Computational Vision Practicum: Object Recognition

Due on Thursday, January 15th , 2015

Jordi Vitrià

Contents

The evaluation of all Object Recognition exercises will be based on a students written report that must be submitted (via `campusvirtual2.ub.edu`) before Jan 15th, 2015. The report must provide answers for all questions highlighted in red.
Deadline: January 15th, 23:55h by Campus Vitual.
You have to submitt a file "StudentName1+StudentName2_OR.zip" containing a report entitled "Object Recognition Practicum".

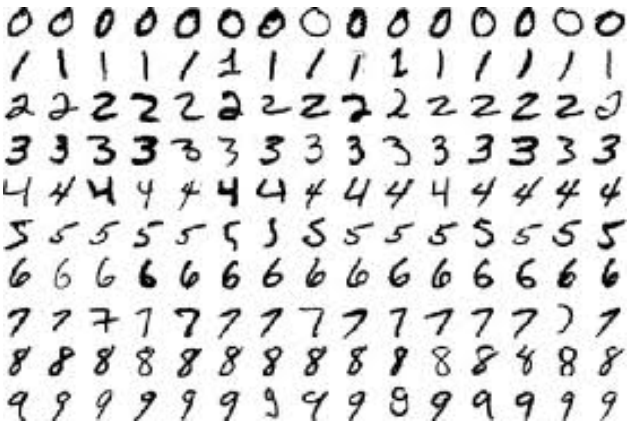
Day 1 and 2

Digit Classification with Support Vector Machines

- Download and install (follow the included instructions) the file `svmdemo-1.4.tar.gz` from Campus Virtual.
- Run svmdemo_linear with different parameters (numbers of points).
- Run svmdemo_rbf with different parameters (numbers of points, gamma).
- Run svmdemo_compare to see the effect of different C values.
- Download from Campus Virtual, in the `svmdemo` directory, the MATLAB structure `mnist_all`. You can inspect its contents with these instructions:

```
load('mnist_all.mat');
imshow(reshape(train7(5,:), 28, 28)')
```

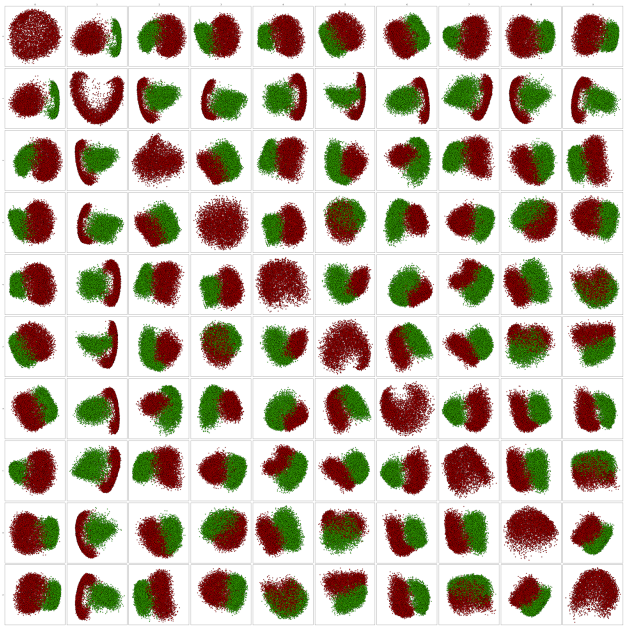
The MNIST database (Mixed National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems. The database is also widely used for training and testing in the field of machine learning. It was created by "re-mixing" the samples from NIST's original datasets. The database contains 60,000 training images and 10,000 testing images (with 784 pixels):



Follow these instructions to built a digit classifier (one-against-all classifier, pixel data) from the MNIST data:

- Train a SVM with 100 random samples per class and a linear kernel (you can use different C soft margin parameters: 0.01, 0.1, 1, 2, 4). Report the accuracy [1] you get on the test set (the rest of samples) and the best C value. Report the confusion matrix you get.
- Train a SVM with 500 random samples per class and a linear kernel (you can use different C soft margin parameters: 0.01, 0.1, 1, 2, 4). Report the accuracy you get on the test set (the rest of samples) and the best C value. Report the confusion matrix you get.
- Train a SVM with 100 random samples per class and a RBF kernel (you can use different C soft margin parameters and different gamma values). Report the accuracy you get on the test set (the rest of samples) and the best C value. Report the confusion matrix you get.

Basically all pairs of numbers can be separated using a linear classifier. And even you project the raw data to two dimensions (f.e. by using PCA), they can pretty much still be linearly separated:



There are some classes that are not obviously separated: 3 vs 5, 4 vs 9, 5 vs 8 and 7 vs 9. But keep in mind, this is just a PCA to two dimensions. Is this set of classes the classes the one with higher values in the confusion matrices you have reported?

Poselets

Download and Install the file April 2013 release (MATLAB) from:
`http://www.cs.berkeley.edu/~lbourdev/poselets/`

- Run the file `demo_poselets.m`. This is a demo file that loads an image, finds the people and draws bounding boxes around them.
- Play with the interactive visualization of the function:

[1] accuracy = (true positives + true negatives)/(true positives + false positives + false negatives + true negatives)

  - Click a point inside the detection window to see which the active poselets are.
  - Click on an active poselet to see its corresponding training images and the contour hypothesis it can generate on the test image.
  - If you click on all poselets you will get all voting hypothesis drawn on the image.

- Look for different images depicting people and change the target image of the function.
- Report an image you have selected with all voting hypothesis drawn on the image.

Day 3 and 4

Image Classification

In image classification, an image is classified according to its visual content (this is not detection, where a bounding box is expected). For example, does it contain an airplane or not. An important application is image retrieval - searching through an image dataset to obtain (or retrieve) those images with particular visual content.
The goal of this session is to get basic practical experience with image classification. It includes: (i) training a visual classifier for five different image classes (aeroplanes, motorbikes, people, horses and cars); (ii) assessing the performance of the classifier by computing a precision-recall curve; (iii) varying the visual representation used for the feature vector, and the feature map used for the classifier.

Getting started

Download the code and data files (`practical-image-classification-code-only.tar.gz` and `practical-image-classification-data-only.tar.gz`) from:
`https://sites.google.com/site/vggpracticals/category-level-recognition`.
The data includes images and pre-computed features.

1. Unpack the code archive. This will make a directory called practical-image-classification.
2. Unpack the data archive in the directory practical-image-classification.
3. Finally, start MATLAB in the directory practical-image-classification.
4. Try running `setup.m` command (type setup without the .m suffix). If all goes well, you should obtain a greeting message.

As you progress in the exercises you can use MATLAB help command to display the help of the MATLAB functions that you need to use. For example, try typing help setup.

Exercise description

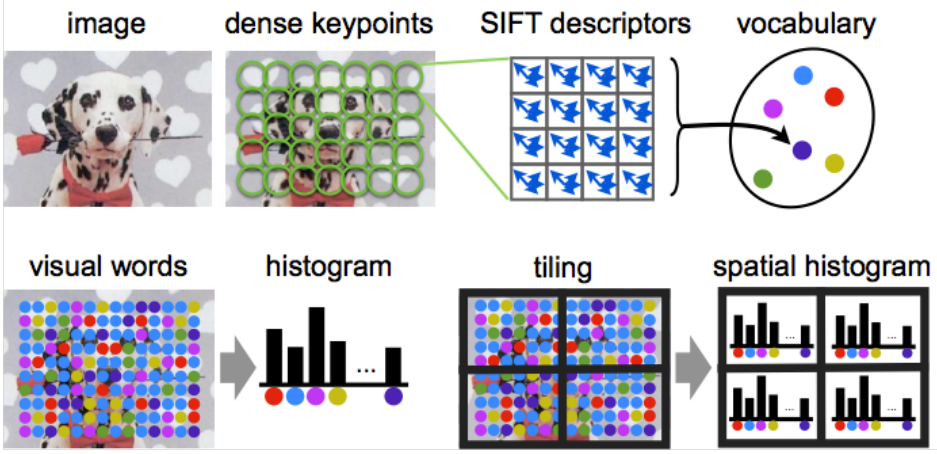
Open and edit the script exercise1.m in the MATLAB editor. The script contains commented code and a description for all steps of this exercise. You can cut and paste this code into the MATLAB window to run it, and will need to modify it as you go through the session.

Stage A: Data Preparation

The data provided in the directory data consists of images and pre-computed feature vectors for each image. The JPEG images are contained in data/images. The data consists of three image classes (containing aeroplanes, motorbikes or persons) and background images (i.e. images that do not contain these three classes). In the data preparation stage, this data is divided as:

	AEROPLANE	MOTORBIKE	PERSON	BACKGROUND
TRAINING	112	120	1025	1019
TESTING	126	125	983	1077
TOTAL	238	245	2008	2096

The feature vector consists of SIFT features computed on a regular grid across the image (*dense SIFT*) and vector quantized into visual words. The frequency of each visual word is then recorded in a histogram for each tile of a spatial tiling as shown. The final feature vector for the image is a concatenation of these histograms. This process is summarized in the figure below:



We will start by training a classifier for images that contain aeroplanes.
The `filesdata/aeroplane_train.txt` and `data/aeroplane_val.txt` list images that contain aeroplanes. Look through example images of the aeroplane class and the background images by browsing the image files in the data directory.

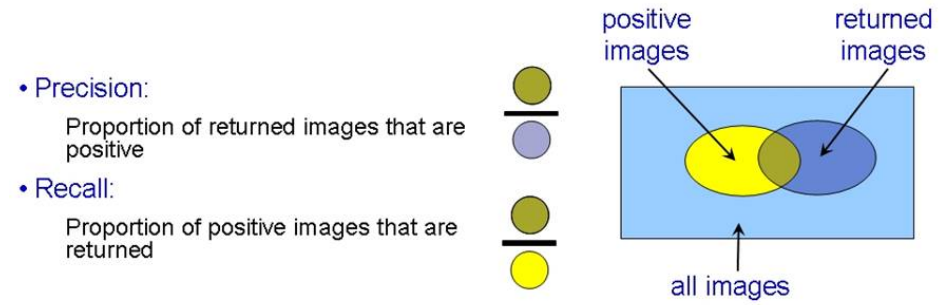
Stage B: Train a classifier for images containing aeroplanes.

The aeroplane training images will be used as the positives, and the background images as the negatives. The classifier is a linear Support Vector Machine (SVM). Train the classifier by following the steps in `exercise1.m`.
We will first assess qualitatively how well the classifier works by using it to rank all the training images. What do you expect to happen? View the ranked list using the provided functiondisplayRankedImageList as shown in excercise1.m.
You can use the function displayRelevantVisualWords to display the image patches that correspond to the visual words which the classifier thinks are most related to the class (see the example embedded in exercise1.m).

Stage C: Classify the test images and assess the performance.

Now apply the learnt classifier to the test images. Again, you can look at the qualitative performance by using the classifier score to rank all the test images. Note the bias term is not needed for this ranking, only the classification vector w. Why?

Now we will measure the retrieval performance quantitatively by computing a Precision-Recall curve. Recall the definitions of Precision and Recall:



The Precision-Recall curve is computed by varying the threshold on the classifier (from high to low) and plotting the values of precision against recall for each threshold value. In order to assess the retrieval performance by a single number (rather than a curve), the Average precision (AP, the area under the curve) is often computed.

Stage D: Learn a classifier for the other classes and assess its performance.

Now repeat stages (B) and (C) for each of the other two classes: motorbikes and persons. To do this you can simply rerun exercise1.m after changing the dataset loaded at the beginning in stage (A). Remember to change both the training and test data. In each case record the AP performance measure.

- Does the AP performance match your expectations based on the variation of the class images?

Stage E: Vary the image representation.

Up to this point, the image feature vector has used spatial tiling. Now, we are going to'turn this off' and see how the performance changes. In this part, the image will simply be represented by a single histogram recording the frequency of visual words (but not taking any account of their image position). This is a *bag-of-visual-words* representation.
A spatial histogram can be converted back to a simple histogram by merging the tiles. Edit exercise1.m to turn the part of the code that does so. Then evaluate the classifier performance on the test images.

- Make sure you understand the reason for the change in performance.

Stage F: Vary the classifier.

Up to this point we have used a linear SVM, treating the histograms representing each image as vectors normalized to a unit Euclidean norm. Now we will use a Hellinger kernel classifier but instead of computing kernel values we will explicitly compute the feature map, so that the classifier remains linear (in the new feature space). The definition of the Hellinger kernel (also known as the Bhattacharyya coefficient) is

$$k(\mathbf{h}, \mathbf{h}') = \sum_i \sqrt{h(i)h'(i)}$$

(compared to a linear kernel $\sum_i h(i)h'(i)$), where h and h' are normalized histograms.
So, in fact, all that is involved in computing the feature map is taking the square root of the histogram values and normalizing the resulting vector to unit Euclidean norm.

- Edit exercise1.m so that the square root of the histograms is used for the feature vectors. Note, this involves writing a line of Matlab code for the training and test histograms.
- Retrain the classifier for the aeroplane class, and measure its performance on the test data.
- Why is this procedure equivalent to using the Hellinger kernel?
- Why is it an advantage to keep the classifier linear, rather than using a non-linear kernel?
- Try removing the L2 normalization step. Does this affect the performance? Why? (Hint: the histogram are L1 normalized by construction)
- Go back to the linear kernel and remove the L2 normalization step. What do you observe?

Note: when learning the SVM, to save training time we are not changing the C parameter. This parameter influences the generalization error and should be learnt on a validation set when the kernel is changed.

Stage G: Vary the number of training images.

Up to this point we have used all the available training images. Now edit exercise1.m the fraction-variable to use 10% and 50% of the training data.

- What performance do you get with the linear kernel? And with the Hellinger kernel?
- Do you think the performance has 'saturated' if all the training images are used, or would adding more training images give an improvement?

Links and further work.

- The code for this practical is written using the software package VLFeat. This is a software library written in MATLAB and C, and is freely available as source code and binary, see `http://www.vlfeat.org/`
- The images for this practical are taken from the PASCAL VOC 2007 benchmark, see: `http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2007/`.
- If there is a significant difference between the training and test performance, then that indicates over fitting. The difference can often be reduced, and the test performance (generalization) improved by changing the SVM C parameter. In Part I, vary the C parameter in the range 0.1 to 1000 (the default is C=100), and plot the AP on the training and test data as C varies for the linear and Hellinger kernels.