Machine Learning and Neural Computing: Data Classification Coursework

Date to be handed in: by 12 noon 20/03/2020

Introduction

Fozziwig's Software Developers have contracted you to explore the possibility of an automated software defect prediction system. They want to know if developing such a system would be cost-effective, based on the predictive *accuracy* (also known as the *correct classification rate*) that you can achieve with a sample of their data.

Static code metrics are measurements of software features. They can be used in an attempt to quantify various software properties which may potentially relate to defect-proneness, and thus to code quality. Examples of such properties and how they are often measured include: size, via lines of code (LOC) counts; readability, via operand and operator counts (as proposed by [1]); and complexity, via linearly independent path counts (this relates to the control flow graph of a program, and was proposed by [2]).

The data that you have been given contains the static code metrics for each of the *functions* which comprise a software system. This system was developed by Fozziwig's Software Developers several years ago. As well as the metrics for each function, it has also been recorded whether or not a fault was experienced in each function. This data came from the software testers who examined the system before it was publicly released.

You have been given two *labeled* data files, a training data set (`trainingSet.csv`) and a test data set (`testingSet.csv`). Each data set contains 13 features (each one a software metric). Class labels are shown in the last column of each file: a value of '+1' means 'defective' (the software module contained a defect (fault)) while a value of '-1' means 'non-defective'. Note that this is clearly a simplification of the real world, as both fault quantity and severity have not been taken into account.

The training set you have been given consists of 400 instances, 196 labeled as defective and 204 labeled as non-defective. This is known as a *balanced* data set, as the ratio of the number of instances in each *class* is almost equal to 1. The test set is also balanced,

and contains 100 instances. You can assume that the data is of satisfactory quality and requires no preprocessing / data cleansing **other than normalisation**.

To classify the data you will be using Support Vector Machines (SVMs). The **type** of SVM you need to use is the **C-SVC (Cost-Support Vector Classifier)** and the **kernel function** you should use is the Gaussian **radial basis function (RBF)**.

Software Required

For this coursework you will need to use Python (in version 3 and above). You can use functions from the following packages: Numpy, Pandas, Matplotlib, Seaborn and Sklearn. Your practical session notes should be very useful - these are all available on *Canvas*.

Task 1 - Data Exploration (12 marks)

In this task, you need to use Principal Component Analysis (PCA) to understand the characteristics of the datasets.

- Use Pandas to load both the training data set and the test set (1 mark).
- Show one scatter plot, that is, one feature against another feature on the training set. It is your choice to show which two features you want to use (2 marks).
- Normalise the training set and the test set using **StandardScaler()** (Hint: the parameters should come from the training set only) (2 marks).
- Perform a PCA analysis on the scaled training set and plot the scree plot to report variances captured by each principal component (2 marks).
- Obtain projections of the test set by projecting the scaled test data on the same PCA space produced by the training set (1 mark).
- Plot two subplots in one figure: one for projecting the training set in the PC1 and PC2 projection space and label the training data using different colours in the picture according to its class; the other one is for the test set in the same PC space and also label the test data using different colors according to its class (2 marks).
- Write a short critical analysis (no more than 100 words) on your observations. (2 marks)

Task 2 (2 marks)

- Divide the training dataset into a smaller training set (II) and a validation set using **train_test_split** and report the number of points in each set. Usually, we use 20%-30% of the total data points in the whole training set as the validation data. It is your choice on how to set the exact ratio. (1 mark)
- Normalise both the training set (II) and the validation set (Hint: the parameters should come from the training set (II) only).(1 mark)

Task 3 - Choosing the Most Suitable Parameters (6 marks)

When using the C-SVC SVM with the Gaussian radial basis kernel there are two tunable parameters, *C (cost)* and γ *(gamma)*. To achieve the highest classification rate possible it is very important to search for an optimal pair of these values.

You have been given the following combinations: [C=50, γ=0.01], [C=50, γ=10], [C=5, γ=1], [C=100, γ=0.01], [C=100, γ=10] and [C=100, γ=1].

You should train an SVM model for each combination from the given 6 combinations and then test it on the normalised validation set. The accuracy rate for each combination on the validation set should be reported. Finally, you need to select the best combination of parameters and report your result.

Task 4 - Non-linear Classification (8 marks)

- Basic task (2 marks)
  You should now be in a position to further test your model with the selected parameters by classifying the test data. With the normalised whole training set as the input file, you will need to train an SVM model with the suitable parameter values discovered for C and γ during Task 3. When the classification model is built you will then need to use it to classify the normalised testing set, and report the accuracy rate.
- Advanced task (6 marks)
  Write a Python function that can locate false-positives, that is those patterns originally labeled as non-defective which are incorrectly predicted as defective, and report the results on the test set (3 marks).

Summarize your findings and write your conclusions in critical thinking. For example, can you find any reason/reasons as to why you think those instances are misclassified? (3 marks).

What to Submit

The deliverable for this coursework is an experimental report with no more than 10 pages including appendix and less than 1500 words (Please use a single column format. Font size should be set to 11 or 12 point, and the line spacing should be set to 1.5 lines or single) in the PDF format. Please put your Python code in Appendix. All submitted screenshots should be clear and readable.

Overall, there are **2 marks** for presentation and clarity of the submitted report. Note that you must do this coursework individually. You need to submit your coursework via Canvas.

References

[1] Halstead, M. H. *Elements of Software Science (Operating and programming systems series).* Elsevier Science Inc., New York, NY, USA, 1977.

[2] McCabe, T. J. A complexity measure. In *ICSE '76: Proceedings of the 2nd international conference on Software engineering* (Los Alamitos, CA, USA, 1976), IEEE Computer Society Press, p. 407.