CS696, Applied Computer Vision

Homework Assignment 5

Instructor: Xiaobai Liu

**How to submit**

* Submit your source codes and writeup through SDSU Blackboard, before 11:59pm, April 23, 2017
* Three submissions are allowed.

**Overview**

Object classification is an important task in many computer vision applications, including surveillance, automotive safety, and image retrieval. For example, in an automotive safety application, you may need to classify nearby objects as pedestrians or vehicles. Regardless of the type of object being classified, the basic procedure for creating an object classifier is:

* Acquire a labeled data set with images of the desired object.
* Partition the data set into a training set and a test set.
* Train the classifier using features extracted from the training set.
* Test the classifier using features extracted from the test set.

In this mini-project, you will be instructed to program flower image classifiers using various features, e.g., histogram, template, or others, and a multiclass SVM (Support Vector Machine) classifier. This type of classification is often used in many image applications.

The example uses the |fitcecoc| function from the Statistics and Machine Learning Toolbox(TM). You should feel free to use any SVM function available.

**Datasets**



We use the classical flower category dataset, which contains images of flowers belonging to 3 different categories (as shown in the above). The images were acquired by searching the web and taking pictures. There are 80 images for each category. Note that the original dataset includes 17 flower categories and we use a subset.

In the folder ‘data’, we use three subfolders to store the three types of flowers, respectively. Each subfolder corresponding to a flower type includes 80 images.

**Starter Codes**

The script *main\_SVM.m* provides an pipeline for this project. You will need to complete it. In particular, please replace the coding lines of PLACEHOLDER as yours.

**Programing Instructions**

There are Four major steps. Detailed instructions can be found in the script file.

**Step-1**: Load training and test data using imageSet (You don’t need to make any changes)

This step is to load images from the data folder and visualize an exemplar flower for each flower type.

**Step -2:** Partition the data set into a training set and a test set (You don’t need to make any changes)

In this example, for each flower type, we randomly select 50 images for % training; and use the other 30 images for testing.

**Step-3:** Train the classifier using features extracted from the training set.

Image classification is a multiclass classification problem, where you have to classify a flower image into one out of the three possible flower classes.

In this example, the |fitcecoc| function from the Statistics and Machine Learning Toolbox(TM) is used to create a multiclass classifier using binary SVMs.

Start by features from the training set. These features will be used to train the classifier.

This step includes three components.

**Step 3.1** Extract features from training images. Loop over the training images and extract features from each image. A similar procedure will be used to extract features from the testing set

You will need to write your own codes to extract various features from the input images. Please try at least **three types of features**, e.g., histogram of color, histogram of gradient, SIFT-like features, and compare their performance.

**Step 3.2**, train a classifier using the extracted features. We use fitcecoc uses SVM learners and a 'One-vs-One' encoding scheme.

**Step 4**: Evaluate the Flower Classifier

Evaluate the flower classifier using images from the test set, and generate a confusion matrix to quantify the classifier accuracy.

As in the training step, first extract features from the test images. These features will be used to make predictions using the trained classifier. The procedure is similar to what was shown earlier.

**Step 4.1**: Loop over the testing images and extract features from each image.

**Step 4.2**: Make class predictions using the test features.

**Step 4.3**: Tabulate the results using a confusion matrix.

We use the function confusionmat() to calculate the confusion matrix. The columns of the matrix represent the predicted labels, while the rows represent the known labels.

**Step 4.4**: calculate accuracy, i.e. the number of images with correct predictions over the total number of images.

**Summary**

This example illustrated the basic procedure for creating a multiclass image classifier using the feature descriptors and the |fitcecoc| function from the Statistics and Machine Learning Toolbox(TM). Other feature descriptors and machine learning algorithms can be used in the same way. For instance, you can explore using different feature types for training the classifier; or you can see the effect of using other machine learning algorithms such as k-nearest neighbors.

**You will need to write your codes for Step 3.1, and Step 4.1.**

**Write-up**

In the report you will describe your algorithm and any decisions you made to write your algorithm a particular way. Then you will show and discuss the results of your algorithm.

In the case of this project, show how well your classifiers work using different types of features. You can compare two classifiers in both qualitative and quantitative ways. For example, you can show a few testing images for which a model can’t work while the other model work. Also, you can compare accuracies of two classifiers side by side which directly tells the differences.

A good writeup will assess how important various design decisions were. E.g. by using SIFT-like features instead of histogram of color I went from 0.6 accuracy to 0.65 accuracy.