School of Engineering and Computer Science
COMP422 – Data Mining, Neural Nets, and Genetic Programming

Project 1: Basic Data Mining Algorithms and Vision Applications

20% of Final Mark — Due: 11:59pm Monday 21 August 2017

Objectives

The overall goal of this project is to review and practise the basic concepts, operators, algorithms and methods commonly used in data mining and computer vision. Specifically, the following ideas, methods and algorithms should be reviewed, understood and practised in the project:

- The process of data mining and knowledge discovery in databases.
- Data mining common tasks, algorithms, and applications.
- Basic concepts and applications of computer vision and image processing.
- Basic operators for preprocessing, edge detection and segmentation.
- Feature extraction in images and object classification.
- Feature types and object recognition algorithms.
- Preprocessing data and creating well-formatted data sets.
- Performance evaluation.

Most algorithms considered in this assignment/project are fairly simple. However, they constitute the basic but important concepts and applications in data mining, pattern classification, and computer vision. The project has been divided into three sections: preprocessing, image mining, and mining data with extracted features.

1 Preprocessing

1.1 Edge Detection [10 marks]

Edge detection is a pre-processing step for many machine vision and image mining applications. In some applications (e.g. analysis of cells) it can be directly used to highlight special patterns in image data (e.g. making it easier for a user to distinguish between infected and non-infected cells).

Apply the Sobel edge detection operator on the image in Figure 1 and briefly explain the procedure in your report. Include both the original and the resulting image in your report. Provide discussion on:

- whether edges always represent the outline of objects; and
- how the Sobel operator performs in noisy areas, corners, and around small object.

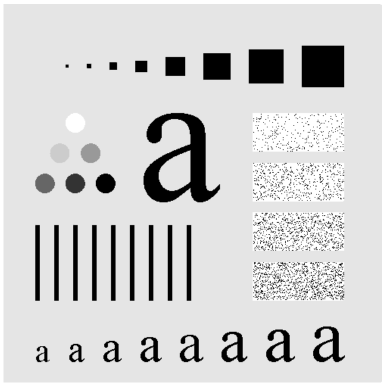

Figure 1: Test patterns (image courtesy of `www.imageprocessingplace.com`). The image is available at `/vol/courses/comp422/images/test-pattern.tif`.

1.2 Noise Cancellation [10 marks]

Many real world data mining applications have to deal with *noisy* data. Since noise does not have a predictable pattern it can considerably affect the performance of mining algorithms on unseen data. Two common types of noise in image data are: *impulse* (salt and pepper) noise and *Gaussian* noise. Figure 2 shows an image with impulse noise. Choose and apply two different image filters (from those discussed in the course) in order to improve the quality of the image (*image restoration*). Choose one of the filters to be based on convolution. Investigate the effectiveness of the noise reduction procedure you used by visually comparing the two noise-reduced images. Discuss which noise reduction filter is more effective in this case and why.

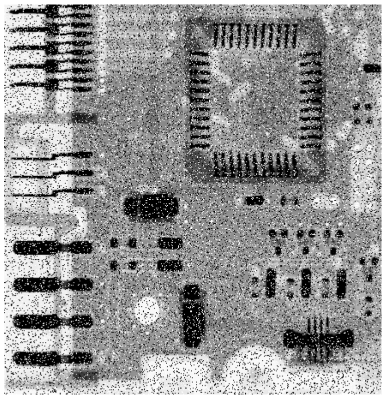

Figure 2: A noisy image of a printed circuit board (image courtesy of Lixi Inc and `www.imageprocessingplace.com`). The type of noise is *salt and pepper* where $p = 0.05$. The image is available at `/vol/courses/comp422/images/ckt-board-saltpep.tif`.

1.3 Image Enhancement [10 marks]

Figure 1.3 shows a blurry image of the moon. Enhance (deblur) the image by using the convolution operator and an appropriate mask. Include both the original and enhanced image in your report and explain your approach.

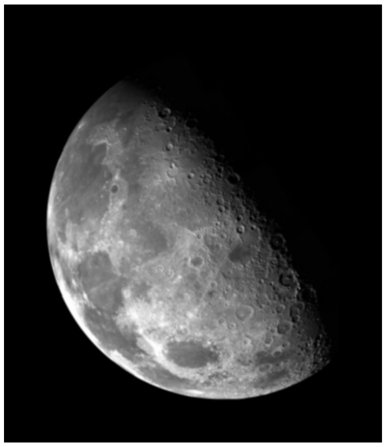

Figure 3: A blurry image of the moon (image courtesy of NASA). The image is available at `/vol/courses/comp422/images/blury-moon.tif`.

2 Mining Image Data

2.1 Mining Space Images [15 marks]

Figure 2.1 shows an image of space taken by the Hubble Space Telescope. Use the techniques you learnt in the course to create a *map* of galaxies in this image. The map will be a binary image (matrix) with the same size as the image. A '1' in the map means that the corresponding pixel in the image is part of a galaxy whereas a '0' means that the corresponding pixel is void (hypothetically). It is preferred to have only large (closer-to-us) galaxies in the map.

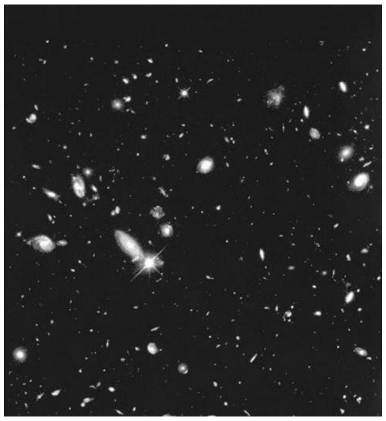

Figure 4: Image from the Hubble Space Telescope (courtesy of NASA). The image is available at `/vol/courses/comp422/images/hubble.tif`.

You first need to smooth the image (using a mean filter) and then apply thresholding. Explain your method and include the original image and the map in your report and explain your solution. Discuss what would happen if thresholding was applied without smoothing the image. What property of the smoothing mask or thresholding must be adjusted to detect only large galaxies?

2.2 Face Detection [35 marks]

Object detection is the process of finding the location of objects of interest (if any) in an image. A common approach to object detection involves moving a window over the image, extracting the value of some predefined features from the window, feeding the values to a classifier and then mark the region (the location of the window) as positive (object of interest) or negative (other).

In this part, you are provided with a collection of images (cutouts) that are the result of of moving a window (of a fixed size) over a number of original images (not provided). The images form a database of faces and non-faces, that has been used extensively at the Center for Biological and Computational Learning at MIT[1]. The database is locally available at `/vol/courses/comp422/image-databases/mit-cbcl-faces`. The tarballs `train.tar.gz` and `test.tar.gz` are for training and test correspondingly and contain sub-directories with the images stored in individual PGM files.

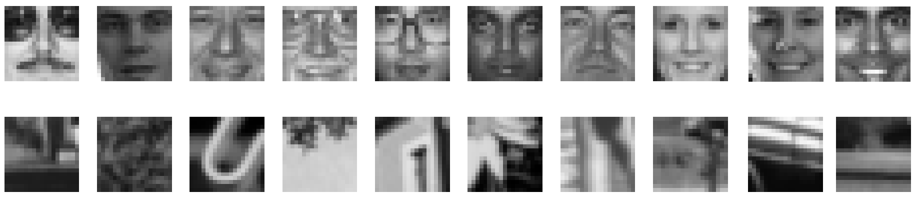

Figure 5: A sample from the MIT-CBL face detection image set.

You are expected to design (extract) a number of features that—you think—are appropriate for face detection, generate instance vectors (feature vectors), write these vectors into tabular data sets, apply data mining algorithms to perform classification, and choose appropriate criteria to measure the performance of your method. More details on these steps follows.

Feature Extraction

For all the images in the training and test set, you need to extract a number of features to form feature vectors for the objects. Design eight or more features. Try to choose features that you think could be useful in separating a face image from a non-face one (a combination of general statistical and domain-specific features). Write a program that extracts the features for each image in the data sets. Describe your method and present a diagram that shows what features and from what parts of the image have been extracted.

[1]CBCL Face Database, MIT Center For Biological and Computation Learning, `http://www.ai.mit.edu/projects/cbcl`

Creating Tabular Data Sets

You need to create two data sets (pattern files): one containing training examples and one containing test (unseen) examples. The data sets must be flat text files. Each line in the data set is an *instance vector*. Each instance vector corresponds to one image (instance) and has two parts: the input part which contains the value of the extracted features for the image; and the output part which is the class label (positive or negative). The class label can be simply a number (e.g. 0 or 1). Choose an appropriate format (ARFF, Data/Name, CSV etc) for you pattern files. Comma Separated Values (CSV) format is a good choice; it can easily be converted to other formats. Include a compressed version of your generated data sets in your submission.

Object Classification and Performance Evaluation:

Train a simple Bayesian classifier (naïve Bayes) using the training data and test its performance on the unseen data. Choose appropriate evaluation criteria (such as *TPF/FPF, classification accuracy* or *error rate*) to measure the performance of the classifier on both training and test data. Provide and discuss the evaluation results. Produce and analyse the ROC curve of the classifier. Also discuss the quality of the features you extracted for this task.

3 Data Mining Using Extracted Features

3.1 Object Recognition: Classification of Hand-Written Digits [20 marks]

In this part you are provided with a large number of extracted features from a collection of hand written numerals ('0'–'9'). 200 patterns per class (for a total of 2,000 patterns) have been digitised in binary images. These digits are represented in terms of the following six extracted feature sets (files):

1. `mfeat-fou`: 76 Fourier coefficients of the character shapes;
2. `mfeat-fac`: 216 profile correlations;
3. `mfeat-kar`: 64 Karhunen-Love coefficients;
4. `mfeat-pix`: 240 pixel averages in 2 x 3 windows;
5. `mfeat-zer`: 47 Zernike moments;
6. `mfeat-mor`: 6 morphological features.

These files are stored in `/vol/courses/comp422/datasets/mfeat-digits`. In each file the 2000 patterns are stored in ASCII on 2000 lines. The first 200 patterns are of class '0', followed by sets of 200 patterns for each of the classes '1' - '9'. Corresponding patterns in different feature sets (files) correspond to the same original character. The source image dataset is not available and therefore the original images are not provided [2].

[2]Using the pixel-dataset (`mfeat-pix`) one could reconstruct the original images (15×16 pixels). This is NOT part of this project though.

The total number of instances is 2000 (200 instances per class). Each file contains 2000 lines, one for each instance. The total number of attributes is 649 (distributed over 6 files, see above). The attributes are SPACE separated. The total number of classes is 10. There are no missing values in the data sets.

Do the following:

- Put the values of these extracted features together to form a tabular data set that contains all the features and a column for class labels (use the information above to generate class labels for instances).
- Divide the initial data set into two halves for training and testing. The number of instances from each class in the two sets should be roughly equal.
- Train a C4.5/J48 decision tree classifier and test it on the test data; report the performance.
- Look at the constructed decision tree; are all the extracted features required for prediction? Discuss this in your report.
- Take the above mentioned steps using only the six morphological features. Print and analyse the confusion matrix. Is the classifier good at predicting all classes? What class seems to be the easiest to predict? What classes seem to be difficult to distinguish from each other?
- Compare the results using the two different sets of features (all features vs morphological features only) and make your observations/conclusions.

Note: Do not copy the data files to your home folder; they are too large. Use `/local/scratch/`, a USB stick, or similar means to store the files. On the other hand, keep all your programs including those that preprocess the original data files in your home directory. Do not submit the original data set along the project.

Submission Guidelines

Preparing Your Submission: Requirements

Project report/document: This document will allow you to practise your technical writing skills. It should include the detailed description of the methods, algorithms and criteria used in your project, appropriate presentation, comparison and analysis of the results for each task. In general, **readers should be able to understand what you have done without looking at your program codes.**

Programs and other required files: This part should include the programs you wrote for performing all the tasks. Add necessary comments for each program. If you select programs from the course public directory, write the detail parameters you used. ANSI C/C++/Java are recommended, other programming languages are also accepted. **All the programs should be able to run on the `unix` system in the school.** A `makefile` should be provided and a `readme` file is also required (describe how to run your programs). You should also submit a `script` file to show how your programs run properly. If your programs can not properly run, you should provide a `buglist` file.

Submission Method

The printed hardcopy of the project report/document should be submitted to the hand-in box labelled COMP422 in the main corridor near the Computer Science labs in level 2 of the Cotton Building by the due time.

The project related programs, files and the PDF version of the project report/document should be submitted electronically through web submission system from the COMP422 course web site by the due time.

Late Penalty

In the usual case, you should have no problem in completing this project on time, and requests for extension of the deadline will only be granted in rare circumstances. Unless an arrangement has been approved, projects handed in late will be penalised 10% per day, and will not be accepted beyond a week after the deadline.