Assignment 4 – CV1 WS14/15 12/01/2015

Prof. Stefan Roth
Jochen Gast
Stephan Richter

This assignment is due on January 26th, 2015 at 13:00.

Please refer to the previous assignments for general instructions and follow the handin process from there.

Problem 1 - PCA for Face Images (15 points)

You will be working with a training database of human face images and build a low-dimensional model of the face appearance using Principal Component Analysis (PCA). The outline is given in `problem1.m` and should be completed with the necessary function calls:

- In the data folder you will find a directory `yale_faces` containing 20 face images for 38 different persons, respectively. Implement the function `load_faces.m` that loads all face images into a big data matrix.
- Implement the PCA of all face images in `compute_pca.m`. Each face image has over 8000 pixels and there are many fewer training images than this. Consequently, you want to use SVD as we discussed in class; in particular you want to use the "economy mode" (`[U,S,V] = SVD(X,0)` in Matlab). The output arguments of the PCA function contain the PCA basis vectors, their Eigenvalues as well as the the data mean. Make sure that the principal components are properly sorted in decreasing order. Also compute a vector that contains the cumulative variance of the principal components.
- Show a plot of the cumulative variance of the principal components. Briefly explain in `answers_problem1.txt` how you compute the variance of a single principal component and why the variance can be obtained in that way.
- Implement `compute_ncomponents.m` that, based on the cumulative variance, computes the number of bases required to account for 80% and 95% of the variance, respectively. Print these numbers on the command window.
- Display the mean face and the first ten Eigenfaces in `show_faces.m`.
- Choose a random face (i.e. random index) out of your data matrix and fetch its image by implementing `take_face.m`. Project the chosen image onto a low-dimensional subspace using 5/15/50/150 components, respectively, and compute reconstructions of the original image using these low-dimensional projections in `compute_reconstruction.m`. Finally, display the 4 different reconstructions as well as the original face in a single figure.

Problem 2 - Bag-of-Words Model and Categorization (25 points)

In this problem you will implement a simplified Bag-of-Words model as well as a naive Bayes classifier for categorizing airplanes and motorbikes. The outline is given in `problem2.m` and should be completed with the necessary function calls.

Creating the codeword dictionary:

First of all, you should build a codewords dictionary – representative keypoints, over which each image could be represented by a distribution. We represent these keypoints using the k-means cluster centers of all *feature vectors* from the training images. k-means clustering aims to partition n observations $(\mathbf{x}_1, \cdots, \mathbf{x}_n)$ into k ($k \leq n$) clusters $(s_1, \cdots, s_k)$ so as to minimize the within-cluster sum of squares

$$\sum_{i=1}^{k} \sum_{\mathbf{x}_j \in s_i} \|\mathbf{x}_j - \mu_i\|^2, \tag{1}$$

where μ_i is the mean of points in s_i. To find the means, you can start with some (random) initial k means $\mu_1^{(1)}, \cdots, \mu_k^{(1)}$, and then alternate between two steps:

- assign each observation to the cluster with the closest mean

$$s_i^{(t)} = \{\mathbf{x}_j : \|\mathbf{x}_j - \mu_i^{(t)}\| \leq \|\mathbf{x}_j - \mu_{i^*}^{(t)}\| \text{ for all } i^* = 1, \cdots, k\},$$

- and calculate the new means to be the centroid of the observations in the cluster

$$\mu_i^{(t+1)} = \frac{1}{|s_i^{(t)}|} \sum_{\mathbf{x}_j \in s_i^{(t)}} \mathbf{x}_j$$

until the assignments no longer change.

Tasks:

- In the data directory you will find image folders for both airplanes and motorbikes. Please implement the function `load_data.m` that creates both a training and a testing structure containing the images, class labels and number of samples per set. You should randomly separate the whole set of images (i.e. including airplanes and motorbikes) into two equal-size parts for training and testing, respectively. Please denote airplanes with class label $C = 0$ and motorbikes with class label $C = 1$.
- Apply the Harris detector and SIFT descriptor to obtain feature vectors for all images in both training and testing set. Use the parameters $\sigma = 1.4$, filter size 15×15 and threshold 10^{-7} for interest point detection and ignore points within a 10-pixel wide boundary. Please implement the feature computation for a single image in `im2feat.m` and the feature computation for the whole set in `extract_features.m`.
- Different images may have a varying number of interest points and thus a different number of feature vectors. Concatenate the features of all images in the training set by implementing `concatenate_features.m`. The resulting feature matrix will be used to compute the codebook of keypoints.
- Implement `compute_codebook.m` that computes the codeword dictionary using k-means clustering with $k = 50$. If any of the cluster centers has no data points associated with it after an iteration, replace it with a random point of the given data.

Categorize images with naive Bayes classifier

To categorize an image I_m with the naive Bayes classifier, first label each extracted feature descriptor from I_m with the keypoint *(i.e.*, cluster center) to which it lies closest in feature space, and count the number $n(i,m)$ of times of keypoint μ_i occurs in I_m; then apply Bayes rule and take the category C_j with the largest posterior as the prediction

$$p(C_j|I_m) \propto p(C_j) \cdot p(I_m|C_j) \propto p(C_j) \prod_{i=1}^{k} p(\mu_i|C_j)^{n(i,m)}. \tag{2}$$

Here $p(C_j)$ are the category priors. The class-conditional probabilities of keypoint μ_i given category C_j are estimated from the labeled training images. In order to avoid probabilities of zero, these estimates are computed with Laplace smoothing:

$$p(\mu_i|C_j) = \frac{1 + \sum_{I_m \in C_j} n(i,m)}{k + \sum_{t=1}^{k} \sum_{I_m \in C_j} n(t,m)}. \tag{3}$$

Tasks:

- For all images you need to compute the histogram of keywords over the codebook. Implement the function `compute_histogram.m` that takes a set of images as well as a codebook and computes the number of occurrences of keypoints for each image. Use the (squared) Euclidean distance to assign feature vectors to the closest keypoint in the codebook, respectively.
- Train a naive Bayes classifier for separating both classes in `train_bayes.m`. To do that you should estimate the class priors $p(C_j)$ as well as the class conditional probabilities according to the training data and the codebook.
- Reformulate (2) in log space, i.e. compute the log posterior probabilities in `compute_posterior.m`. Why is it better to compute and compare log probabilities rather than using posterior probabilities directly? Please answer this question in `answers_problem2.m`.
- Based on the log probabilities compute the final predictions for both training and testing set in `test_bayes.m`. Also compute the error rates for both sets and the error rates on the command window.