An Improved Oscillating-Error Classifier with Branching

KIERAN GREER
Distributed Computing Systems
Belfast, UK
kgreer@distributedcomputingsystems.co.uk,
http://www.distributedcomputingsystems.co.uk

Abstract: - This paper extends the earlier work on an oscillating error correction technique. Specifically, it extends the design to include further corrections, by adding new layers to the classifier through a branching method. This technique is still consistent with earlier work and also neural networks in general. With this extended design, the classifier can now achieve the high levels of accuracy reported previously.

Key-Words: - classifier, oscillating error, neural network, multiple layers, branching, cellular automata

1 Introduction

This paper extends the earlier work, based on an oscillating error correction technique [8]. The method uses an error correction update that includes a very simple rule, of either adding or subtracting the error adjustment, based on whether the variable value is currently larger or smaller than the desired value. This has relations with cellular automata [3], where the small add or subtract decision gives the classifier an added dimension of flexibility. The results reported in the first paper were unusually good over a wide range of datasets and it was subsequently found that an error had been made in how the classifier decides on the correct output category. The earlier paper measured the error amount between the desired output value and the value produced by the corresponding classifier. If the error was small enough, then the classification was considered to be correct. When training the classifier, the data rows for each category would be put together and averaged. The classifier would then try to learn these average values, but that would lead to distinct weight sets for each output category. It was overlooked that even if the desired output category correctly classified the input data row, one of the other category weight sets could produce an even smaller error. If an unknown input is presented, then the classifier has to choose the output category with the smallest error and so the results reported in the first paper were misleading.

Because the method shows promise, this paper considers options for extending the first design, to try to improve on that failing. Specifically, this paper considers extending the design to include further corrections, by adding new layers to the classifier through a branching method. The reasons for the addition is quite clear and can even be put in terms of a logical proof. With this extended design, the classifier can now achieve the high levels of accuracy that were reported previously.

The rest of this paper is organised as follows: section 2 introduces some related work. Section 3 describes the classifier with a new branching technique and section 4 gives a very basic proof for why it should work. Section 5 re-runs the test set to verify the classifier's accuracy, while section 6 gives some conclusions to the work.

2 Related Work

This paper is very much an extension of the oscillating error design of [8] and that paper should be read first. One novelty in the process is the heavy use of averaged values from batch datasets and another is the inclusion of a very simple rule, that a cellular automaton might use, as described in the Introduction. While the classification error can reduce to a very small amount, an incorrect classifier's averaged value can still match closer to an input data row than the correct classifier's value. This suggests that each input category may have more than 1 averaged value in it, or that natural bands or boundaries may exist. The idea of a banded input [9] and also a shape [10] to the averaged value sets is therefore important. Other papers on this topic include [1].

The paper [3] presents a proof that dynamic cellular automata are flexible enough to simulate arbitrary computations, which means algorithms in general. As they describe, this has been put in the

context of state machines, where classical algorithms were axiomatized by Gurevich [11], who also showed that a simple, generic model of computation, called abstract state machines (ASMs), suffices to emulate state-for-state and step-for-step any ordinary (non-interactive, sequential) algorithm. Their proof and general theory could be of interest, but this paper is more concerned with a specific process for categorical classification, albeit one that also exhibits a high level of genericity.

Another interesting fact could be the parallelization of the algorithm. Each classifier is mostly self-contained without too much interaction with the other classifiers. That should make it easier to parallelise the algorithm ([17], for example).

3 Extending the Classifier with Dataset Branching

As is typical with neural networks, one option would be to add new levels or layers to the classifier, to continually refactor the feature set. The first level is what was described previously [8], but in this paper, separate classifiers are used to classify each distinct category and so weight sets are not re-combined between each matrix phase. The dataset is therefore split into groups for each distinct category and the data row values averaged. There are *x* classifiers in the first level, one for each category. They are trained to reduce the error for the averaged row values of their category only. The whole train dataset is then passed through all of the classifiers and each produces an error for each data row. After this test phase, there is a list of data rows that each classifier has produced the smallest error for, with relation to its own category. Most of the rows would be for the correct category group, but some would be for other classifier groups. The extension here is therefore to add a new level to the classifier, to refine it with respect to the incorrectly classified data rows.

The second level therefore uses a subset of the whole dataset that is specifically only the data rows that the first level classified as closest. It is a whole new layer however, with classifiers for each category of data row that it is currently best at classifying. The second level takes the input subset and re-classifies it to correctly recognise the data rows in each category. For the classifier's own category, this is probably almost the same as for the first level. For any categories it incorrectly classified, there are new classifiers in the layer to reclassify those rows. One thing to note might be the following: the classification for the incorrectly selected rows in the first level needs to be more accurate in the second level, and so the classifier needs to continue to classify more accurately each level, but the dataset size would continue to reduce.

Using the classifier is then automatic. When passed a data row for classification, each classifier in the first level produces an error with respect to its own category. Select the classifier with the smallest error. If it has a second level, then pass the second level the data row. For its own category, the second level is essentially the same, but it can produce a different result for the other category groups. Repeatedly select the classifier with the smallest error and if it has a next level go to that. If it does not, then the category of the current classifier is the final result. The schematic of Figure 1 shows the classification process, where a new layer has been added to a classifier that has incorrectly classified part of a different category group.

4 Proof

With this branching technique, there is also a filtering out of the dataset rows to consider at each layer. When training, if the classifier moves to its next layer, it only needs to consider the dataset rows related with its current layer. So, when these are adjusted for incorrect categories, the consideration is for that subset only. This is a fairly basic argument for why the classifier should work. In the next layer therefore, as there is a smaller number of rows, already filtered to a particular averaged value, it should be able to classify them more accurately, because there should be less variability in them.

A new layer would use a whole new group of classifiers on the subset of data and needs to be more concerned about minor adjustments to a shape than the vertical adjustment. The most extreme case would be to split the dataset so that there is a classifier for every single data row, when it would be able to classify that with 100% accuracy. But there would then be a question about generalising. As the branching process reduces the global dataset at each stage, it is also quite stable. One small problem seems to be the fact that a set of equal (usually 0) error results can occur, when the category then has to be a choice between any of those results, but this does not happen too often.

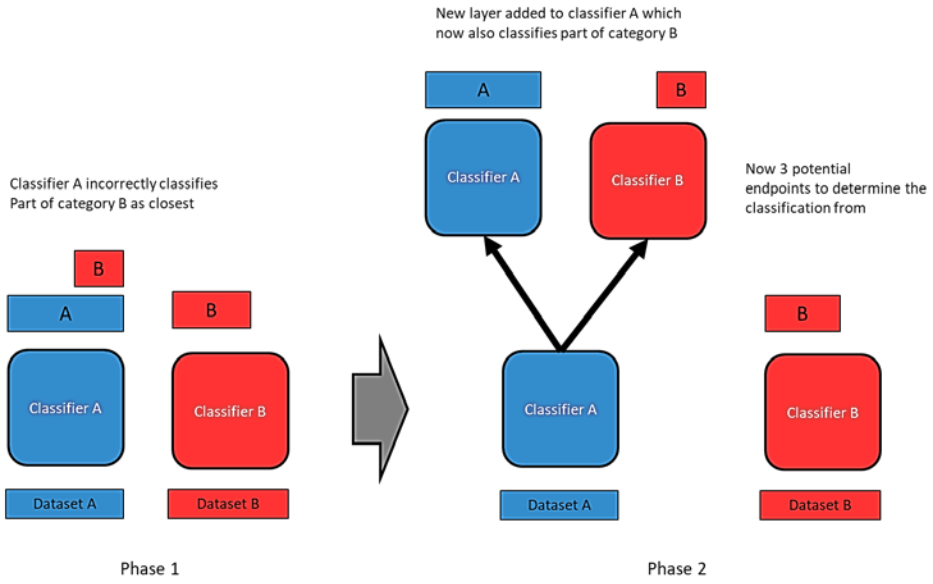

Figure 1. Schematic of the classifier in action. Phase 1 realises that classifier A also classifies part of category B better. Phase 2 then adds a new layer to layer 1 of classifier A, to correctly re-classify this problem.

5 Test Results

This paper repeats the set of tests carried out in the first paper [8], to verify that the classifier can produce a high level of accuracy. Please see that paper for a more detailed discussion about the original and other researchers results. A test program has been written in the C# .Net language. It can read in a data file, normalise it, generate the classifier from it and measure how many categories it subsequently evaluates correctly. Each output category is now represented by a separate classifier. As each category is represented separately, there are two choices to what value the category should be given. It still seems preferable to give each output category a different value, but not in every case. Therefore, 3 categories would result in desired output values of 0, 0.5 and 1, even if in 3 separate classifiers. With separate classifiers, another option is to centre the category output on the value 0.5 (in the range 0 to 1). This is because the data row values can then correct to this value from above or below, by an equal amount. As this is not necessarily better, the adjustment or error correction to the output category value must have some type of distinct effect as well, or it may reflect the input averaged value in some way.

As a side note, an earlier wave shape paper [10] suggested that a vertical adjustment could be helpful in realigning the input data values to the output ones. This vertical adjustment is actually implicit in the oscillating error adjustment. It is still possible to vertically adjust after the error corrections, but it becomes negligibly small after the final error correction.

Two types of result were measured. The first was an average error for each row in the dataset, after the classifier was trained, calculated as the average difference between actual output and the desired output value. The second measurement was how many categories were correctly classified. For these tests, an error margin is not considered, but the errors of each output category are compared and the smallest one is selected.

5.1 Benchmark Datasets with Train Versions Only

The classifier was first tested on 3 datasets from the UCI Machine Learning Repository [18]. These are the Wine Recognition database [5], Iris Plants database [4] and the Zoo database [19]. Wine Recognition and Iris Plants have 3 categories, while the Zoo database has 7. These do not have a separate training dataset and are benchmark tests for classifiers. A stopping criterion of 10 iterations was used to terminate the tests. For the Wine dataset, the

UCI [18] web page states that the classes are separable, but only RDA [7] has achieved 100% correct classification. Other classifiers achieved: RDA 100%, QDA 99.4%, LDA 98.9%, 1NN 96.1% (z-transformed data) and all results used the leave-one-out technique. So that is the current state-of-the-art. Three other datasets were also tested. These were the Abalone shellfish dataset [2], the Hayes-Roth concept learning dataset [12] and the BUPA Liver dataset [15]. As shown by Table 1, for most of the datasets, the new classifier can classify to the accuracy required by these benchmark tests. The final column 'Selected Best %' lists the best results found by some other researchers.

5.2 Separate Train and Test Datasets

To better test the generalising properties of a classifier, a previously unseen test dataset can be used. Four tests that have separate train and test datasets were tried and the results, given in Table 2, are again favourable. A stopping criterion of 10 iterations was used to terminate the tests. Two of the datasets are User Modelling [13] and Bank Notes [16]. A third dataset was a heart classifier from SPECT images [14] and a fourth dataset was a letter recognition task [6]. The new version of the classifier has really only failed on the letter recognition, while the Zoo database was also worse. Both of these fared better when centring the desired output value for each separate classifier, as did the user modelling.

Dataset	Average Error	Correctly Classified	% Correct	Selected Best %
Wine	0.03	178 from 178	100%	100%
Iris	0.05	150 from 150	100%	97%
Zoo	0.02	85 from 101	84%	94.5%
Abalone	0.002	4093 from 4177	98%	73%
Hayes-Roth	0.09	130 from 132	99%	50%
Liver	0.04	345 from 345	100%	74%

Table 1. Classifier Test results. Average output error and number of correct classifications. All datasets points normalised to be in the range 0 to 1.

Dataset	Average Error	Correctly Classified	% Correct	Selected Best %
UM	0.05	138 from 145	95%	97.9%
Bank notes	0.05	100 from 100	100%	61%
Heart	0.08	187 from 187	100%	84%
Letters	0.009	1207 from 4000	30%	82%

Table 2. Classifier Test results. The same criteria as for Table 1, but a separate test dataset to the train dataset.

6 Conclusion

This paper has extended the work reported in [8] and corrected the error in that paper, to show that the classifier can work to a very high standard. In this paper, separate classifiers are used and trained on each category batch and so the process would branch through different classifier sets to find each result. This paper also gives a very basic proof for why the classifier should work. The test results are almost as good as the unbeatable results of the first paper, with probably only 1 failure from that earlier test set. Are there some new best results however? The classifier is almost instantaneous. There is no complicated setup and it will always produce the same result, so this should make it an attractive option for some cases. These tests have also helped to show that a classifier selects some averaged value to represent a category. It does not select any of the

data rows exactly but a distinct value somewhere in the range space of all of the data points. When a new data row is presented, it is the adjustment to this averaged value that determines what category it belongs to. In fact, generalisation could be a weakness of the method, or overfitting, because it does re-classify on smaller and smaller subsets of the data. For the letters dataset, for example, it did correctly classify 15585 of the 16000 train examples, with an average error of 0.002, but with 1038 counts coming from equal classifications. This was by far the largest equal classifications count, with other datasets producing a 0 count there. Branching appears to be the only option at the moment. Other ideas did not work as well, but this research does offer an interesting alternative to the more commonly applied approaches.

References:

[1] Akkar, H.A. and Mahdi, F.R., 2016. Evolutionary algorithms for neural networks binary and real data classification. Int J Sci Technol Res, 5(7), pp.55-60.

[2] Asim, A., Li, Y., Xie, Y. and Zhu, Y. (2002). Data Mining For Abalone, *Computer Science 4TF3 Project*, Supervised by Dr. Jiming Peng, Department of Computing and Software, McMaster University, Hamilton, Ontario.

[3] Dershowitz, N. and Falkovich, E. (2015). Cellular Automata are Generic, *U. Dal Lago and R. Harmer (Eds.): Developments in Computational Models 2014 (DCM 2014). EPTCS 179*, pp. 17-32, doi:10.4204/EPTCS. 179.2.

[4] Fisher, R.A. (1936). The use of multiple measurements in taxonomic problems, *Annual Eugenics, 7, Part II*, pp. 179-188, also in *Contributions to Mathematical Statistics* (John Wiley, NY, 1950).

[5] Forina, M. et al. (1991). PARVUS - An Extendible Package for Data Exploration, Classification and Correlation. *Institute of Pharmaceutical and Food Analysis and Technologies, Via Brigata Salerno*, 16147 Genoa, Italy.

[6] Frey, P.W. and Slate, D.J. (1991). Letter recognition using Holland-style adaptive classifiers, *Machine learning*, Vol. 6, No. 2, pp. 161-182.

[7] Friedman, J.H. (1989). Regularized Discriminant Analysis, *Journal of the American Statistical Association*, Vol. 84, No. 405, pp. 165-175.

[8] Greer, K. (2017). A New Oscillating-Error Technique for Classifiers, *Cogent Engineering, Taylor and Francis Online*, https://doi.org/10. 1080/23311916.2017.1293480. Also available on arXiv at http://arxiv.org/abs/ 1505.05312.

[9] Greer, K. (2015). A Single-Pass Classifier for Categorical Data, *Special Issue on: IJCSysE Recent Advances in Evolutionary and Natural Computing Practice and Applications, Int. J. Computational Systems Engineering, Inderscience*, Vol. 3, Nos. 1/2, pp. 27 - 34. Also available on arXiv at http://arxiv.org/abs/ 1503.02521.

[10] Greer, K. (2013). Artificial Neuron Modelling Based on Wave Shape, *BRAIN. Broad Research in Artificial Intelligence and Neuroscience*, Vol. 4, Issues 1-4, pp. 20-25, ISSN 2067-3957 (online), ISSN 2068-0473 (print).

[11] Gurevich, Y. (2000). Sequential Abstract State Machines Capture Sequential Algorithms, *ACM Transactions on Computational Logic* Vol. 1, No. 1, pp. 77 – 111, doi:10.1145/ 343369.343384. Available at http://research. microsoft.com/~gurevich/opera/141.pdf.

[12] Hayes-Roth, B. and Hayes-Roth, F. (1977). Concept Learning and the Recognition and Classification of Exemplars, *Journal of Verbal Learning and Verbal Behavior*, Vol. 16, No. 3, pp. 321-338.

[13] Kahraman, H.T., Sagiroglu, S. and Colak, I. (2013). The development of intuitive knowledge classifier and the modeling of domain dependent data, *Knowledge-Based Systems*, Vol. 37, pp. 283-295.

[14] Kurgan, L.A., Cios, K.J., Tadeusiewicz, R., Ogiela, M. and Goodenday, L.S. (2001). Knowledge Discovery Approach to Automated Cardiac SPECT Diagnosis, *Artificial Intelligence in Medicine*, Vol. 23, No. 2, pp 149-169.

[15] Liver dataset (2016). https://archive.ics.uci. edu/ml/datasets/Liver+Disorders.

[16] Lohweg, V., Dörksen, H., Hoffmann, J. L., Hildebrand, R., Gillich, E., Schaede, J., and Hofmann, J. (2013). Banknote authentication with mobile devices. *In IS&T/SPIE Electronic Imaging (pp. 866507-866507). International Society for Optics and Photonics*.

[17] Potter, M.A. and De Jong, K.A., 2000. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary computation*, Vol. 8, No. 1, pp.1-29.

[18] UCI Machine Learning Repository (2016). *http://archive.ics.uci.edu/ml/*.

[19] Zoo database (2016). *https://archive.ics.uci.edu/ml/datasets/Zoo*.