COMP6223 - Computer Vision
Coursework 3

Drago Matteo (29824648) & Azzino Tommy (29824613)

Introduction

All the code for this coursework has been written in MATLAB.
In all our algorithms we used `imageDatastore` in order to speed up images retrieval and manipulation. In particular, we decided to use 25% of the training set for validation: in this way we tested our different algorithms multiple times and in different configuration before categorizing the test set. The split of the training set, where possible, has been done with `splitEachLabel`.

Run 1

As first approach we had to develop a classifier which use as features *tiny images*. This particular representation consists on few simple steps: initially we resize the image to a 16x16 matrix, then we build a vector of 256 pixels composed of the consecutive rows of the new image.
After that, in order to improve classification performances, we implemented two different types of normalization to the vector, which in the following we refer to as $\mathbf{x}$:

- **standard normalization**: $\bar{\mathbf{x}} = \frac{\mathbf{x}-\mu}{\sigma}$ where μ is the mean of $\mathbf{x}$ and σ is its standard deviation
- **unit length normalization**: $\hat{\mathbf{x}} = \frac{\bar{\mathbf{x}}}{\|\bar{\mathbf{x}}\|}$

In this way we obtained a set of unit length vectors of zero mean.
The idea behind the classifier is that one vector representing an image of a specific class will likely be similar to other vectors of the same category. So, once we perform this operation for all the training set, we can determine the category of each image of the validation set using the **k-nearest-neighbour** classifier: this means that we evaluate the distance between the tiny image to validate and all the vectors from the training set; then, we pick the k nearest vectors (of which we know the class) and we classify our image with the most represented class in the neighbourhood. In case we have two classes most represented we can implement different choice polices, in our case we used the first classes returned by MATLAB.
In order to find better performances, we tried to tune the value of k for the several measures of distance available in `vl_alldist2`, in particular:

$$\mathbf{L_{INF}} = max|X-Y| \qquad \mathbf{L_2} = sum(X-Y)^2 \qquad \mathbf{L_1} = sum|X-Y| \tag{1}$$

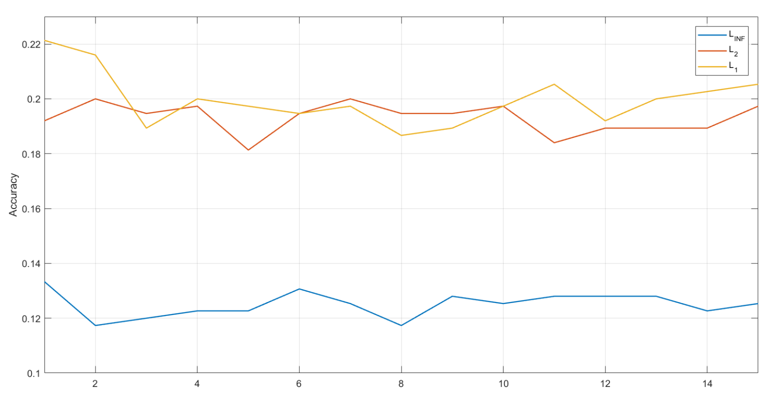

Figure 1: Performances of different distance measures in function of k

As we can see from Figure 1, while L_{INF} can't reach even the 14% of accuracy, L_2 and L_1 can reach up to 20% and beyond. It's interesting to highlight that the latter perform better when we use either just the nearest neighbour or we increase the number of neighbours beyond 10, while L_2 presents its maximum around 7 or 8. In conclusion, considering its stable behaviour on different runs that we performed, we decided to implement our classification using L_2 with k = 7. Of course, given the simplicity of these features, we couldn't expect more than 20% of accuracy. On the following sections we will implement different features extraction methods to improve these performances.

Run 2

For the run 2 of coursework 3, the objective is to develop a set of linear classifiers for the classification task of our test set images. Before training the 15 one vs all classifiers we need to represent our training, and later the test, images as bag of feature histograms, indeed we first need to define a vocabulary of visual words. A dictionary is represented by the centroids obtained from clustering using *k-means*. The clusters are computed over the features collected across each image that makes the training set. The features that are going to be take in account in our model are 8×8 patches, sampled at every 4 pixels in x and y directions. The code for building a vocabulary is reported in `make_vocabulary`, where, after import training images and splitting them between train set (75%) and validation set (25%), a dictionary is creating exploiting MATLAB's *bagOfFeatures* function. It takes as input a training set, the size of vocabulary (which corresponds to the number of *kmeans* clusters) and a custom features extraction function. This function, according to code in `myFeatureExtractor`, resizes input image to a 256×256 picture in order to avoid problem with indexes in MATLAB and extract good features from image. It returns mean-centred, normalized patches representing the image.
At the end of this process, we ended up with k (500 in default case) representative visual words (the centroid of each cluster). Once they have been computed, the following two steps are important for the final image representation before training classifiers. Firstly, we applied an "encoding" process that assigns at each 8×8 patch within an image the closest word (nearest neighbour) in the dictionary. Secondly, each image is represented as a *bag of words* (BOW) which is a histogram of the words, early computed, that together form the image. This type of representation is called a *feature vector* representation of an image. By building a feature vector the system will be invariant to changes in the order of words, namely that it's invariant with respect to rotations, for example, of the image, which is a desirable property to exploit. This process is summarized in `get_image_bag` where the MATLAB's function *encode* incorporates the two processes and produces a histogram that becomes a new and reduced representation of all the images inside training folder.
Finally, 15 one-vs-all classifiers were trained. The classifiers are linear hence suitable for binary classification, but here we have a $15 - way$ classification problem. Therefore, 15 binary 1-vs-all support vector machines are trained in order to find a linear decision boundary. One vs all means that each classifier is trained to recognize object of a class with respect to all others. So, when training one of the classifiers, values from one class are taken and labelled as 1 while other classes' values are labelled as -1, thus bringing back the problem to a binary classification problem for each classifiers combination. When making classification, input data is evaluated over all classifiers and final prediction corresponds to the classifier with highest predicted score. As reported in `run2`, classifiers are trained using *vl_svmtrain* function provided by VL-FEAT [1]. This function receives, among training data and labels, an important tunable parameter *lambda*, that controls the learning rate for each SVM. In this code the prediction accuracy is evaluated over validation set (for different configurations and values for *lambda*) and all images in "testing" folder are classified.
In Fig. 2, it's reported an accuracy comparison considering two dictionaries: *vocabulary 1* has been obtained with default patch size and has 500 words, while *vocabulary 2* consists of 4×4 patches and 800 words. As we can see, as learning rate decreases accuracy increases as expected, since classifiers can learn a better boundary. For high lambda values the accuracy with *vocabulary 2* outperforms the other one but for lower lambda values (from $\lambda = 0.0001$) *vocabulary 1* is better and achieves an accuracy of $\approx 63\%$.

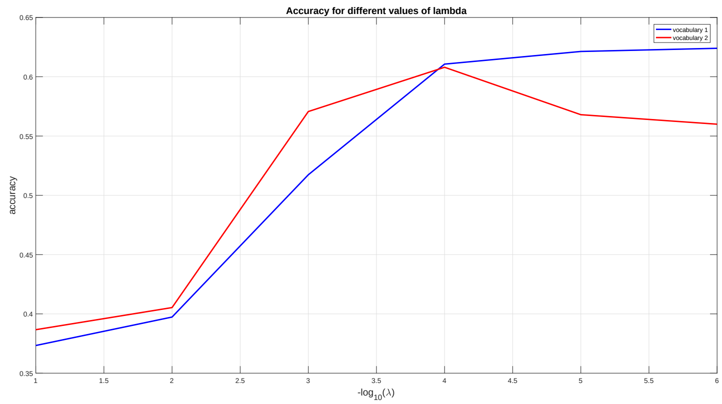

Figure 2: Accuracy comparison for different lambda values and dictionaries.

Run 3

For this last approach we decided to implement **transfer learning**: this means that in order to extract significant features from our images we used layers of a pre-trained neural network. So, to complete our classification architecture, we re-trained just the last layers used for classification on our training set, without doing backpropagation on the previous layers. Surely one advantage of this solution is that allowed us to save a considerable amount of computational time for the training of all the layers of the network.
For this particular problem we decided to use the convolutional neural network (CNN) `alexnet`, winner of the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC2012). At the time, it takes an entire week to train the complete network with the competition dataset (1.2M of images).

References

[1] http://www.vlfeat.org/matlab/matlab.html

Contributions statement

Each member of the group has contributed in equal manner during the code development of the 3 runs and writing of the report. Also considerations regarding each run have been conducted together.