Multipipes: Exploring Disjuntive Classifications in Hyperpipes

[In an exciting manner] [*]

Aaron Riesbeck
West Virginia University
100 Address Lane
Morgantown, WV 26505
ariesbeck@theriac.org

Adam Brady
West Virginia University
100 Fake St.
Morgantown, WV 26505
adam.m.brady@gmail.com

ABSTRACT

This paper explores classifiction with disjunctive sets using a modified form of HyperPipes called MultiPipes. Rather than apply HyperPipes it's intended sparse datasets, we find that it's application to non-sparse, many-class datasets typically results in several tied classification scores which we then union into a disjunction. This union presents interesting possibilities in it's high accuracy in containing the target class. Although we initially cannot predict single classes, we find that these disjunctions often eliminate large portions of possible classes. Essentialy we aren't certain what the class is, but we are very certain of what the class is not. The rest of the paper explores two alternative strategies with MultiPipes. The first involves methods of reducing the disjunctive sets to single classifications. The second considers growing the disjunctive sets to optimize the accuracy of containment vs. set size.

Keywords

HyperPipes, disjoint sets, LaTeX, multiple classes, indecisive learners

1. INTRODUCTION TO HYPERPIPES

Background on hyperpipes, description of algorithm, benefits, trade-offs, relevant applications (spare datasets)

1.1 Pseudocode for hyperpipes

1.2 The Problem with HyperPipes

1.2.1 Tied Classes

During development of the original HyperPipes implementation it was noted that the code only kept track of the most recently seen class with a score equal to or greater than the

[*]A full version of this paper is available as *Author's Guide to Preparing ACM SIG Proceedings Using LaTeX2$_\epsilon$ and BibTeX* at `www.acm.org/eaddress.htm`

Program 1 HyperPipes Pseudo Code.

```
procedure RunHyperPipes($Nothing$)
    $HyperPipes := array[]$
    $Guessed := 0$
    $GuessedCorrect := 0$
    for all $LineinDataFile$ do
        $Guessed++$
        $GuessedCorrect += Classify(MyHyperPipes, Line)$
        $HyperPipes := AddExperience(Line, HyperPipes)$
    end for
    $Accuracy := GuessedCorrect/Guessed$
end procedure
```

Program 2 HyperPipes Classify Pseudo Code.

```
procedure Classify($HyperPipes$,$Line$)
    $BestScore := 0$
    $BestClass := []$
    for all HPipe in HyperPipes do
        $HPipeScore := 0$
        for all Attr in Line do
            if $Attr <= HPipe.Attr.max$ && $Attr >= HPipe.Attr.min$ then
                $HPipeScore++$
            end if
        end for
        if $HPipeScore >= BestScore$ then
            if $HPipeScore := BestScore$ then
                $BestClass := BestClass + HPipe.class$
            else
                $BestClass := array(HPipe.class)$
                $BestScore := HPipeScore$
            end if
        end if
    end for
    if $BestClass$ contains $Line.class$ then
        return 1
    end if
    return 0
end procedure
```

Program 3 HyperPipes Classify Pseudo Code.

```
procedure AddExperience($HyperPipes$,$Line$)
    $CurrentPipe := FindPipe(Line.class, HyperPipes)$
    if $null\ CurrentPipe$ then
        $HyperPipes := HyperPipes + CreateHyperPipe(Line.class)$
        $CurrentPipe := FindPipe(Line.class, HyperPipes)$
    end if
    for all Attr in Line do
        $CurrentPipe.max := max(CurrentPipe.max, Attr)$
        $CurrentPipe.min := min(CurrentPipe.min, Attr)$
    end for
end procedure
```

largest score seen to that point. This means that when running HyperPipes if multiple classes tied in score the class tested last would be blindly chosen. We found that this anomoly caused our HyperPipes implementation to prefer the last classes discovered once it learns many rows.

1.2.2 Susceptible to Over Fitting

HyperPipes has the ability to have very low overhead costs in terms of memory. However, this benefit is overshadowed by HyperPipes susceptibility to overfitting. As more and more rows are added as experience to a HyperPipe its min and max values with slowly approach the min and max of the attribute alone regardless of class. This causes HyperPipes guessing ability to completely fall apart. As the min and max values for the attributes in every HyperPipe expand the scores become very high and the likelyhood of encountering the issue in the previous sub-section becomes greater. That is, as each of the HyperPipes expand their min and max attribute values start to become so similar to eachother that they all begin to acheive common scores.

1.2.3 Outliers Ruin Scoring

As shown in the AddExperience function pseudocode a HyperPipe contains a set of bounds for each attribute. These bounds indicate the min and a max value ever seen for this attribute for this class. This becomes a problem when an outlier for that attribute occurs. Imagine after 10,000 rows your min is 23 and your max is 45. You encounter a new row where this attributes value is 431. This causes your new max value to be 431. This can be a major pitfall if this is the only time a number this large is encountered for this class. You have now expanded this HyperPipe to a max so large that this attribute loses its ability to classify (It always matched on this attribute for this class).

1.2.4 Memory Management

The original HyperPipes required prior knowledge of the dataset. In other words it created all of the necessary HyperPipes at the beginning and modified them as it went through the dataset. This means that if a new class is created the entire algorithm would have to start from the beginning with this knowledge that this new class existed. If you were able to control the outliers and the over fitting for this algorithm you could use this type of classifier in a real world situation where ne classes are constantly being discovered.

1.3 Patching HyperPipes

Plumbing reference

Explain appending fix

2. NARROWING VS. CLASSIFYING

Why narrow when you can classify?

3. PRELIMINARY RESULTS

Description of results

- incremental learning
- batch learning
- weighted distance
- centroids via overlap
- increasing alpha

3.1 Disjoint Learning

nb vs. multipipes on >1 dataset (incremental) nb vs. multipies on >1 dataset (batch) (see menzies.us/iccle/?nb chart for dataset scores comparison)

size of sets returned relative to number of total classes

3.2 Breaking the Ties

Description of weighted distance measure

graph of weighted distance classification accuracy

Description of centroid acquisition from overlap

graph of centroid learning results

3.3 Casting a wider net

Description of alpha value

Purpose of alpha value for expanding class set

Results of expanding alpha (graph)

Analysis of growth in enclosure with alpha changes

4. PRELIMINARY CONCLUSIONS

WE CONCLUDE

4.1 References

Generated by bibtex from your .bib file. Run latex, then bibtex, then latex twice (to resolve references) to create the .bbl file. Insert that .bbl file into the .tex source file and comment out the command `\thebibliography`.