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

Project 2: Neural Networks, Genetic Programming and Feature Manipulation

20% of Final Mark — Due: 11:59pm Friday 19 October 2018

Objectives

The overall goal of this project is to review and practise the concepts of neural networks (NNs), genetic programming (GP), and Particle Swarm Optimisation (PSO) and the way they are used in solving data mining, computer vision, and optimisation problems. Specifically, the following ideas, methods and algorithms should be reviewed, understood and practised in the project:

- Main concepts and paradigms of machine learning, particularly supervised learning;
- Main aspects of object classification or detection, especially domain independent approaches (including the use of raw image pixels and pixel statistics as inputs to neural networks);
- Main concepts, types of neural networks and their applications;
- Methods of determining the network architecture;
- Design, implementation and experiments with neural networks for real world applications;
- Network training, overfitting and generalisation;
- Network training termination criteria;
- Main idea of GP techniques;
- Program representation and generation;
- Determination of a terminal set and a function set for a given problem;
- Determination of a appropriate fitness function for a problem;
- Choice of a set of parameters for GP learning; and
- Different classification techniques
- Main idea of PSO
- Determination of the fitness function, particle encoding and topology for a given optimisation problem, and
- Choice of appropriate parameter values in PSO;

- Data preprocessing: Ranking and selection of features
- Data preprocessing: Feature construction
- Use data preprocessing techniques to improve the quality of the input data
- Improving your scientific analysis and writing skills

This project involves applying neural networks, genetic programming and particle swarm optimisation to a number of tasks in different application domains and writing them up.

Unless specified otherwise, the neural network for each task can be either a standard multilayer feed forward neural network (with some variations when necessary) or convolutional neural networks.

The network training algorithm can be the standard back propagation algorithm or any other variations. The program structure in GP is either the standard tree structure (numeric expression) or strongly typed genetic programs.

For any question of this project, you can **make your own assumptions if necessary**. The data sets you need to work with have been described at the end of this document.

1 XOR Problem [20 marks]

For the **xor** problem with the four following patterns:

Input 1	Input 2	Desired Output
1	1	0
1	0	1
0	1	1
0	0	0

- Use multilayer feed forward neural network for this task. Determine the network architecture, choose the learning parameters, describe the training process, and report the result in an appropriate form.
- Use GP techniques for this task. Determine the terminal set and function set, design a fitness function, determine parameters and evolutionary process termination criteria, and report three best genetic programs that perfectly perform this task.
- Compare and analyse the two methods and results, and draw your conclusions.

2 Digit Recognition [20 marks]

This question concerns applying neural networks to nine digit-recognition tasks. Each task involves a collection of binary images (pictures) of printed digits $(0, 1, \ldots, 9)$. The tasks are classification problems of increasing difficulty, as shown in Table 1. In all of these recognition problems, the goal is to automatically recognize which of the 10 classes (digits 0, 1, 2, ..., 9) each pattern (digit example) belongs to. Except for the first task which contains clean

images (no noise), all data examples in the other eight task have been corrupted by noise. The noise is generated by randomly flipping a pixel (from 0 to 1 or from 1 to 0). The *noise ratio* determines what proportion of pixels in an image have been flipped.

There are nine files corresponding to the nine tasks. There are 100 examples for each digit; that is 1000 images (examples) in each file (task). The files are available at `/vol/courses/comp422/datasets/digits`. The images are binary and of size 7×7. Each image is represented as one row in the file. The noise ratio is given by the two numbers `nn` in the file names. Out of 1000 patterns in each task 500 are used for training and 500 for testing.

Table 1: Nine digit-recognition tasks (ten classes each).

Task	File name	noise ratio	Total patterns	training set	test set
1	digit00	0%	1000	500	500
2	digit05	5%	1000	500	500
3	digit10	10%	1000	500	500
4	digit15	15%	1000	500	500
5	digit20	20%	1000	500	500
6	digit30	30%	1000	500	500
7	digit40	40%	1000	500	500
8	digit50	50%	1000	500	500
9	digit60	60%	1000	500	500

Example digits from the 9 tasks are shown in Figure 1. The example images are enlarged to give a clear view of the digits. The 9 rows of examples correspond to the 9 recognition tasks in Table 1. The first 3 tasks—one with clean data and two with only 5% and 10% noise ratio—are relatively straightforward for human eyes, though there is still some difficulty in distinguishing between "3" and "9". As the noise ratio increases, it becomes more difficult to classify the patterns, even if humans can still recognize the majority (e.g. task 4 and task 5). From task 6 onwards, it is very difficult (or even impossible), for human eyes to distinguish between patterns.

In this question, you need to do **at least four tasks** including tasks 4 and 6 and two or more of your own choice.

- For each task, develop a neural network classifier with an appropriate architecture which does the best possible job of correctly recognising unseen digits. Consider the pattern file format, network architecture, learning parameters, termination criteria, etc.
- Use the nearest neighbour method to perform your tasks. Consider the pattern file format.
- Present the results of the two methods in an appropriate form. Note that the results should be an average (or mean and standard deviation) of multiple runs (such as 10 runs).
- Compare the two methods and results and draw your conclusions.

Figure 1: Example patterns (digits) from each task. The nine rows of examples correspond to the nine tasks.

3 Image Classification using Neural Networks [20 marks]

This question concerns designing *neural networks* for image classification. You will use the Jaffe dataset, which is to classify a set of images based on the different emotions shown in each image.

In the provided `jaffe.zip` file, there are seven folders, each containing a set of images with a different emotion (i.e. class label), namely "an" (angry), "di" (disappointed), "fe" (fear), "ha" (happy), "ne" (neutral), "sa" (sad), and "su" (surprised). This is a multi-class (7-class) classification task.

- Split the dataset into *training* and *test* sets in a proper way. Describe how you split the data and justify your split approach clearly in your report.
- Use a *multilayer feedforward neural network* for this task. For input layer, you can either design features manually or use the pixels directly. Determine the network architecture, choose the learning parameters, describe the input features, network architecture, training and test processes, and the training and test results in an appropriate form.
- Use a convolutional neural network (CNN) for this task. Determine the CNN architecture, such as the number and type of hidden layers, size of weight matrix for feature maps, number of pixels shifted, activation function, etc. Describe your configuration, and *justify your configuration* (why you choose this configuration) clearly in your report.
- Train the CNN with appropriate parameters (e.g. learning rate, momentum, stopping criteria). Describe the training algorithm and parameter values you used in your training process clearly in your report.

- Test the trained CNN on the test set. Compare and discuss the test results obtained by the feedforward neural network and the CNN you developed. Draw the *learning curves* of the two neural networks and discuss the differences between them. Describe your conclusions clearly in the report.

4 Symbolic Regression Problem [10 marks]

In this question, your task is to build a genetic programming system to automatically evolve a number of genetic programs for the following extended regression problem:

$$f(x) = \begin{cases} \frac{1}{x} + \sin x & , \quad x > 0 \\ 2x + x^2 + 3.0 & , \quad x \leq 0 \end{cases}$$

Determine and describe the terminal set, the function set, the fitness function, the fitness cases, necessary parameters, and the evolutionary termination criteria. Report the best three genetic programs evolved and the corresponding performance, and draw your conclusions.

5 Function Optimisation [10 marks]

In this question, your task is to use particle swarm optimisation (PSO) to search for the minimum of the following two functions, where D is the number of variables, i.e. x_1, x_2, ..., x_D.

Rosenbrock's function:

$$f_1(x) = \sum_{i=1}^{D-1} \left(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2\right), x_i \in [-30, 30]$$

Griewanks's function:

$$f_2(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} cos(\frac{x_i}{\sqrt{i}}) + 1, x_i \in [-30, 30]$$

For D = 20, do the following:

- Choose appropriate values for c_1, c_2, w, and population size,
- Determine the fitness function, particle encoding, topology type, and stopping criterion in PSO
- Since PSO is a stochastic process, for each function, $f_1(x)$ or $f_2(x)$, repeat the experiments 30 times, and report the mean and standard deviation of your results, i.e. $f_1(x)$ or $f_2(x)$ values.
- Using the same settings in PSO, compare the performance of PSO on the Griewanks's function when $D = 20$ and $D = 50$
- Analysis your results, and draw your conclusions.

6 Feature Construction [20 marks]

First apply the naïve Bayes and Decision Tree (C4.5) classifiers to the Balance Scale and Wine Recognition datasets and measure the classification performance using 10-fold cross-validation. Then develop a GP-based feature construction algorithm to construct features for the two problems. Feed the constructed features to these classifiers and report changes in their performance. Consider the following points:

- Design an appropriate fitness function.
- Use the best individual out of the 10 runs to transform the data. Include the program (the best individual) in your report (in the form of Lisp code, tree, ...).
- Report and analyse changes in performance.

7 Feature Selection [10 marks]

Choose two single feature ranking algorithms (e.g. Information Gain and Chi-square) and apply them to Wisconsin Breast Cancer and Sonar data sets. Choose a classifier (e.g naïve Bayes) and a number n representing the desired number of selected features (e.g. 5). Do the following:

1. Compare the performance of the classifier using all the features vs n top ranked features from each method. Analyse your results.
2. Use a wrapper approach to select n features and then measure the performance of the classifier using the selected features. Compare the result with the filter (single-ranking) approach and analyse your findings.

The data sets have already been downloaded and are available in the course directory (`/vol/courses/comp422/uci-datasets`). The format of the data sets have been changed so they are readable by Weka and also easy to process with your programs. For the sake of simplicity all instances with missing values—rows in the data set where there is no value for one (or more) feature(s)—have been removed. The data sets do not come with a test set and they contain relatively small number of instances. Therefore, to test the performance of an algorithm on these datasets, you need to use the 10-fold cross-validation technique.

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 program files (including the source code) and the PDF version of the project report/document should be submitted electronically through the 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.