|  |  |
| --- | --- |
| **NGUYEN, Réal** | **COMP 472 SEC F** |
| **ID#27566263** | **DUE 18/11/2018** |

**MINI-PROJECT 2 REPORT**

**1. INTRODUCTION**

The purpose of this report is to analyse the performance of three machine learning algorithms using Weka, to recognise characters from the English alphabet and a subset of the Greek alphabet, along with the impact on their performance when experimenting with different hyper-parameters. This report will include an introduction to the algorithms and the hyper-parameters that I experimented with, an analysis of the data collected while experimenting, and what I have learned from this mini-project.

**2. BASIC EXPERIMENTAL SETUP**

- Describe the algorithms that you chose to experiment with. Do not re-explain the theory of ML

models we have seen in class, just indicate the hyper-parameters you used; and why you chose them.

If you selected an ML model we have not covered in class, explain in a paragraph or two how it

works and why you chose it.

- If you experimented with various parameters, explain what you did and why you did it.

- Describe any additional code that you may have written.

In this section, assume that “better performance” implies better accuracy, better precision, better recall, and better F-measure. Given the nature of this mini-project, false positives and false negatives have equal weight, that is, precision and recall have the same importance. Assume that every hyper-parameter is tested in isolation, with all the other hyper-parameters set to their default values.

Run times are negligible, unless noted. The actual results of changing the hyper-parameters will be discussed in the next section.

**2.1 Naïve Bayes Classifier**

The simplest algorithm of the three, the Naïve Bayes classifier does not have any hyper-parameters that directly affect its performance, whether negatively or positively. Thus, the model used to train the algorithm is the same as for validation and testing, for both data sets.

**2.2 J48 Decision Tree**

*Hyper-parameters that improved performance:* Confidence factor, minimum number of objects, unpruned

*Hyper-parameters used:* Unpruned (DS1), confidence factor (DS2), minimum number of objects (both)

*Why?* Better metrics, shorter tree

Before looking at the hyper-parameters used, let us look at the choice of algorithm for the decision tree (DT). Weka has a multitude of decision tree algorithms to choose from, and J48 proved to be the most reliable in terms of performance. Being based on the C4.5 DT algorithm, it is one of the most popular machine learning algorithms, being ranked #1 in popularity according to a paper by Wu et al. (2008). Some other DT algorithms, like REP Tree, did not perform as well, or had a 0% rate of correctly identifying some classes. Notably, the Random Tree and Random Forest algorithms were very obviously overfitting the training data sets, having either 99.9% or 100% accuracy every time.

For J48’s hyper-parameters, I have experimented with all of them that are not obviously only there for convenience or bookkeeping (e.g. debug, batch size). However, I will only give details about the ones that have positively impacted J48’s performance: confidence factor, minimum number of objects, and unpruned. Note also that for DT algorithms, having a shorter DT is favourable.

The confidence factor (CF) hyper-parameter incurs more pruning the lower it is. In other words, the lower the CF is, the shorter the DT is. This will also slightly increase performance, up to a certain threshold.

The minimum number of objects (labelled minNumObj in Weka) hyper-parameter is the minimum number of instances per leaf the higher it is. Like the CF, it will increase the performance of the algorithm up to a certain threshold and will consistently reduce the size of the tree the higher it is.

The unpruned hyper-parameter leaves the DT unpruned. This will remove the confidence factor, incur a larger DT, and affect the performance. For the first data set, the performance is slightly better, for the second one, slightly worse.

**2.3 Multilayer Perceptron**

*Why this algorithm?* Seen in class, known to be good with more complex problems

*Hyper-parameters used:* Decay

*Why?* Better metrics

**3. ANALYSIS OF RESULTS**

- State the results of all your experiments. A table would be a good format here. For each method and

for each data set, show the results.

- Analyse your results. For example, does the same algorithm perform the same way for different data

sets? Why? What if you change some hyper-parameters?

*[Insert data comparison table here]*

*Refer to the spreadsheet you made. Pay attention to the metrics, the effects of the hyper-parameters, and how performance changes on different data sets. Note that DS2 is unbalanced.*

**3.1 Naïve Bayes Classifier**

**3.2 J48 Decision Tree**

**3.3 Multilayer Perceptron**

**4. CONCLUSION AND FUTURE WORK**

Draw final conclusions from your experiments. Do these results surprise you or are they expected?

If you were to continue working on this project, what do you feel would be interesting to investigate?

Are there questions that you would like to investigate more, if you had the time and the energy?

**5. REFERENCES**

<https://machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/>

Official Weka PDF and YouTube MOOC tutorials

Weka JavaDocs documentation

Xindong Wu; Vipin Kumar; J. Ross Quinlan; Joydeep Ghosh; Qiang Yang; Hiroshi Motoda; Geoffrey J. McLachlan; Angus Ng; Bing Liu; Philip S. Yu; Zhi-Hua Zhou; Michael Steinbach; David J. Hand; Dan Steinberg (2008) “Top 10 algorithms in data mining.” Knowledge and Information Systems vol. 14, pp 1-37.

Sam Drazin; Matt Montag “Decision Tree Analysis Using Weka.” University of Miami.