Exercise 6: Recognition with Bag of Visual Words

In this exercise, we will first have a look at the k-means algorithm. You will be required to implement the algorithm by yourself. Then you will learn about how to do image classification and retrieval using Matlab's built-in functions.

1 k-means clustering

k-means clustering is a method to partition the data space into k exclusive clusters. It is an iterative process which starts with k random seeds. The algorithm aims to minimize:

$$D(X, M) = \sum_{cluster\, k} \sum_{point\, i} (\mathbf{x}_i - \mathbf{m}_k)^2$$

One possible implementation have the following steps:

1. randomly select k instances from the data
2. define k clusters by taking the selected instances as cluster centers
3. repeat the following steps until data point associations do not change anymore
   - calculate distances for all data points from k cluster centers
   - associate each data point to the closest cluster
   - update the cluster centers with the mean value of the points in each cluster

The first exercise is to implement the function `kmeans_cluster.m`. Please read carefully and follow the description at the beginning of the function. Once you have finished, run the script `test_kmeans.m` to test your implementation on several 2D datasets and observe the results. Also compare your implementation with the built-in `kmeans` function.

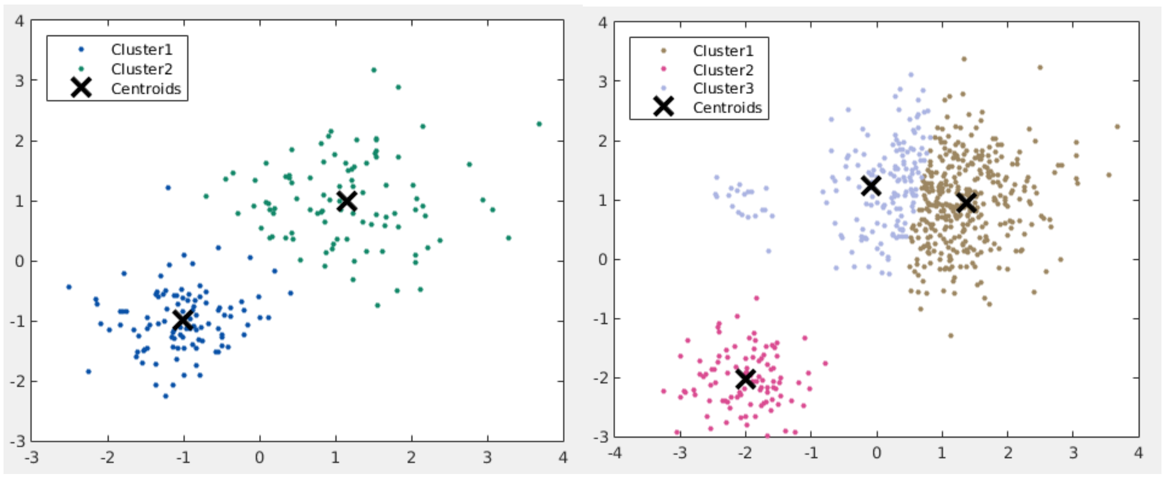

Figure 1: sample cluster result

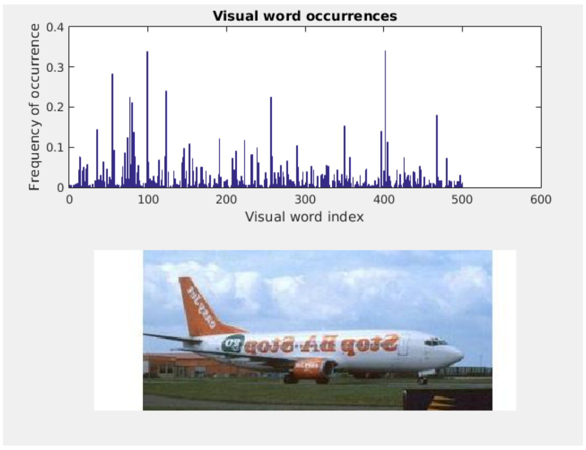

Figure 2: visual words occurence frequency in a sample image (use the `encode` function)

k-means is a simple algorithm and the only parameter is the cluster number. However, as you have probably observed from the exercise, it may fail in several cases[1] (e.g., unbalanced cluster, as in the right image of Fig. 1). Also the built-in k-means function in Matlab uses a variant called k-means++, which uses a better initialization scheme.

2 Bag of Visual Words

Both image classification and retrieval are based on the representation of Bag of Words. To construct a vocabulary (of visual words), the following steps are needed:

1. detect features
2. calculate the descriptors for the feature detected
3. cluster the features (e.g., using k-means) in the descriptor space into individual visual words

It is easy to use Matlab's built-in function `bagOfFeatures` to get a visual vocabulary:

```
bag = bagOfFeatures(image_set)
```

By default, the `bagOfFeatures` function uses the SURF feature to build the vocabulary. It is also possible to use user-defined descriptors (e.g. color information, read this Matlab tutorial for more details).

Matlab uses `imageSet` class to manage different categories/folders of images. The code for parsing the dataset and generating the vocabulary is already given (in the script `image_classification.m` and `image_retrieval.m`). Read and understand this part, since you may want to test on your own dataset later.

3 Image Classification

We use images from the Caltech101 dataset for image classification. The images are sorted into different categories according to the objects in the images. In this exercise, we use three categories of objects to train and test the classifier.

The workflow of the image classification is as follows:

1. Separate the images into two part: training set and validation set (using the function `partition`)
2. Training:

[1]read this page for more examples

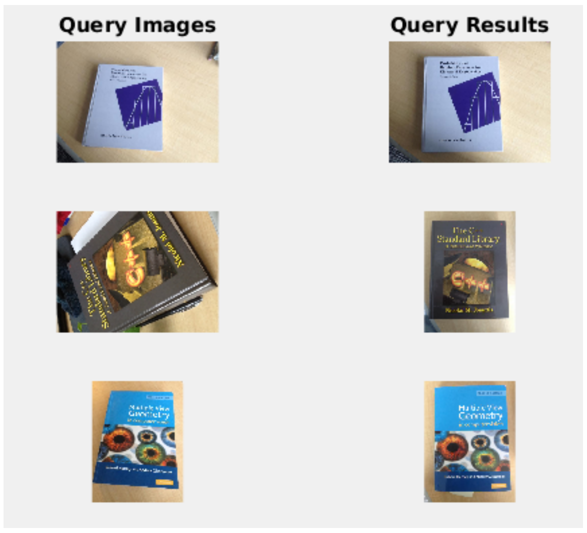

Figure 3: Retrieval result

- create vocabulary from the training set (using the function `bagOfFeatures`)
- train the classifier using the training set (using the function `trainImageCategoryClassifier`)

3. Testing:

   - validate the classifier using the images from the validation set

For the testing phase, you can choose to evaluate the whole validation set (using the function `evaluate`) or only for one image (using the function `predict`).

The second exercise is to complete the script `image_classification.m` and observe the classification result (refer to the imageCategoryClassifier class for API usages). Note you can press `Ctrl+Enter` to evaluate the last section only so that you can test your classfier on multiple (random) images.

4 Image Retrieval

We use the `bookCovers` dataset that comes with Matlab to practice the image retrieval. The dataset contains images of different books covers and under the `queries` subfolder there are three images of book covers that will be used as query images.

The work flow is:

1. index the images:
   - create visual vocablary
   - create inverted image index to map from visual words to images
2. use an (unindexed) image to query the index and find the most similar image.

The matlab function `indexImages` accepts an image set and do the index part (both the vocabulary buiding and the indexing). Then use the `retrieveImages` function to query the index. The third exercise is to use these two functions to finish the script `image_retrieval.m` (refer to this page for the workflow and API usages). The result should be identical to Fig. 3