An Experiment with Bands and Dimensions in Classifiers

Kieran Greer, Distributed Computing Systems, Belfast, UK.
kgreer@distributedcomputingsystems.co.uk.
Version 1.0

Abstract – This paper presents a new version of an oscillating error classifier that has added fixed value ranges through bands, for each column or feature of the input dataset. It is shown that some of the data can be correctly classified through using fixed value ranges only, while the rest can be classified by using the classifier technique. It also presents the classifier in terms of a biological model of neurons and neuron links.

Keywords: classifier, oscillating error, neural network, multiple layers, bands and branching.

1 Introduction

This paper presents a new version of an oscillating error classifier that has added fixed value ranges through bands, for each column or feature of the input dataset. An earlier version of the classifier added branches [8] to a categorical classification technique that allows the error update to be independent for each column value and can therefore oscillate around the desired output, reducing to some minimum. Because that classifier works off averaged values, it may be the case that some data can be classified directly, without it having to be sorted by weight sets, for example. The averaged value is simply 1 value for a whole range of actual input values and so maybe a value band can represent that range as a fixed set of boundaries. It may also be possible to construct these fixed boundaries for single dimensions, when much more complex hypercubes are not required. It is shown that some of the data can in fact be correctly classified through using fixed value ranges only, while the rest can be classified by using the classifiers. With the idea of these fixed bands that do not process very much, plus the more complex classifiers, the paper also presents the whole system in terms of a biological model of neurons and neuron links.

The rest of this paper is organised as follows: section 2 introduces some related work. Section 3 reviews the earlier work on the classifier, while section 4 describes the new method that uses fixed bands as well as branching. Section 5 re-runs the test set to verify the classifier's accuracy, while section 6 compares the new classifier with biology. Finally, section 7 gives some conclusions to the work.

2 Related Work

This research is based specifically on two earlier papers that introduced the oscillating error classifier [8][9]. The classifier can be used to classify categorical data, or data rows grouped into categories. It uses averaged values for each category and an oscillating error technique that decides whether to add or subtract the error from each cell value, to minimise the overall error. The first paper [9] actually had a mistake in its evaluation, that was corrected by the branching mechanism suggested in [8]. The idea of batch processing or averaged values is not new and was used in some of the earlier neural network models [7] [11]. The research of this paper also considers classifying each data column, or dimension, separately. This has been looked at previously, usually in relation to kNN classifiers [2]. This is not typically associated with a neural network, but with the oscillating error classifier, the column weights are learned separately, before being linked for the evaluation process and so the separate learning process is a bit orthogonal.

The new method may also be of interest with respect to biological models of the human brain, which will be elaborated on later, but papers that note the importance of the neuron links in the brain communication process include [10][18].

3 Oscillating Error Classifier Review

The classifier can be used to classify categorical data, or data rows that are classified into categories. Essentially, the classifier works off averaged numerical values, not specific values for each data row. It creates a classifier for each output category and groups all of the input data that belongs to each category. The classifier for that category then learns to adjust its

input, which is the averaged data row, to the desired output, which can even be set to 1 or 0.5, if there are separate classifiers. The premise for this is the fact that there can only be one weight value for all of the input values and so learning the averaged value looks reasonable. The oscillating error method adds or subtracts the difference between the actual and the desired value from each data column value separately. So, for example, the difference could be subtracted from the averaged value in column 1, but added to the averaged value in column 2. This is repeated until a minimum error value is reached or a maximum number of iterations is reached. The data is also normalised, to be in the range 0 to 1. This process is still not accurate enough, where lots of data rows will be incorrectly classified, with a classifier for a different category. If a classifier is associated with data rows from more than 1 category, the classifier can therefore add a new level with new classifiers for the said categories. This new level however should have a less complicated problem to solve, because it will be passed only a subset of the whole dataset to learn. When using the classifier then: if it has branches, they are always passed the input data and asked to return their evaluation. If there is only 1 classifier with no branches, it returns its own result. Therefore, each base classifier will return some category evaluation and error result and the classifier with the smallest error is selected as the best match. Logically this can lead to a classifier learning a single data row and might be expected to be 100% accurate, but that is not the case. For one thing, the data would need to be clearly linearly separable for the branching to work in that way. Nested regions or overlaps in the data values would probably lead to some confusion. This paper tries to address that issue and offers a new alternative method. The new method also leads to some interesting biology-related theories.

4 Extending the Classifier with Fixed Bands and Branching

If the classifier deals with averaged values, then it does not try to learn too much outside of that and so one question might be if ranges of input values can be classified directly. It would certainly be the case if it was linearly separable, because the separating line would allow a clear distinction to be determined. The branching still has a problem however if one part of a classification is maybe contained inside of another classification. Then the averaged values may become confused. If trying to separate the input data then, multi-

dimensional hypercubes would be the first choice, but this looks very difficult when trying to separate overlapping regions. Therefore, another option might be to try to separate each dimension, feature, or data column individually and the rest of the paper demonstrates that this can be quite successful. The data can be read, a column at a time, as it is organised in the data file. The category that each row belongs to is also retrieved and if there is a change in category, the previous set of values can be placed into a band. The only problem is when categories overlap in a single dimension, which would mean that they have the same value, when the band then continues to the next value and category change. For example, consider the following values for a column and related categories, shown in Figure 1.

Column Value	Category
0.1	A
0.2	A
0.3	A
0.4	B
0.5	B
0.5	C
0.6	C

Figure 1. Example of Data column values with related categories, placed into bands: Band 1 - 0.1 to 0.3 and Band2 - 0.4 to 0.6.

The program would therefore firstly sort the data column values into order. It would then read down the column until there is a change in the category. In this case, the first change is at the value 0.4. The value range 0.1 to 0.3 all belongs to a single category and so a band can be made from that. Then process continues and the next break would be at the value 0.5, but there is an overlap with this value as both categories B and C use it. Therefore, the process must continue to the end value, when both of those categories are placed in a single value range of 0.4 to 0.6.

The process is repeated for the next column of values that produces another set of bands. It is also important to link the bands from one dimension to the next depending on the exact values in each data row. So for example, if there are 3 columns in a dataset and each column has 5 bands; then if a data row relates to bands 1, 2 and 4, these bands will have links added between them. Then the band 1 relating to column or dimension 1 can only move to band 2 in the next dimension, and so on. Figure 2 shows the bands and links created for the Iris Plants dataset [3].

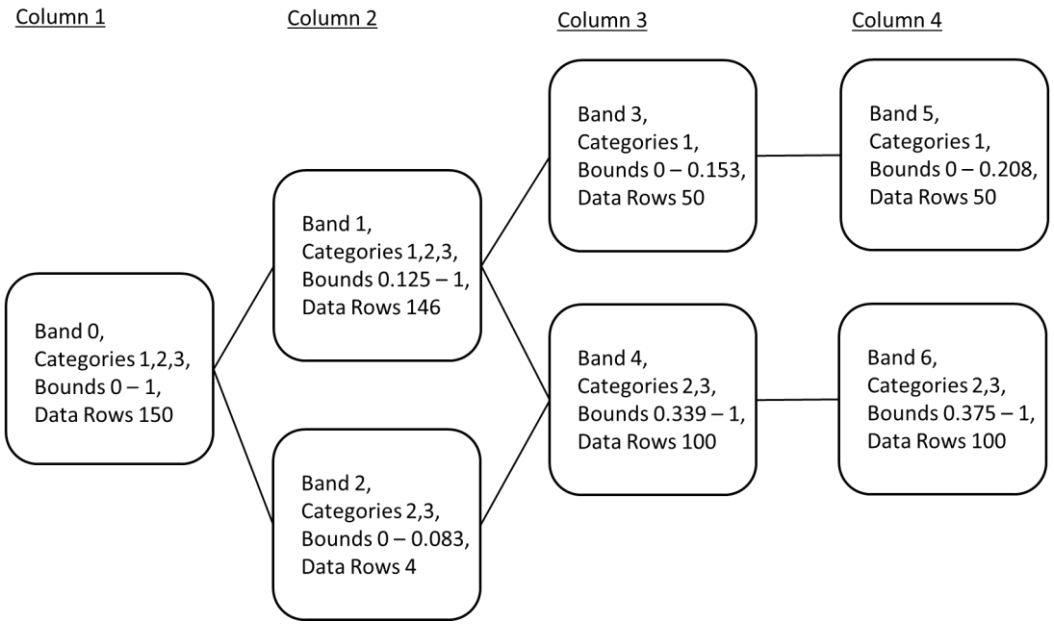

Figure 2. Bands created for the Iris Plants dataset [3].

When presented with an actual data row to evaluate, the procedure traces through the band links, to check if any represent a single category only. If that is the case, then the fixed band ranges can be used to classify the input data directly. Because of the overlap however, there are lots of cases where a band represents more than 1 category. It would be interesting to train the classifiers for those cases only, but for a first test, the classifier system with branching was also trained and used if the bands did not return a result. The bands can help a bit here however, because if they can classify and data rows directly, those rows do not need to be considered by the classifiers and so this was also checked for and

those data rows removed from the training dataset. The bands can make a significant contribution to the classification process. For example, Figure 2 shows bands, where category 1 is linearly separable and can be identified completely from using the band ranges. Categories 2 and 3 would need a classifier to be separated. Also, for category 1, the classification is clear from column 3 and so column 4 is probably not required. In effect, column 3 provides a unique feature for category 1.

5 Test Results

This paper repeats the set of tests carried out in the earlier paper [9], to verify that the classifier can produce a high level of accuracy. Please see that paper for a more detailed discussion about the original design 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, where the classifier typically tries to match with an output value of 0.5. 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.

5.1.1 Benchmark Datasets with Train Versions Only

The classifier was first tested on 3 datasets from the UCI Machine Learning Repository [20]. These are the Wine Recognition database [4], Iris Plants database [3] and the Zoo database [21]. 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 [20] web page states that the classes are separable, but only RDA [6] 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 [1], the Hayes-Roth concept learning dataset [12] and the BUPA Liver dataset [16]. 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.

Dataset	Average Error	Correctly Classified	% Correct	Selected Best %
Wine	0.003	131 from 131	100	100
Iris	0.03	150 from 150	100	97
Zoo	0.005	96 from 101	96	94.5
Abalone	0.001	4165 from 4177	99	73
Hayes-Roth	0	132 from 132	100	50
Liver	0.03	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.

5.1.2 Separate Train and Test Datasets

If some of the data is classified directly in bands, then there is no real generalisation and each classifier is used for a smaller more specific set of data rows. A better test for the classifier would therefore be to have a different test dataset, and the results of this are shown in Table 2. The training time would be instantaneous for something like the Iris Plants dataset, but for the letter recognition dataset in this section, it took 1-2 days with the current version of the software. Each row in the dataset needs to be repeatedly re-tested to determine if branching is required, etc., but the system is very low on resource usage. The process is also completely deterministic and should give exactly the same result each time, and there is no real fine tuning required either.

Six tests that have separate train and test datasets were tried and the results, given in Table 2, are again favourable. The datasets are User Modelling [13], Bank Notes [17], SPECT images heart classification [14], Letter recognition [5], the first Monks dataset [19] and Solar flares [15]. The desired output value for each classifier was again centred, or set to 0.5 for every test. Both the earlier version with branching only [8] and this version classified the

first Monks dataset with 100% accuracy. This version with the bands classified the Solar dataset slightly better than the earlier version and both failed with the Letters dataset. The version with branching only however, was better at both the User Modelling and the Bank Notes, so there might still be a question about the generalising properties of the bands, but probably not too serious.

Dataset	Average Error	Correctly Classified	% Correct	Selected Best %
UM	0.05	126 from 145	87	98
Bank notes	0.004	85 from 100	85	61
Heart	0.1	187 from 187	100	84
Letters	0.007	1238 from 4000	31	82
Monks-1	0.11	432 from 432	100	100
Solar	0.01	984 from 1066	92.5	84

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

5.2 Test Conclusions

The new version that includes bands is certainly worth considering, even if the earlier version can produce better results in some cases. Bother versions are very easy to train and use. The size of the whole classifier structure is quite large, but then the level of functionality for each unit is reduced. For example, for the Abalone dataset, 2700 classifiers were created and the letters dataset produced 12600 classifiers, although some of that would be branching to the next level. In real time, it might then have to test the input across that number of classifiers as well, which is quite a lot for a relatively small dataset. Ideally, new data would be incorporated into the existing classifiers instead of creating new ones. If the dataset is representative, then this might happen up to some type of limit, which could be interesting. While using bands should reduce the number of data rows that a classifier needs to learn, it typically resulted in more classifiers a well, and so there may be a coherence factor over the dataset that also determines the number of classifiers. So, each classifier solves only a small part of the larger problem, but it is not the case that each classifier has been given a few rows of data to classify. The system has generated the classifiers and paths through them for itself. It should also be possible to update the system,

as the classifiers can be updated dynamically and a band can have a boundary value changed quite easily.

6 Biological Comparison

There is biological evidence that the links between neurons (synapse/dendrite) play an important role in the actual signal interpretation and understanding in the human brain [18]. While the neuron makes some processing decisions, the length and firing sequences in these branch structures are also important. The bands would be an analogy to the neuron links. If the signal fits inside of the band boundaries, it can be classified as whatever the band represents without any further processing. If there is any discrepancy, then a classifier is required to sort that out and so this is analogous to a neuron being created to process a more mixed signal. If the neuron behaves like a filter, then the process might be to convert the mixed signal back into more singular parts again.

7 Conclusions

This paper has extended the work reported in [9], by introducing the idea of fixed bands. These bands can be used to classify some input data directly, simply by using value ranges. The bands and the classifiers are currently trained separately and are not linked up, but the bands can remove some data rows for the training of the classifiers. There can be a lot of separate parts now and even some that represent just 1 data row and so the generalisation properties of the classifier might be a problem. However, a dynamic learning phase can help with this and may continually update boundaries until it becomes more stable.

Comparing the classifier with something like PCA, feature selection and biological systems can lead to some logical conclusions. If a data object has a unique feature (data column) then a data band can use that to classify it directly. Maybe in effect, it could be passed down a unique channel relating to that band. If the data object does not have a unique feature, then the current system takes all of the features together and compares that with an averaged value of the features present in each category. The data object is then allocated

the category that it is a closest match to. The comparison with neurons and links between them is clear. Of course, there are still many problems, such as when the classifier fails, or the best way to process a data object using subsets of features.

References

[1] 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.

[2] Bay, S.D. (1999). Nearest Neighbor Classification from Multiple Feature Subsets, Intelligent data analysis, Vol. 3, No. 3, pp. 191-209.

[3] 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).

[4] 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.

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

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

[7] Gallant, S.I. (1990). Perceptron-Based Learning Algorithms, IEEE Transactions on Neural Networks, Vol. 1, No. 2.

[8] Greer, K. (2017). An Improved Oscillating-Error Classifier with Branching, available at arXiv at https://arxiv.org/abs/1711.07042.

[9] 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.

[10] Greer, K. (2015). New Ideas for Brain Modelling, BRAIN. Broad Research in Artificial Intelligence and Neuroscience, Volume 6, Issues 3-4, pp 26 - 46, December 2015, ISSN

2067-3957 (online), ISSN 2068 - 0473 (print). Also available on arXiv at http://arxiv.org/abs/1403.1080.

[11] Hagan, M.T. and Menhaj, M.B. (1994). Training Feedforward Networks with the Marquardt Algorithm, IEEE Transactions on Neural Networks, Vol. 5, No. 6, pp. 989-993.

[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] Li, J., Dong, G. and Ramamohanarao, K. (2000). Instance-based classification by emerging patterns. In European Conference on Principles of Data Mining and Knowledge Discovery, pp. 191 – 200, Springer, Berlin, Heidelberg.

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

[17] 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.

[18] Shira Sardi1, S., Vardi1, R., Sheinin, A., Goldental1, A. and Kanter1, I. (2017). New Types of Experiments Reveal that a Neuron Functions as Multiple Independent Threshold Units, Nature Scientific Reports, 7:18036, DOI:10.1038/s41598-017-18363-1.

[19] Thrun, S.B., Bala, J., E. Bloedorn, I. Bratko, B. Cestnik, J. Cheng, K. De Jong, S. Dzeroski, S.E. Fahlman, D. Fisher, R. Hamann, K. Kaufman, S. Keller, I. Kononenko, J. Kreuziger, R.S. Michalski, T. Mitchell, P. Pachowicz, Y. Reich H. Vafaie, W. Van de Welde, W. Wenzel, J. Wnek, and J. Zhang has been published as Technical Report CS-CMU-91-197, Carnegie Mellon University in Dec. 1991.

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

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