Laboratory 2: Bag-of-Words exercises

Instructors: Lamberto Ballan <lamberto.ballan@unifi.it> and Lorenzo Seidenari <lorenzo.seidenari@unifi.it>
(materials at http://www.micc.unifi.it/ballan/?p=1156)

Open and edit the script **exercises.m** in Matlab editor. The script contains commented code and description for all steps of this exercise. You will need to fill in some parts of this script with your own code. The steps of the exercise requiring your code modification are marked in red below.
You need to complete **all the mandatory** exercises (i.e. Exercise 1, 2, 3 and 4) and solve **at least one optional task** (Exercise 5).
You are free to propose your own optional task to solve but in that case you need tutor approval.

Experimentation on *Caltech-4* and *Caltech-15* datasets is suggested (specially on part 3, 4 and on the optional tasks).

You can find a good introduction to Matlab at www.math.toronto.edu/mpugh/primer.pdf

Exercise 1: Feature quantization

- Follow the steps in **exercises.m** and load pre-computed features for all training and test images. Visualize features for some of the training images. Next, compute visual dictionary by clustering a subset of feature descriptors from the training images.

- **1.1** Write your code computing L2 distances between SIFT descriptors in all your images and cluster means of visual dictionary.
- **1.2** Assign each feature in all of the images a unique visual word label by minimizing its distance to cluster centers

- Show image patches with common visual word IDs.

Exercise 2: Bag-of-Features image representation

- **2.1** Represent each image with L1-normalized histogram of visual word labels: Bag-of-Features (BoF).

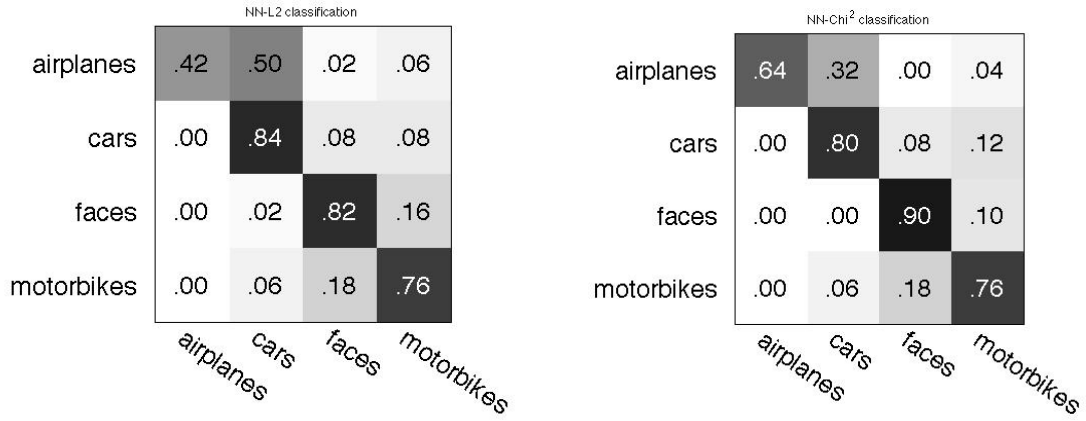
Exercise 3: Image classification - NN classifier

- Follow the steps of Nearest Neighbor image classification: compute L2 distances between BoF of all test and training samples. For each test image find its nearest training image and assign its label to the considered test image. Compute overall and per-class classification accuracies.

- **3.1** Repeat classification steps above with Chi2 distance instead of L2 distance. Compare classification accuracies for both distances.

Expected results:

You should obtain an overall NN classification accuracy of ~**71%** with L2 distance on 4_ObjectCategories dataset. You should obtain better results with NN-Chi^2 classification, i.e. ~**77.5%**.
As an example, see the following confusion matrices for NN-L2 and NN-Chi^2 image classification.



Exercise 4: Image classification - SVM classifier

- Show correctly classified and miss-classified images for all four classes. Reason about the success of the classifier and the possible sources of errors.

- You have until now employed NN classification and linear SVM. Support Vector Machines provide a better performance and allow to consider non-linear feature remapping, usually improving the performance.
- Use LIBSVM to train a linear model. Pay attention to the cross-validation procedure used to select the best C parameter for the trained model.
- **4.1** Now use the precomputed kernel interface to pass in a kernel matrix of scalar products (this should produce the same results).
- **4.2** Train the SVM with a precomputed non-linear histogram intersection kernel [Lazebnik06] and select the best C parameter for the trained model using cross-validation.

Exercise 5: Analysis of results

- **5.1** How does the training set size affects the classification performance?
- **5.2** How does the system works with different features (i.e. "sparse" vs "dense" SIFT features)?
- **5.2** What are the other parameters of the pipeline that can improve/degrade the performance?

- **5.3** How does the system behave varying those parameters? Can you give a brief explanation of the system behavior at the variation of each of the parameters you have tested?

Optional exercises:

Exercise 6.1: Feature quantization

- Instead of assigning each SIFT to a visual word devise a scheme in order to "soft weight" and represent each descriptor with multiple visual words [vanGemert08].

Exercise 6.2: Bag-of-Features image representation

- Bag-of-words representation lacks spatial structure. Data structures of SIFT descriptors contain x,y and scale. Devise a scheme in order to preserve a certain amount of this information (spatial pyramid kernel [Lazebnik06])

Exercise 6.3: Feature fusion

- Repeat exercise 1, 2, 3 and 4 with color descriptors to obtain bag of words representations for each image and kernels.
- Experiment with two feature combination schemes: early (bag of words histogram concatenation) and late fusion (kernel combination) [Gehler09].
- Repeat the exercise using also another different feature: GIST. It is a global feature and so you have to change the pipeline. Extract GIST features and directly pass them to the classifier. Compare the results obtained with this feature with the previous ones and also experiment a late fusion scheme with kernels obtained using dense-SIFT and color descriptors.

Final Report

The final report must not exceed 6 pages (using the course report-template) and must follow the following structure.
- For each section report briefly how you solved the mandatory and optional (if related) exercises.
- Related optional exercises are reported in square brackets for each section.

1. Introduction: which introduces the report with an overview of the work done, a brief description of the optional exercises assigned, and insights gained through the exercise.
2. Feature extraction: which describes the features employed and their properties. [**6.3**]
3. Image representation: that addresses the representation of the image with the bag of features paradigm (vocabulary creation, quantization, bag-of-words representation). [**6.1**]
4. Image classification: which briefly describes the classifiers used and analyze their performance. [**6.2**,**6.3**]
5. Conclusion: a concluding section containing an analysis of the whole system implemented, in particular with highlights of the optional exercises assigned.

VERY IMPORTANT: We encourage collaboration and learning from examples available on the internet. But **all** source code submitted must be the sole work of the group submitting it. Studying the implementation of someone else is always a good idea, but you must synthesize the ideas into *your own implementation.* If you *must* include code from someone else, make sure you include the correct attribution of ownership in the source code and in your report. Code copied and pasted from the internet or from other students without attribution is plagiarism and **your grade will suffer**.