Classification for accuracy and insight: A weighted sum approach

Anthony Quinn **Andrew Stranieri** **John Yearwood**

School of Information Technology and Mathematical Sciences
University of Ballarat,
Gear Ave, Ballarat, Victoria 3350,
Email: quinn@clearmail.com.au

Abstract

This research presents a classifier that aims to provide insight into a dataset in addition to achieving classification accuracies comparable to other algorithms. The classifier called, **A**utomated **W**eighted **Sum** (AWSum) uses a weighted sum approach where feature values are assigned weights that are summed and compared to a threshold in order to classify an example. Though naive, this approach is scalable, achieves accurate classifications on standard datasets and also provides a degree of insight. By insight we mean that the technique provides an appreciation of the influence a feature value has on class values, relative to each other. AWSum provides a focus on the feature value space that allows the technique to identify feature values and combinations of feature values that are sensitive and important for a classification. This is particularly useful in fields such as medicine where this sort of micro-focus and understanding is critical in classification.

Keywords: data mining, insight, conditional probability.

1 Introduction

Many classifiers provide a high level of classification accuracy, yet their use in real world problems is limited because they provide little insight into the data. The classifier presented in this research, **A**utomated **W**eighted **Sum** (AWSum), provides a degree of insight into the data whist maintaining accuracy that is comparable with other classifiers.

By insight we mean that the technique provides an analyst with an appreciation of the influence that a feature value has on the class value. For example it is intuitive to ask the question: what influence does high blood pressure have on the prospects of having a heart disease? Or, does smoking suggest heart disease more than it suggests a lack of heart disease? A classifier that can provide simple to grasp answers to these sorts of questions could be expected to provide a degree of insight and be useful in real world data mining, particularly if its classification accuracy is comparable to other techniques.

Probabilistic approaches, such as Naive Bayes (Duda and Hart 1973) rely largely on maximising the probability that an example belongs to a given class and only indirectly provide any indication of the influence the feature values have on the classification. Connectionist approaches such as neural networks offer little or no direct insight although some attempts at deriving meaning from internal connection weights have been made (Setiono and Liu 1996). Geometric approaches such as Support Vector Machines (SVM) (Vapnik 1999) clearly identify the feature values in the support vectors as being the most important but it is difficult to generalise from this. Rule and tree based approaches provide some insight, though features are not certain to appear in the rules, or trees, even if they are influential to classification.

A further advantage provided by AWSum, that can be useful in real world data mining situations, is an assessment of the confidence of a classification. For example a forward feed neural network trained with back propagation can indicate that an example belongs to a given class but not whether this is a strong assertion or a weak assertion. The ability to assess the confidence of a classification is important in many diverse real world situations. In the medical field we may chose to medicate a patient if we are only reasonably sure of a cancer diagnosis but operate when we are very sure of the diagnosis. In a political scenario we may choose not to direct campaign time to those voters we are very confident will vote for us but dedicate resources to those voters that we are only mildly confident will votes for us.

AWSum focuses at the feature value level in order to identify the feature values and combinations of feature values that are sensitive and important to a classification. This is useful in fields such as medicine where a micro-focus on the influences on classification and an understanding of the data is critical. Other techniques such as trees and probabilistic approaches consider the importance of the values of a feature as a group. A simple example of this can be found in the Cleveland Heart dataset. If the values of the feature *age* are considered as a group, the relationship between *age* and *heart disease* identified would be that as *age* increases so does *heart disease* as seen in figure 1. This fails to identify the reversal of trend as we tend toward the extreme of *age*.

AWSum assesses the contribution of each feature value to the classification individually by assigning it a weight that indicates its influence on the class value. A weighted sum approach is taken, combining these influence weights into an influence score for the example. Figure 2 shows the weights AWSum has assigned to each feature value on a scale. The class values are placed at the extremes of the scale, -1 and 1. These extremes represent the points at which the probability of the relevant class outcome is 1. The influence weight of -0.03 assigned to *age 50* indicates that this value of *age* influences an outcome of *heart disease = yes* approximately the same amount of times as it influences an outcome of *heart disease = no*. The ratio of the occurrence of *heart disease = yes* to *heart disease = no* strengthens in favour of the class value represented at the extreme as it nears that extreme. The reversal of the influence of *age* on *heart disease* can now be readily identified.

The intuition behind AWSum's approach is that each feature value has an influence on the classification that can be represented as a weight and that combining these influence weights gives an influence score for an example. This score can then be compared to a threshold in order to classify the example.

The algorithm for calculating and combining weights, and determining thresholds is explained in section 2.

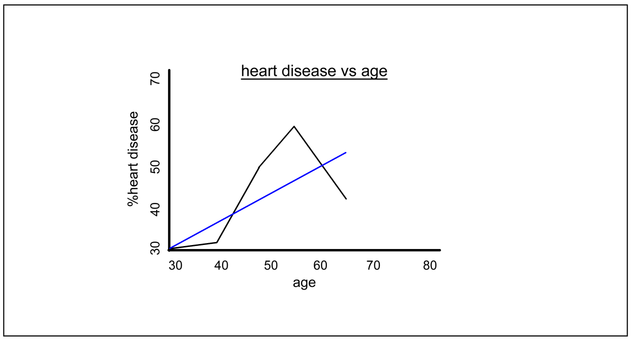

Figure 1: Feature level focus

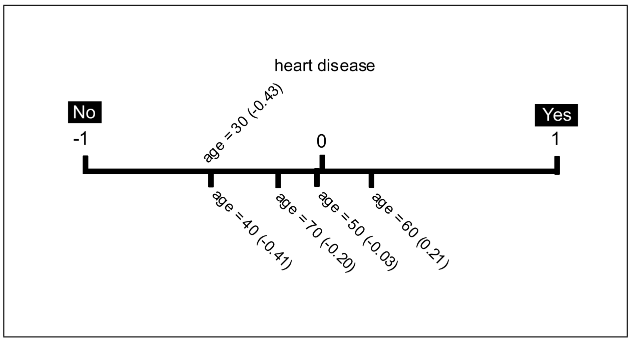

Figure 2: AWSum feature value focus

2 The Algorithm

The algorithm can be split into two major parts described separately below;

- Influence - Influence weights are established for each feature value that give a measure of the feature value's influence on the outcome and threshold/s are calculated
- Classification - New examples are classified by calculating an influence score for the example from the influence weights of the component feature values. This can be seen as a combination of evidence for the classification.

2.1 Influence

The first phase of the AWSum approach lays the foundations for classification and provides insight into the dataset by providing an influence weight for each feature value. For simplicity we will only consider binary classification tasks. Higher order tasks will be discussed later in section 5. For any feature value the sum of the conditional probabilities for each possible class value is 1 as the events are mutually exclusive, as illustrated for a binary outcome in equation 1.

$$Pr(O_1|F_v) + Pr(O_2|F_v) = 1 \tag{1}$$

Where: O_1 and O_2 are the first and second value on the class feature. F_v is the feature value.

A feature value's influence weight, W represents its influence on both class values and so it needs to simultaneously represent both probabilities from equation 1. To do this we arbitrarily consider one class outcome to be positive, and map probabilities to a range of 0 to +1. The other class is considered to be negative and map probabilities to a range of 0 to -1. The range of mapped probabilities for both feature values is therefore -1 to +1. By summing the two mapped probabilities we arrive at a single influence weight that represents the feature value's influence on both class values. Equation 2 demonstrates this calculation and figure 3 shows an example where $Pr\left(O_1|Fv_1\right) = 0.2$, or -0.2 when mapped and $Pr\left(O_2|Fv\right) = 0.8$.

$$W = Pr\left(O_1|Fv\right) + Pr\left(O_2|Fv\right) \tag{2}$$

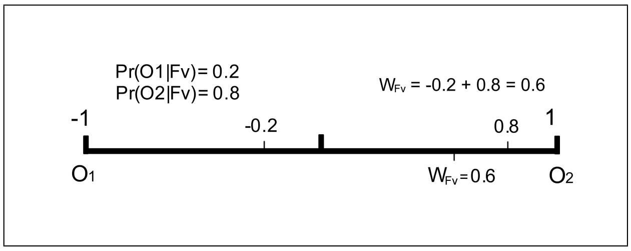

Figure 3: Binary class example

Additional assumptions are required to be made in the case of class features that are ternary or of a higher order. This is discussed below in Section 5.

2.2 Classification

Classification of an example is achieved by combining the influences from each of the example's feature values into a single score. By summing and averaging feature value influences we are able to arrive at a score that represents the evidence that the example belongs to one class and not to another. Equation 3 depicts this. Performing the combination by summing and averaging assumes each feature value influence is equally comparable. Although this is a relatively naive approach, it is quite robust as described later in this section.

$$e_1 = \frac{1}{n}\sum_{m=1}^{n} W_m \tag{3}$$

e_1 = the influence weight of the i^{th} example
n = the number of examples

The influence score for an example is compared to threshold values that divide the influence range into as many segments as there are class values. For instance, a single threshold value is required for a binary classification problem so that examples with an influence score above the threshold are classified as one class value, and those with a score below the threshold are classified as the other class value. Each threshold value is calculated from the training set by ordering the examples by their weight and deploying a search algorithm based on minimising the number of incorrect classifications. This is a simple linear optimisation problem that is solved by calculating the misclassification rate at each point along the scale. For instance, the examples with total influence scores that fall to the left of the threshold in Figure 4 are classified as class outcome A by AWSum. This however includes two examples that belong to class B in the training set so these two examples are misclassified. Two examples to the right of the threshold are misclassified as class B when they are A*'s*. In cases where there are equal numbers of correctly and incorrectly classified examples the threshold is placed at the mid-point under the assumption that misclassification of class A and B is equally detrimental.

New examples can be classified by comparing the example's influence score to the thresholds. The example belongs to the class in which its influence score falls.

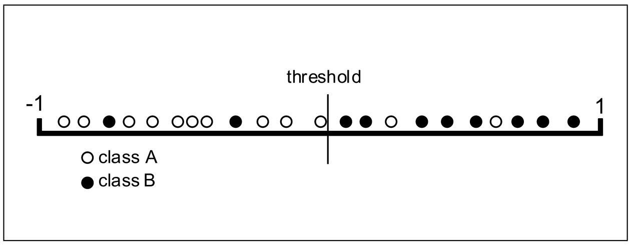

Figure 4: Threshold optimisation

AWSum is suited to nominal feature values and class outcomes although it is not necessary that they are ordinal. Continuous numeric features require discretisation before use in AWSum.

Classification accuracy of the AWSum approach compares favourably with that of many other algorithms. Experimental results are presented in section 4.

3 Extending the algorithm

The combining of influence weights for single feature values into a total influence score for an example and using this to classify is intuitively based however, it is plausible that feature values may individually not be strong influences on a class outcome but when they occur together the pair is a strong influence. For example both *drug A* and *drug B* may individually be influential toward low blood pressure but taken together lead to an adverse reaction that results in exceedingly high blood pressure. This sort of insight into a dataset can be very useful, particularly in medical domains.

The influence weights for each feature value pair can be calculated in the same way as they were for the single feature values. Equation 4 shows this calculation.

$$W = Pr\left(O_1|Fv_1, Fv_2\right) + Pr\left(O_2|Fv_1, Fv_2\right) + \quad (4)$$

$$\cdots Pr\left(O_k|Fv_1, Fv_2\right)$$

Where:
W =the influence weight of the pair.
O_1 is the first class value and O_k is the k^{th} class values.
Fv_1 is the 1^{st} feature value of the pair and Fv_2 is the 2^{nd}.

These pairs have the ability to both increase insight because the influences on the outcome are now more granular and increase accuracy. When using a feature value pair in the classifier the corresponding single feature weights are not used in order to avoid double counting the influence of the feature values.

3.1 Model selection

There is a need to select which feature value pairs to include in the classifier. There have been 2 methods employed in testing. The first , used on the UCI Cleveland Heart, Mushroom and Vote datasets is to include each feature value pair into the classifier and retain it if it improves classification accuracy. The second method, used on the Iris dataset was to select a support threshold for the feature value pairs and include all pairs that meet this threshold. The support for a feature value pair weight is a calculation of the number of times the pair occurs divided by the total number of examples.

3.2 Fine tuning

AWSum includes a technique that can be used to emphasise important feature values. A power is applied to the influence weights. This process occurs before the threshold algorithm is applied. Equation 5 shows the new calculation for the example weights. Note that the original sign of the influence weight is kept. This fine tuning technique gives more emphasis to influence weights whose absolute weight is larger.

$$e_1 = \frac{1}{n}\sum_{m=1}^{n} W_m^p \quad (5)$$

e_1 = the weight of the i^{th} example
n = the number of examples
p = the power to which the features values are raised
nb. the original sign of the influence weight is kept

4 Experiments

Four datasets were sourced from the University of California, Irvine's Machine Learning Repository (Blake et el 1988) for the comparative evaluation of the AWSum approach:

- ***Cleveland Heart***- 14 numeric features, 2 classes, 303 instances, 6 missing values
- ***Iris***- 5 numeric, continuous features, 3 classes - 1 linearly inseparable, 150 instances, 0 missing values
- ***Mushroom*** - 22 nominal features, 2 classes, 8124 instances, 2480 missing values
- ***Vote*** - 17 boolean features, 2 classes, 435 instances, 0 missing values

Classification accuracy has been assessed using 10 fold stratified cross validation. Table 1 represents classification accuracy using single influence weights only. The classification accuracy of AWSum on the four UCI datasets is comparable though not better than the Naive Bayes Classifier, TAN, C4.5 and the Support Vector Machine.

Table 1: Classifier comparison

Data	**AWSum**	**NBC**	**TAN**	**C4.5**	**SVM**
Heart	83.14	**84.48**	81.51	78.87	84.16
Iris	94.00	94.00	94.00	96.00	**96.67**
Mush	95.77	95.83	99.82	**100**	**100**
Vote	86.00	90.11	94.25	**96.32**	96.09
Avg	89.72	91.11	92.40	92.80	**94.23**

Table 2 shows the classification accuracies achieved by including influence pairs and they are quite comparable with other approaches. AWSum performs better than the others on the Cleveland Heart dataset and better than Naive Bayes and TAN on the Iris set. The Support Vector Machine outperforms all others on Iris, C4.5 and SVM perform perfectly on the Mushroom dataset and C4.5 outperforms the others on the Vote data.

Table 2: Classifier comparison including influence pairs

Data	AWSum	NBC	TAN	C4.5	SVM
Heart	**85.83**	84.48	81.51	78.87	84.16
Iris	94.67	94.00	94.00	96.00	**96.67**
Mush	99.37	95.83	99.82	**100**	**100**
Vote	95.86	90.11	94.25	**96.32**	96.09
Avg	93.93	91.11	92.40	92.80	**94.23**

Table 3 shows the best results achieved using influence values, influence pairs and the power based fine tuning method discussed in section 3.2. The average classification using 10-fold cross validation over the four sample datasets is slightly higher than the other approaches. The objective of this study was to advance a classifier that demonstrated comparable classification accuracy while providing some degree of insight about influential factors. Results indicate AWSum achieves comparable accuracy.

Table 3: Classifier comparison including influence pairs, fine tuned

Data	AWSum	NBC	TAN	C4.5	SVM
Heart	**87.18**	84.48	81.51	78.87	84.16
Iris	96.00	94.00	94.00	96.00	**96.67**
Mush	99.93	95.83	99.82	**100**	**100**
Vote	**97.01**	90.11	94.25	96.32	96.09
Avg	**95.03**	91.11	92.40	92.80	94.23

4.1 Insight

Insight is provided by identifying the influence that feature values have in classification. This can be important in identifying key features in a problem domain as well as eliminating features that are not important. Being able to represent the influence that feature values have on class values graphically provides a informative description of the problem domain. Figure 5 shows this information for the Cleveland Heart dataset. The figures in braces on the right of the scale are the influence pairs added to the classification model, although all pair weighings are calculated and can be used for insight.

Insight can be drawn from this figure. For example, if a patient does not get exercise induced angina (exang no) this has an influence weight of -0.39 indicating a moderate influence toward no heart disease. Similarly if the number of major vessels coloured by fluoroscopy is 0 (ca 0) there is a moderate influence toward no heart disease with an influence weight of -0.47, but if these two factors occur together there is a strong influence toward no heart disease as indicated by the influence pair weight of -0.72. These types of insights can help confirm an understanding of the problem domain or provide new and interesting paths for investigation.

The pairs included in figure 5 fall into two categories. Influence pairs, *cp - Typical angina and slope - down* and *cp - typical angina and thal - fixed defect* are examples of rare cases. They appear in the dataset 1% of the time but always lead to the same outcome. It can be important, particularly in a field such as medicine to be able to identify rare cases. Most techniques fail to identify rare cases because they are concentrating at a feature level. For example a rare case may not be include in a tree based classifier if collectively the values of the feature don't split the data well. Likewise an important dependency may not be modeled in an augmented Bayesian approach if collectively the values of the features are not important. The other three pairs in Figure 5 occur frequently and indicate interesting relationships between the feature values in the pair. This interest occurs because their influence as a pair is markedly different to their influence as single values. Their inclusion both increases accuracy and provides insight. Currently, a heuristic search is deployed to locate pairs of possible

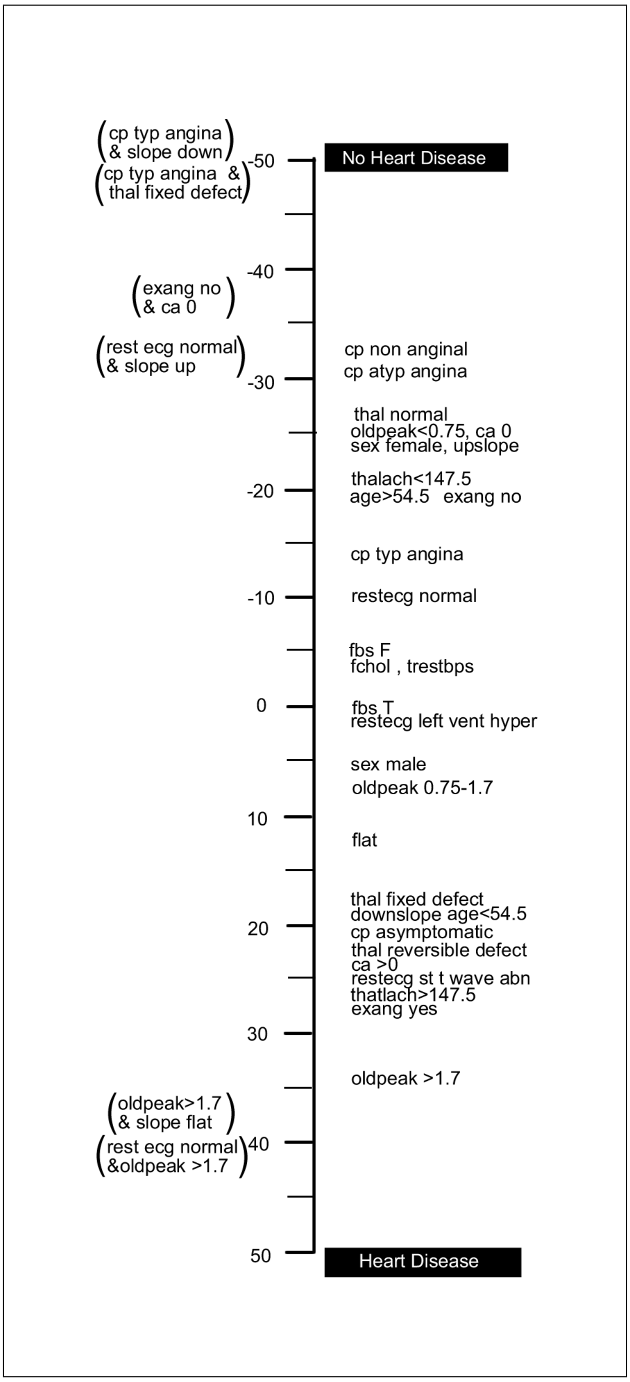

Figure 5: feature value and feature value pair weights

interest. Rare item pairs are considered interesting. Pairs that are not rare are considered interesting if the influence the pair has is markedly different from the average of the influences of each member of the pair. In this way we can identify both rare cases and important relationships that have high levels of support. Work is in progress to apply other search algorithms to enable pairs, triples and higher order pairings that are interesting to be identified.

Tree based classifiers tend to provide more insight than most classifiers and so the insights provided by AWSum are compared to those provided by the implementation of C4.5 (Quinlan 1993) provided by the Weka data mining tool (Witten and Frank 2000). The tree generated selects nodes from the root that are good for splitting the data with regard to the class values. The features closer to the root of the tree could be seen as more important in some senses but this does not convey the relative influence of the feature values in the same way as conveyed by AWSum.

The C4.5 tree generated on the Cleveland Heart dataset uses 9 of the 13 features. Those omitted includes *chol, fbs, trestbps* and *thalach.* It can also be seen that the tree does not necessarily contain all the important or influential features. For instance, *thalach* is identified as important in both AWSum and two feature selection techniques, first best and information gain attribute evaluator (Witten and Frank 2000) yet does not appear in the decision tree. This is understandable because features selected as nodes for a decision tree are those that represent the greatest information gain of the features in contention.

4.2 Discussion

The AWSum approach represents a concentration on feature values that most other techniques do not take. Other techniques tend to consider the values of a feature as a group and identify them as important or otherwise collectively. Probabilistic approaches such as augmented Bayes either relax Naive Bayes' independence assumptions by including dependencies between selected features or they look for independent features. In either case the feature values of a given feature are selected if they are collectively significant Tree based classifiers also focus on features as they search for the best features to split the data on at each node. This again is a collective consideration of the feature values of a given feature. Connectionist approaches such as neural networks include hidden nodes in a pragmatic approach that consumes any concentration on feature values. Geometrical approaches such as SVM (Vapnik 1999) consider a select number of important boundary instances, or support vectors, in order to construct a linear function to separate the class and so are not focusing on identifying the influence of feature values.

In contrast to many other classifiers, AWSum is simple, scalable and easy to implement. Classification processes are easily understood by the non expert and this is often as important as the classification itself. AWSum's use of conditional probability is markedly different to that of Bayesian approaches, such as Naive Bayes. NB compares the probability that the example's feature values were derived by the class outcome, scaled by the prior probability of the class. AWSum, on the other hand, uses conditional probability to calculate a weight that indicates a feature values influence on the class value and combines these influences for each example, comparing the result with a threshold. This style of approach can be seen as a combination of evidence although it is a very different approach to that of the Dempster/Schafer work (Shafer 1976).

The use of pairs or combinations of feature values in AWSum differs from that of probabilistic approaches like Naive Bayes. These style of approaches look for computationally economical ways to model probabilities. Rather than this AWSum is looking for combinations of feature values that may have a strong influence on the class value, and using these as pieces of evidence for a given class outcome.

The addition of pairs involves calculating a weight for each feature value pair in the dataset and so adds to the approach computationally. These calculations can be done in a single pass of the dataset and used in a lookup table to classify. This means that the overhead is not large.

5 Higher dimension class features

In order to represent 3 or more class values on a linear scale certain assumptions need to be made. The class values need to be considered as ordinal. For example if the 3 class outcomes are light, medium and heavy and we have 5 light examples, 0 medium examples and 5 heavy examples we have conditional probabilities of $Pr(light|F_v) = 0.5$, $Pr(medium|F_v) = 0.0$ and $Pr(heavy|F_v) = 0.5$. The feature value, F_v would be assigned a weight of 0 using AWSum which places it in the middle of the influence scale. In terms of conditional probability this is inconsistent as there are no medium examples, but in terms of influence on the outcome it is intuitive because we can reasonably say that the influence of 5 heavy examples and 5 light examples is the same as 10 medium examples. This approach can be demonstrated to classify well even in cases such as the Iris dataset where the outcomes are not ordinal but the visualisation may be misleading in that a value at the middle of the scale could appear there either because there is a high probability of that outcome or because class values at the extremes have the same probability.

For a ternary class outcome, as illustrated in figure 6, the influence value weight can be decided using the the conditional probabilities of the 2 class values represented at the extremes of the scale. Equation 6 illustrates the calculation.

Problems that contain 4 or more class values can simply be seen as combinations of scaled binary outcomes that can be summed to give an influence weight. Figure 7 shows a situation with 4 class values. Each binary feature weight is calculated as per equation 2, with the weight for outcomes 2 and 3 being scaled and summed as per equation 7. This approach to calculating feature value weights can be extrapolated to any number of feature values

$$W = \frac{-Pr\left(O_1|Fv\right) - Pr\left(O_3|Fv\right)}{2} \tag{6}$$

$$W = W_{O_{1,4}} + \frac{1}{3}W_{O_{2,3}} \tag{7}$$

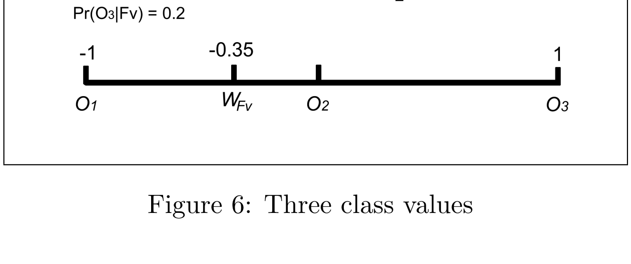

Figure 6: Three class values

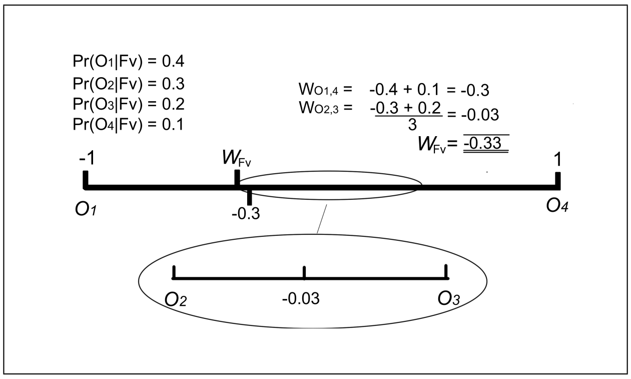

Figure 7: Four class values

6 Conclusion

AWSum demonstrates that classification accuracy can be maintained whist providing insight into the problem domain. This sort of insight can provide important information in itself or be used in preprocessing the data for another approach. It is not intended that AWSum replace traditional approaches but rather that it provides a different and possibly useful resource for analysts to use in approaching real world datasets. It may be that its usefulness is in identifying important features, visualising the problem domain or in its classification ability. It is hoped that in providing insight with classification that data mining can be made more understandable and accessible to the non expert. Future directions for this work include the addition of influence weights for three and four feature value combinations to both test any increase in accuracy and to provide insight into important combinations of feature values. It is also envisaged that when classifying data with more than two class outcomes that a multidimensional scale may be useful to classification if not visualisation.

References

Blake, C.L., Newman, D.J., Hettich, S. & and Merz, C.J. (1988), UCI repository of machine learning databases.

Duda, R., Hart, P. (1973), *Pattern Classification and scene analysis.*, John Wiley and Sons.

Quinlan, J. (1993), *Programs for Machine Learning.*, Morgan Kaufmann

Setiono, R. & Liu, H. (1996), Symbolic Representation of Neural Networks, *in* ‘Computer’, Vol. 29,IEEE Computer Society Press, Los Alamitos, CA, USA, pp. 71–77.

Shafer, G. (1976), *A Mathematical theory of evidence.*, Princeton University Press.

Vapnik, V. (1999), *The nature of statistical learning theory.*, Springer - Verlag.

Witten, I.H. & Frank, E. (2000), *Data Mining: Practicle machine learning tools and techniques with java implementations.*, Morgan Kaufmann