Lab Instructions - session 13

Image Classification

Mohammad Akbarizadeh

NOTE: You will need to have the **scikit-learn** and **scikit-image** libraries installed:

```
pip install scikit-learn
```

```
pip install scikit-image
```

Read and Display data

In the folder “**dataset**”, you can find 32 by 32 images of Persian handwritten digits. Browse through the subfolders and look at the images.

Task 1

Write a program that reads the images from the dataset and randomly displays four examples of each Persian digit.

Two class classification using SVM

In the following code, we use a Support Vector Machine (SVM) classifier to classify images into two classes. We simply vectorize each image and feed it as features to the SVM. To save your precious time, the classifier is saved to a file from which you can later load your model. You can see the accuracy of the model presented as output.

File: **svm_classifier.py**

```
import numpy as np
import cv2
from sklearn.svm import SVC
from sklearn.externals import joblib
import os
import random

dataset = './dataset/train/{}/'

train_images_list1 = os.listdir(dataset.format('1'))
train_images_list0 = os.listdir(dataset.format('0'))
```

```
train_images_list1.sort()
train_images_list0.sort()

train_labels = [1 for i in range(len(train_images_list1))]
train_labels.extend([0 for i in range(len(train_images_list0))])

input_data = []

for addr in np.ravel([train_images_list0, train_images_list1]):
    I = cv2.imread(os.path.join('dataset/train/{}/'.format(str(addr[2])),
addr))
    input_data.append(I.ravel())

classifier = SVC(gamma='auto')
file_name = 'saved_svm.sav'

if not os.path.isfile(file_name):
    classifier.fit(input_data, train_labels)
    joblib.dump(classifier, file_name)

else:
    classifier = joblib.load(file_name)

idx = [random.randint(0, 120) for i in range(10)]
test_input = [input_data[i] for i in idx]
test_labels = [train_labels[i] for i in idx]
results = classifier.predict(test_input)
print('predictions: ', results)
```

- Print the train labels. How are related to the images?

Displaying the accuracy

In order to examine the accuracy of our classifier, we must test it using some test data and see if our classifiers answers match the labels we wanted or not. As you can in previous code some test labels are available. You can print the accuracy by dividing the correct predictions by the total number of predictions.
Add the following piece of code to the previous code to print the accuracy.

```
print('Accuracy: ', (np.sum(results == test_labels) / len(results)) * 100,
'%')
```

Task 2

As you can understand from the previous code, the accuracy is calculated on the train data. This is not a good way to evaluate a classifier, because it can **overfit** the training data (work very well on the training data), but perform poorly when tested on the new (unseen) data. Your task is to examine the accuracy of the classifier both on the **train data** and the **test data**.

Task 3

So far, we have done a two-class classification. Your task is to perform classification on all 10 digits and display the train and test accuracies just like task 2. Your classifier must take all the ten digits as input and classify them.

The HoG feature

The following code extracts the HoG features from an image and displays them. Try it on different images and see the results.
File: **displaying_HoG_features.py**

```
from skimage import exposure
from skimage import feature
import cv2
import numpy as np

logo = cv2.imread('01.jpg')

(H, hogImage) = feature.hog(logo, orientations=9, pixels_per_cell=(8, 8),
                            cells_per_block=(2, 2), transform_sqrt=True,
block_norm="L1",
                            visualize=True)
hogImage = exposure.rescale_intensity(hogImage, out_range=(0, 255))
hogImage = hogImage.astype("uint8")

cv2.imshow("HOG Image", hogImage)
cv2.waitKey(0)
```

- How do you interpret the displayed HoG features? How are they related to the image gradients?

Task 4

Write a program to do the classification but instead of giving the vectorized image as input, give the HoG feature as input to the SVM and display the accuracy of your model.

LBP features

Local Binary Patterns (LBP) is a fast and effective visual features used for image classification. The following code extracts the LBP features and displays them.

File: **displaying_LBP_features.py**

```
import numpy as np
import cv2
from skimage import feature

fname = '01.jpg'

numPoints = 28
radius = 3

image = cv2.imread(fname, cv2.IMREAD_GRAYSCALE)
lbp = feature.local_binary_pattern(image, numPoints, radius)
J = np.copy(lbp)
J = np.array(J, dtype=np.float32)
cv2.imshow('J', J)
cv2.waitKey(0)
```

Task 5

Repeat Task 4 using the LBP features.

Task 6

Repeat task 5 using both HoG and LBP features.

References and further reading

Local Binary Patterns
different versions of LBP
HoG-LBP human detector (paper)
scikit-learn SVM
Histogram of Oriented Gradients