67777 Visual Object Recognition
Exercise

Description:

In this project you will be asked to test 2 different object recognition approaches based on the same Bag of Words image representation:

1. The first approach is supervised, using image labels during the training phase, and discriminative. It trains a discriminative classifier (using SVM) to distinguish between different object classes.
2. The second approach is unsupervised and generative. It uses all the images in the training set to learn a set of topics using the formalism of pLSA, see: Sivic, J. and Russell, B. and Efros, A. and Zisserman, A. and Freeman, W., "Discovering object categories in image collections."

The Bag of Words representation is created as follows:

- Keypoints are randomly sampled at different scales from the image (giving higher probability to points around the center).
- The SIFT descriptor is used to describe the region around each point; the size of the region is defined by the keypoint's scale.
- A codebook is learnt by clustering all descriptors from all images in the train set using k-means.
- Each image is represented by a histogram over this codebook.

All the code needed for creating the representation and training both models is given to you. Most of the code was taken from Li Fei-Fei, Rob Fergus & Antonio Torralba's ICCV 2005, Recognizing and Learning Object Categories short course.

The test case is to distinguish between Facial images and images from some Clutter Background set. The images were taken from the Caltech-101 benchmark dataset, with 100 images from each type.

How to run:

- Download VORProj.tgz from the course web site to a new directory (e.g VORProj/)

- Untar using 'tar -xvf VORProj.tgz', creating 3 folders:
  'common/' - holds all the code used in the experiments
  'experiments/' - holds files controlling the parameters of the experiments and several subfolders, where all data created while running the experiments is saved
  'images/' - holds the images used for the experiments.

- In order to run the code put the **common/** directory in your Matlab path and move to the **experiments/bag_of_words/** directory.

- In file Params.m - change ‘DirPath’ so it will point to the root of the project (e.g. VORProj/).

- All of the following functions should receive as input the name of the parameters file, ‘Params’.

do_random_indices: randomly picks the training and test indices and stores them in a file in the current experiment directory.

do_preprocessing: copies and rescales (as specified) all images from the IMAGES_DIR into the images/ directory in the current experiment directory.

do_interest_op: Samples points randomly from the center of the image. Points are sampled from a Gaussian distribution with the center of the image set as the mean. This relies on the output of do_preprocessing.

do_representation: takes the regions produced by do_interest_op.m and characterizes their appearance, hereusing the SIFT descriptor. This is done by means of a linux binary. do_form_codebook.m, whichRrns k-means on the SIFT vectors from all training images to build a visual vocabulary.

do_vq: Vector quantizes the descriptors from all images using the vocabulary built by do_form_codebook.m.

do_plsa: Learns a pLSA model from the training images.

[Pd_z_test,labels] = do_plsa_test: Tests a pLSA model on the testing images. Outputs Pd_z_test - an MxZ matrix where M is the number of images in the test and Z the number of topics which the pLSA was trained with. Cell (m,z) is the probability of topic z in image m. labels is a vector of length M holding the ground-truth label for each image.

[Margins,Tags] = RunSVM: Trains and tests an SVM classifier given the histogram of dictionary/codebook representation of the images obtained after running do_vq.m. Margins is a vector of length M (number of test images) which holds the distance of the classified test image from the separator learnt by the SVM. Tags is a vector of length M holding the ground-truth label for each image in the test.
(This function receives no input and should be run from /experiment/bag_of_words/)

You should be able to run all code on the school’s **linux** computers

Tasks:

In this project you are first asked to use the given code to train and test the pLSA and SVM model. The you should analyze the recognition performance by computing the

Receiver operating characteristic (ROC) curve (see explanation in: http://en.wikipedia.org/wiki/Receiver_operating_characteristic) and its corresponding Equal Error Rate (Sensitivity = Specificity).

Specifically:

1. Run the code with the default settings.

2. Compute a single score for the pLSA . Note that the direct output of the pLSA model provides probabilities over topics given an image. In our case this means two numbers for each image. Think of a sensible heuristic for combining both into a single score, which measures preference for one of the two categories.

3. Compute ROC curves for the default settings. Two curves should be plotted: one given the margins obtained for the test images using the SVM classifier, and one given the score you have computed for the pLSA classifier.

4. Check the effect of the different parameters on the overall performance. Try different values for the following variables found in Params.m:

- IPNum: Number of extracted interest points.
- IPScaleRange: Scale of extracted regions.
- CBSize: Size of code book (number of words).
- CWRSampleVar: Variance of the Gaussian distribution used to sample points around the center of the image.
- TestPortion: Test size.

You should submit 4 ROC plots, each showing 2 curves and the EER (Equal Error Rate) of each curve (one for SVM and one for pLSA). The first plot should correspond to the default settings. The remaining 3 plots should correspond to 3 different configurations that you found interesting.

Discuss the effect of the different parameters you played with.

Submission date: 22.06.09

Bonus:

There are 3 possible bonus “tracks” (bonus points will be added to the final course grade, up to the maximal number in parenthesis, and no more than 10 points):

1. Improve classification results of the default settings by swapping the random sampling of interest points with any interest point (IP) detector of your choice. Submit a comparison of the results, together with your new do_interest_op.m and all auxiliary code needed for it to run your chosen detector. Make sure that the code can be run,

and no change in any of the other functions is needed. Implementations of IP detector can be found in: http://www.robots.ox.ac.uk/~vgg/research/affine/. (**4 points**)

2. Use a non Bag of Words approach (e.g., a representation based on a Histogram of Oriented Gradients (HOG) or a constellation model), and compare the results. Submit all the code with a detailed explanation on how to run it. (**7 points**)

3. Think about different test cases from the caltech101 dataset: show examples of classes that work and some that don't work, and try to discuss the reasons for this. (**3 points**)