>2.0000000000000001e-01</L2HysThreshold>
<gammaCorrection>1</gammaCorrection>
<nlevels>64</nlevels>
</HOG>
</opencv_storage>
```

If you know the details of the algorithm, you should be familiar with these parameters. Regardless, this article provides a good explanation of the parameters.

The next steps involve extracting features using the descriptor. The compute function

calculates the features. It only takes an image as input and returns the HOG feature vector.

Now we have to train an SVM classifier. OpenCV provides its own implementation of SVM. But since OpenCV's SVM is not properly documented, we will be using the SVC (support vector classifier) class in the scikit-learn library, a very popular machine learning package. Find the documentation here. We provide you with a sample code on how to build an SVM model, evaluate its accuracy, and feed the model's final parameters to the **HOGDescriptor** as a classifier:

```
from sklearn.svm import *
import itertools
#builds the model based on parameters provided
svm=SVC(kernel='sigmoid',gamma='auto',C=0.01,max_iter=-1,tol=1e-4,coef0=1)
#training the model
svm.fit(image_features, labels)
#printing the score of our model
print svm.score(Test_data, Test_Label)
#the remaining section of the code puts the final support vector params in
correct order for feeding to our HOGDescriptor
supportvectors.append(np.dot(svm.dual_coef_,svm.support_vectors_)[0])
supportvectors.append([svm.intercept_])
supportvectors = list(itertools.chain(*supportvectors))
#passing the model params to HOG
HOG.setSVMDetector(np.array(supportvectors, dtype=np.float64))
```

One crucial fact about OpenCV's SVM detector is that it takes the SVM parameters in the following order: the coefficients of support vectors machine followed by the intercept (constant) of the model. To test the model, the next line of code returns the bounding box for the found objects:

```
rects, weights = HOG.detectMultiScale(img, winStride=(5,5),scale=1.1,
padding=(0,0),finalThreshold=0,useMeanshiftGrouping=0)
```

The parameters for this function are explained here.

You are now set to do the project. Good Luck!