Volume Under the ROC Surface for Multi-class Problems. Exact Computation and Evaluation of Approximations

Technical Report 2nd-May-2003

C. Ferri J. Hernández-Orallo M.A. Salido

[1] Dep. Sistemes Informàtics i Computació, Univ. Politècnica de València (Spain)
{cferri, jorallo, msalido}@dsic.upv.es

Abstract

Receiver Operating Characteristic (ROC) has been successfully applied to classifier problems with two classes. The Area Under the ROC Curve (AUC) has been determined as a better way to evaluate classifiers than predictive accuracy or error. However, the extension of the Area Under the ROC Curve for more than two classes has not been addressed to date, because of the complexity and elusiveness of its precise definition. In this paper, we present the real extension to the Area Under the ROC Curve in the form of the Volume Under the ROC Surface (VUS), showing how to compute the polytope that corresponds to the absence of classifiers (given only by the trivial classifiers), to the best classifier and to whatever set of classifiers. We compare the real VUS with "approximations" or "extensions" of the AUC for more than two classes.

Keywords: Cost-sensitive learning and evaluation, ROC analysis, AUC, constraint satisfaction.

1 Introduction

In general, classifiers are used to make predictions for decision support. Since predictions can be wrong, it is important to know what the effect is when the predictions are incorrect. In many situations not every error has the same consequences. Some errors have greater cost than others, especially in diagnosis. For instance, a wrong diagnosis or treatment can have different cost and dangers depending on which kind of mistake has been done. In fact, it is usually the case that misclassifications of minority classes into majority classes (e.g. predicting that a system is safe when it is not) have greater costs than misclassifications of majority classes into minority classes (e.g. predicting that a system is not safe when it actually is). Obviously, the costs of each misclassification are problem dependent, but it is almost never the case that they would be uniform for a single problem. Consequently, accuracy is not generally the best way to evaluate the quality of a classifier or a learning algorithm.

Cost-sensitive learning is a more realistic generalisation of predictive learning, and cost-sensitive models allow for a better and wiser decision making. The quality of a model is measured in terms of cost minimisation rather than in terms of error minimisation. When cost matrices are provided a priori, i.e. before learning takes place, the matrices have to be fully exploited to obtain models that minimise cost.

However, in many circumstances, costs are not known a priori or the models are just there to be evaluated or chosen. Receiver Operating Characteristic (ROC) analysis [31][40][33] has been proven very useful for evaluating given classifiers in these cases, when the cost matrix was not known when the classifiers were constructed. ROC analysis provides tools to select a set of classifiers that would behave optimally and reject some other useless classifiers. In order to do this, the convex hull of all the classifiers is constructed, giving a "curve" (a convex polygon).

ROC analysis and the AUC measure have been extensively used in the area of medical decision making [17][25][28][45], in the field of knowledge discovery, data mining, pattern recognition [1] and science in general [40].

In the trivial case, a single 2-class classifier forms a 4-segment ROC *curve* (a polygon in a strict sense) with the point given by the classifier, two trivial classifiers (the classifier that always says class 0 and the classifier that always says class 1) and the origin, whose area can be computed. This area is called the Area Under the ROC Curve (AUC) and has become a better alternative than accuracy (or error), i.e., percentage of instances that are correctly classified (respectively incorrectly classified), for evaluating classifiers.

However, the applicability of ROC analysis and the AUC has only been shown for problems with two classes. Although ROC analysis can be extended in theory for multi-dimensional problems [39], practical issues (computational complexity and representational comprehensibility, especially) preclude its use in practice. The major hindrance is the high dimensionality. A confusion matrix obtained from a problem of c classes has c^2 positions, and $(c\cdot(c-1))$ dimensions (d), i.e. all of the possible misclassification combinations are needed to evaluate a classifier. The problem is not only representation or visualisation, but the fact that complexity of computing a convex hull of n points for d dimensions has cost $O(n^d)$ [39]. For instance, for a problem of 4 classes, we would have cost 20^{12} for evaluating 20 classifiers.

Nonetheless, although difficult, it is possible to perform ROC analysis for more than two classes and to compute the AUC (more precisely, the Volume Under the ROC Surface, VUS). However, the trivial classifiers for more than two classes, the minimum volume, the maximum volume have not been identified to date in the literature.

There are two approximations or extensions for the AUC measure for more than 2 classes: Hand and Till [16], and Mossman [27]. The former is based on an extension based on the average of all the possible pairs of classes combination. The latter is based on the priorisation of one class over the others, and has been shown for three classes. However, these approximations have the nature of intuitive extensions, but they lack a theoretical justification of their goodness (and it seems that this justification would be quite difficult to be obtained). Consequently, being able to compute the real VUS for more than 2 classes would be very important to assess the quality of these approximations.

In this paper, we present the trivial classifiers, the equations, the maximum and minimum VUS, for classifiers of more than 2 classes. We use this to compute the real VUS for any classifier. We then compare experimentally the real VUS with several other approximations, showing which approximation is best.

2 Notions about cost-sensitive learning

In this section we include some classical notions about confusion matrices, cost matrices and ROC analysis. Those familiar with these issues should skip this section until subsection 2.4.

2.1 Misclassification Results and the Confusion/Misclassification Matrix

Given a classifier, it is usual that its accuracy could be lower than 100%, let us say, for instance, 87,5%. In this case, it may be interesting to know to which class the misclassified 12,5% goes and how this error is distributed.

A Confusion Matrix is a very practical and intuitive way of seeing such a distribution. Given 100 test examples and a classifier, an example of a Confusion Matrix for three classes {a, b, c} might be as follows:

		Actual		
		a	b	c
Predicted	a	20	2	3
	b	0	30	3
	c	0	2	40

This matrix is understood as follows. From the hundred examples, 20 were of class 'a' and all were correctly classified, 34 were of class 'b' from which 30 were correctly classified as 'b', 2 misclassified as 'a' and 2 misclassified as 'c'. Finally, 46 were of class 'c' from which 40 were correctly classified as 'c', 3 misclassified as 'a' and 3 misclassified as 'b'.

We use the notation M for this matrix and M(*i,j*) refers to the element of predicted class *i* and actual class *j*.

Note that it is easy to obtain derived information from the confusion matrix. The vertical sums give the distribution of classes in the actual examples; the horizontal sums give the distribution of classes produced by the classifier.

A Confusion *Ratio* Matrix can also be obtained by normalising each column to one:

		Actual		
		a	b	c
Predicted	a	1.0	0.059	0.065
	b	0.0	0.882	0.065
	c	0.0	0.059	0.87

This matrix is represented by $\overline{\mathrm{M}}$. Note that this matrix cannot be used instead of the original one because we have lost the class distribution.

As we have discussed before, frequently accuracy is a much too simplified measure of the quality of a classifier. For instance, given a dataset whose distribution of classes is (p_a= 0.85, p_b= 0.1, p_c= 0.05), i.e., most of the examples are of the class 'a', a simple classifier predicting everything into class 'a' would have 85% of accuracy.

Apart from using confusion matrices to avoid this kind of oversimplification, which can be misleading for assessing quality of classifiers, other measures have been introduced by using the separate predictive accuracies ($PAcc_i$). These can be based on the mean of the separate predictive accuracies, the product, the maximum or derived from informativeness measures. All of them try to balance the results of the accuracy of different classes, in order to improve the results for minority classes.

2.2 Processing given cost information

Given a classification problem domain, the information about the cost of correct or wrong classification can be given in very different degrees of detail.

2.2.1 *Cost Matrix*

The most complete way to provide the information about misclassification costs is a Cost Matrix (also known as Loss Matrix), which indicates the costs for correct and incorrect classifications. An example of a Cost Matrix for three classes {a, b, c} might be as follows:

		Actual		
		a	b	c
Predicted	a	-2.5	4	2
	b	2.1	-3.5	0
	c	1.2	1.3	-4

This example shows the usual portrait, the diagonal of the matrix shows the costs for correct classification (-2.5, -3.5, -4). These values are usually negative or zero, because a correct classification has benefits instead of costs. The other values represent different cases of misclassification. For instance, the value 2.1 in cell (b,a) means that classifying incorrectly an 'a' instance as a 'b' instance has a cost of 2.1.

We use the notation C for this matrix and C(i,j) refers to the element of predicted class *i* and actual class *j*.

From this matrix and the confusion matrix it is very easy to compute the cost of a classifier for a given dataset, just as the 1 by 1 matrix product, given a Resulting Matrix:

$$R(i,j) = M(i,j) \cdot C(i,j)$$

For instance, for the previous two examples, the resulting matrix would be:

		Actual		
		a	b	c
Predicted	a	-60	8	6
	b	0	-105	0
	c	0	2.6	-160

And just adding all the elements of the matrix, we have the overall cost:

$$Cost = \sum_i \sum_j R(i,j)$$

For small datasets or when these costs are computed for parts of a model (e.g. a part of a decision tree), it may be useful to compute a Laplace-corrected confusion matrix before, obtained as [24]:

$$M'(i,j) = n \frac{M(i,j) + \lambda}{c^2 \lambda + n}$$

where λ determines the strength of the Laplace correction, *c* the number of classes and *n* the cardinality.

2.2.2 *Cost Matrix Properties*

Elkan [11] presents some desirable cost matrix properties for the two classes case. Here we extend those properties to multiclass problems.

It is reasonable to assume that costs are greater for misclassification than for correct classification. This reasonableness condition can be stated as:

$$\forall i, j, j \neq i : C(i,i) < C(j,i)$$

The < sign could be replaced by a ≤ to make this condition looser. Another property that can be expected from a cost matrix is that there are no dominant rows, i.e., a row for which all the costs are less than the corresponding costs of other row. More formally.

$$\neg\exists j,k : \forall i : C(k,i) < C(j,i)$$

If this would not be the case, there will be some classes that will never be predicted in any situation whatsoever.

2.2.3 *Cost matrix normalisation*

Given a cost matrix it is easy to see that we can divide any element of the matrix by a constant and the resulting overall cost will be divided by that constant. However, using this property to normalise a matrix does not give any further advantage.

There is, though, a usual normalisation performed to a cost matrix. Provided the cost matrix always has lower costs for correct classifications than for incorrect classification (reasonableness property), the following normalisation can be performed:

1. For each column *i*, select the correct classification cost C(*i*,*i*).
2. Add –C(i,i) to each value of the *i*th column.

Let us see it with an example. Given the following cost matrix:

		Actual		
		a	b	c
Predicted	a	-3	15	3
	b	2	-4	0
	c	7	2	-1.5

The corresponding normalised matrix would be:

		Actual		
		a	b	c
Predicted	a	0	19	4.5
	b	5	0	1.5
	c	10	6	0

However, the normalised matrix cannot be used instead of the original one, because the correction of the derived cost is not independent to a specific dataset distribution. For instance, given the following class absolute frequencies: Na = 100, Nb= 200, Nc= 150, the original cost should be computed as:

$$\text{OriginalCost} = 100 \cdot (-3) + 200 \cdot (-4) + 150 \cdot (-1.5) + \text{Derived Cost}$$

2.3 Traditional ROC Analysis

The Receiver Operating Characteristic (ROC) analysis [31][40] allows the evaluation of classifier performance in a more independent and complete way than just using accuracy. ROC analysis has usually been presented for two classes, because it is easy to define, to interpret and it is computationally feasible.

In ROC analysis with two classes the following notation is used for the confusion matrix:

		Actual	
		T	F
Predicted	T	True Positives (TP)	False Positives (FP)
	F	False Negatives (FN)	True Negatives (TN)

And as we have said, the ROC Analysis is performed using the ratios, which are easily obtained:

- True Positive Rate: TPR = TP/(TP+FN)
- False Negative Rate: FNR = FN/(TP+FN)
- False Positive Rate: FPR = FP/(FP+TN)
- True Negative Rate: TNR = TN/(FP+TN)

This forms a confusion ratio matrix $\bar{M}$:

		Actual	
		T	F
Predicted	T	TPR	FPR
	F	FNR	TNR

Logically, TPR = 1 – FNR and FPR = 1 – TNR. These latter equivalences provides a means to just selecting two of the ratios and construct a bidimensional space with them, with a working area between (0,0) and (1,1).

ROC analysis is based on plotting the true-positive rate over the y-axis and the false-positive rate over the x-axis. For instance, given the following confusion matrix:

		Actual	
		T	F
Predicted	T	30	30
	F	20	70

we would have a TPR= 0.6 and a FPR= 0.3. We can easily draw this point in the ROC space (also called Lorentz diagram [15]) as is shown in Figure 1.

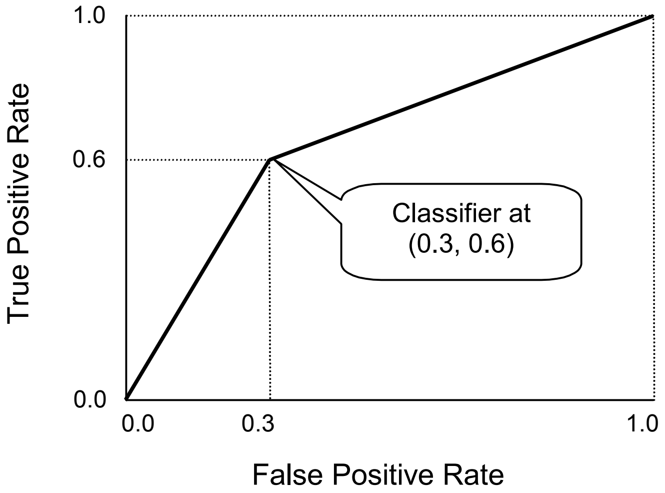


Figure 1. Example of ROC diagram

The points (0,0) and (1,1) represent, respectively, the classifier that classifies anything as negative and the classifier that classifies anything as positive. These are usually known as trivial classifiers.

There are many interesting things about the ROC analysis (summarised from [31]):

- Once a test set is presented, the actual cost given by the classifier can be computed from the actual probabilities of the two classes (p(T) and p(F)) and the cost matrix C(·,·), by:

  $$\text{Cost} = p(T) \cdot (FNR) \cdot C(F,T) + p(F) \cdot (FPR) \cdot C(T,F)$$

  assuming that correct classifications have no cost.

  From the previous formula, two classifiers 1 and 2 will have the same cost when:

  $$\frac{TPR_2 - TPR_1}{FPR_2 - FPR_1} = \frac{p(F) \cdot C(T,F)}{p(T) \cdot C(F,T)} = m$$

  This *m* can be seen as the slope of a line. All classifiers in the same line have the same performance and the lines are called *iso-performance* lines.
- Given two classifiers, we can obtain as many derived classifiers as we want all along the segment that connects them, just by voting them with different weights. Consequently, any point "below" that segment will have more cost for any class distribution and cost matrix, because it has lower true positive rate and higher false positive rate.
- According to that property, given several classifiers, one can discard the classifiers that fall under the convex hull formed by the points representing the classifiers. The convex hull that is formed with all the classifiers (jointly with the points (0,0), (1,0) and (1,1)) is known as ROC Curve.
- Due to the previous rationale, the best learning system is the one that produces a set of classifiers that maximises the Area Under the ROC Curve (AUC).

A typical snapshot of a ROC diagram is illustrated in Figure 2. Suppose that the five classifiers plotted have been obtained A, B, C, D and E. Classifiers B and E fall under the ROC Curve (the convex hull formed by the five classifiers and the two trivial classifiers). Hence, B and E are discarded.

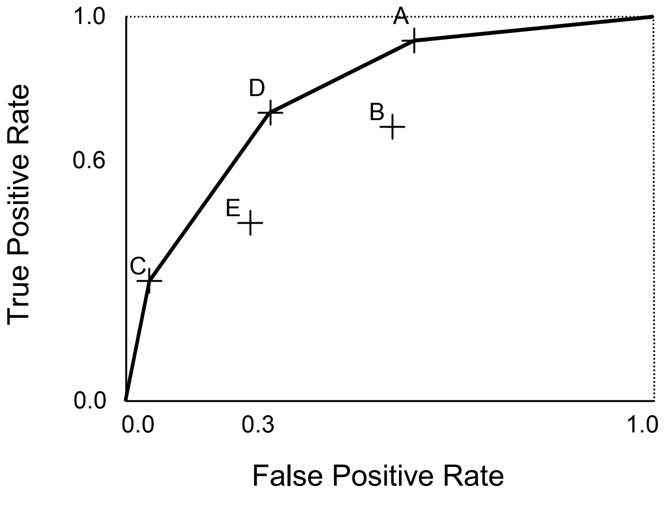


Figure 2. Example of a ROC curve

2.4 Extending ROC Analysis

Although the previous section shows the usual way to represent ROC space, it is not, in our opinion, a very coherent way, since the true class is represented incrementally for correct predictions and the false class is represented incrementally for incorrect predictions. Moreover, as we will see, this choice is not extensible for more than two classes. Consequently, for the previous classifier, we have a FNR= 0.4 and a FPR= 0.3, and we can draw the ROC space as is shown in Figure 3.

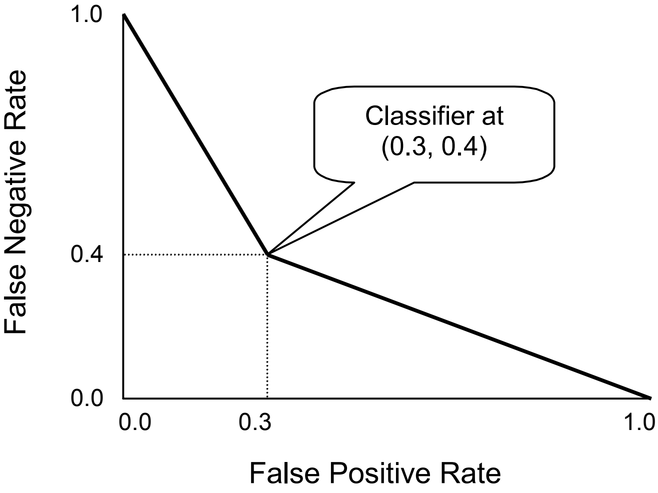


Figure 3. Example of inverse ROC diagram

Now, the points (0,1) and (1,0) represent, respectively, the classifier that classifies anything as negative and the classifier that classifies anything as positive. The ROC curve is now computed with points (0,1), (1,0) and (1,1).

Obviously, with this new diagram, instead of looking for the maximisation of the Area Under the ROC Curve (AUC) we have to look for its minimisation. A better option is to compute the Area Above the ROC Curve (AAC). In order to maintain notation with classical terminology, we will refer to the AAC also as AUC.

Given a confusion ratio matrix $\overline{\text{M}}$:

		Actual	
		T	F
Predicted	T	1 - y	x
	F	y	1 – x

For this representation, this area is given by the following classifiers:

- The point representing the classifier (x, y).
- The 2 trivial classifiers: everything is classified as T (1, 0) and everything is classified as F (0, 1).
- The 1 everything-wrong classifier (1, 1).

This area is illustrated in the following figure:

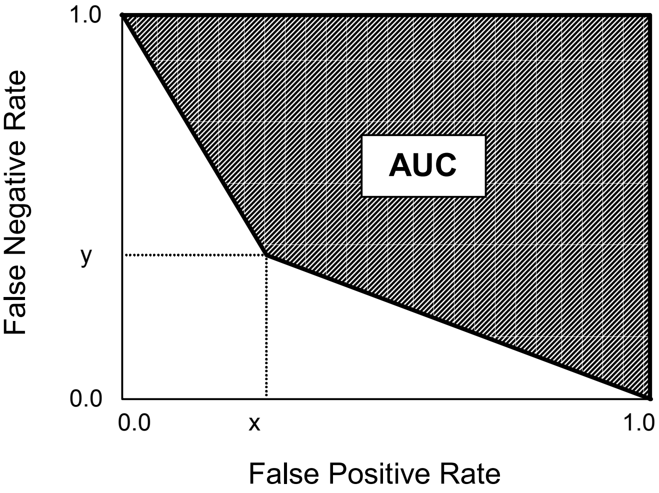


Figure 4. Example of AUC of an inverse ROC diagram

3 ROC Analysis for more than two dimensions

Srinivasan has shown [39] that, theoretically, the ROC analysis extends to more than two classes "directly". For *c* classes, and assuming a normalised cost matrix, we have to construct a vector of *d* = *c*(*c*-1) dimensions for each classifier. In general the cost of a classifier for *c* classes is:

$$Cost = \sum_{i,j,i\neq j} p(i)\cdot C(i,j)\overline{M}(i,j)$$

where $\overline{\mathrm{M}}$ is the confusion ratio matrix and *p*(*i*) is the absolute frequency of class *i*. From the previous formula, two classifiers 1 and 2 will have the same cost when they are on the same iso-performance hyperplane. However, the *d*-1 values of the hyperplane are not so straightforward and easy to obtain and understand as the slope value of the bi-dimensional case.

In the same way as the bi-dimensional case, the convex hull can be constructed, forming a polytope. To know if a classifier can be rejected, it has to be seen if the intersection of the current polytope with the polytope of the new classifier give the new polytope, i.e., the new polytope is included in the first polytope [21].

Provided this direct theoretical extension, there are some problems.

- In two dimensions, doubling *p*(*T*) (i.e. doubling the probability of one class) has the same effect on performance as doubling the cost *C*(*F,T*) or halving the cost *C*(*T,F*). Is this still true for more than two dimensions? The answer is *no* in one direction, because for a change in class distributions, there are infinite many corresponding cost matrices due to many more degrees of freedom.
- The best algorithm for the convex hull generation is O(N log N + $N^{d/2}$) [21][5].
- In the 2-d case, it is relatively straightforward how to detect the trivial classifiers and the points for the minimum and maximum cases.

However, not only there are computational limitations but representational ones. ROC analysis in two dimensions has a very nice and understandable representation, but this cannot be directly extended to more than two classes, because even for 3 classes we have a 6-dimensional space, quite difficult to represent on a paper or a screen[1].

First of all, let us try to rashly extend the points used in the 2 classes case for more than two classes.

3.1 Hasty Extension of the Maximum and Minimum Polytopes

For two classes, we know that the maximum polygon is given by:

- The point representing the best classifier (classifier with no error) (0, 0).
- The 2 trivial classifiers: everything is classified as True (1, 0) and everything is classified as False (0, 1).
- The 1 everything-wrong classifier (1, 1).

This gives 4 points, 4 segments and an AUC = 1.

The minimum polygon is given by:

- The 2 trivial classifiers: everything is classified as True (1, 0) and everything is classified as False (0, 1).
- The 1 everything-wrong classifier (1, 1).

This gives 3 points, 3 segments and an AUC = 1/2.

How does this extend to more than 2 classes?

3.1.1 Reckless (but Wrong) Extension

Given a cost ratio matrix, we have to represent the misclassification cells. For instance, let us consider the following cost ratio matrix for three classes:

		Actual		
		a	*b*	*c*
Predicted	*a*	h_a	x_1	x_2
	b	x_3	h_b	x_4
	c	x_5	x_6	h_c

This gives a 6-dimensional point (x_1, x_2, x_3, x_4, x_5, x_6). The values h_a, h_b and h_c are dependent and do not need to be represented, because:

- $h_a + x_3 + x_5 = 1$
- $h_b + x_1 + x_6 = 1$
- $h_c + x_2 + x_4 = 1$

One can think that the extension would be trivial just by behaving in the same way as the 2-class case. The maximum would be:

- The point representing the best classifier (0, 0, 0, 0, 0, 0).
- The three trivial classifiers: everything is *a* (1, 1, 0, 0, 0, 0), everything is *b* (0, 0, 1, 1, 0, 0), and everything is *c* (0, 0, 0, 0, 1, 1).
- The 8 extreme everything-wrong classifiers:
  - (a->b), (b->a), (c->a) corresponds to (1, 1, 1, 0, 0, 0)
  - (a->b), (b->a), (c->b) corresponds to (1, 0, 1, 1, 0, 0)

[1] In the appendix, we show some graphical alternatives for the classical ROC representation for more than two classes.

- (a->b), (b->c), (c->a) corresponds to (0, 1, 1, 0, 0, 1)
- (a->b), (b->c), (c->b) corresponds to (0, 0, 1, 1, 0, 1)
- (a->c), (b->a), (c->a) corresponds to (1, 1, 0, 0, 1, 0)
- (a->c), (b->a), (c->b) corresponds to (1, 0, 0, 1, 1, 0)
- (a->c), (b->c), (c->a) corresponds to (0, 1, 1, 0, 1, 1)
- (a->c), (b->c), (c->b) corresponds to (0, 0, 0, 1, 1, 1)

The volume of these 12 points is: 1/40

And the minimum would be:

- The three trivial classifiers: everything is *a* (1, 1, 0, 0, 0, 0), everything is *b* (0, 0, 1, 1, 0, 0), and everything is *c* (0, 0, 0, 0, 1, 1).
- The 8 extreme everything-wrong classifiers:
  - (a->b), (b->a), (c->a) corresponds to (1, 1, 1, 0, 0, 0)
  - (a->b), (b->a), (c->b) corresponds to (1, 0, 1, 1, 0, 0)
  - (a->b), (b->c), (c->a) corresponds to (0, 1, 1, 0, 0, 1)
  - (a->b), (b->c), (c->b) corresponds to (0, 0, 1, 1, 0, 1)
  - (a->c), (b->a), (c->a) corresponds to (1, 1, 0, 0, 1, 0)
  - (a->c), (b->a), (c->b) corresponds to (1, 0, 0, 1, 1, 0)
  - (a->c), (b->c), (c->a) corresponds to (0, 1, 1, 0, 1, 1)
  - (a->c), (b->c), (c->b) corresponds to (0, 0, 0, 1, 1, 1)

The volume of these 11 points is 1/120.

And as a natural consequence one would obtain that the maximum is three times bigger than the minimum, quite similarly to the two classes case, where the maximum is two times bigger than the minimum.

However, as we see next, this extension is not valid and does not correspond to the real ROC analysis for three classes.

3.2 Back to the Discarding Conditions

Let us begin by considering the maximum volume. The maximum volume should represent the volume containing all the possible classifiers.

3.2.1 Maximum VUS

A point in the 1-long hypercube is a classifier if and only if:

Valid Classifier Conditions	
	• $x_3 + x_5 <= 1$
	• $x_1 + x_6 <= 1$
	• $x_2 + x_4 <= 1$

It is easy to obtain the volume of the space determined by these equations, just by using the probability of 6 random numbers under a uniform distribution U(0,1) would follow the previous conditions.

More precisely, given 6 values under a uniform distribution between 0 and 1, represented by U(0,1), we have that the probability that a point formed by these six values follows the previous three conditions is:

$$\begin{aligned} VUS_3^{max} &= P(U(0,1) + U(0,1) \le 1) \cdot P(U(0,1) + U(0,1) \le 1) \cdot P(U(0,1) + U(0,1) \le 1) \\ &= P(U(0,1) + U(0,1) \le 1)^3 . \end{aligned}$$

It is easy to see that the probability that the sum of two random numbers under the distribution U(0,1) is less than 1 is exactly ½ , i.e.

$$P(U(0,1) + U(0,1) \leq 1) = \frac{1}{2}$$

Consequently:

$$VUS_3^{max} = (\tfrac{1}{2})^3 = 1/8$$

Quite different from the previous value of 1/40, that is shown to be incorrect.

And now that we know the exact maximum Volume Under the ROC Surface for three classes (1/8); what about the minimum?

3.2.2 *Minimum VUS*

Now let us try to derive the minimum VUS.

Without any knowledge we can construct trivial classifiers as follows:

		Actual		
		a	*b*	*c*
	a	h_a	h_a	h_a
Predicted	*b*	h_b	h_b	h_b
	c	h_c	h_c	h_c

where $h_a + h_b + h_c = 1$. These obviously include the three extreme trivial classifiers "everything is a", "everything is b" and "everything is c".

Given a classifier:

		Actual		
		a	*b*	*c*
	a	v_{aa}	v_{ba}	v_{ca}
Predicted	*b*	v_{ab}	v_{bb}	v_{cb}
	c	v_{ac}	v_{bc}	v_{cc}

we can discard this classifier iff it is above of a trivial classifier , formally:

$$\exists h_a, h_b, h_c \in \mathbb{R}^+ \text{ where } (h_a + h_b + h_c = 1) \text{ such that:}$$

$$v_{ba} \geq h_a$$

$$v_{ca} \geq h_a$$

$$v_{ab} \geq h_b$$

$$v_{cb} \geq h_b$$

$$v_{ac} \geq h_c$$

$$v_{bc} \geq h_c$$

From here we can derive the following theorem:

Theorem 1:

Without any knowledge, a classifier $(x_1, x_2, x_3, x_4, x_5, x_6)$ can be discarded if and only if:

$$r_1 + r_2 + r_3 \geq 1$$

where $r_1 = \min(x_1, x_2)$, $r_2 = \min(x_3, x_4)$ and $r_3 = \min(x_5, x_6)$.

Proof:

(if) if $r_1 + r_2 + r_3 \geq 1$ then the classifier is equal or worst than:

			Actual	
		a	*b*	*c*
	a	v_{aa}	r_1	r_1
Predicted	*b*	r_2	v_{bb}	r_2
	c	r_3	r_3	v_{cc}

Logically there exists a trivial classifier:

			Actual	
		a	*b*	*c*
	a	h_a	h_a	h_a
Predicted	*b*	h_b	h_b	h_b
	c	h_c	h_c	h_c

with $(h_a + h_b + h_c = 1)$ such that :

$$r_1 \geq h_a$$
$$r_1 \geq h_a$$
$$r_2 \geq h_b$$
$$r_2 \geq h_b$$
$$r_3 \geq h_c$$
$$r_3 \geq h_c$$

because $r_1 + r_2 + r_3 \geq 1$, and hence can be discarded.

(only if)

If a classifier $(x_1, x_2, x_3, x_4, x_5, x_6)$ can be discarded then

$$\exists h_a, h_b, h_c \in \mathrm{R}^+ \text{ where } (h_a + h_b + h_c = 1) \text{ such that:}$$

$$x_1 \geq h_a$$
$$x_2 \geq h_a$$
$$x_3 \geq h_b$$
$$x_4 \geq h_b$$
$$x_5 \geq h_c$$
$$x_6 \geq h_c$$

From:

$$x_1 \geq h_a$$
$$x_2 \geq h_a$$

we have that

$$r_1 = \min(x_1, x_2) \geq h_a$$

In a similar way we have

$$r_2 = \min(x_3, x_4) \geq h_b$$
$$r_3 = \min(x_5, x_6) \geq h_b$$

And then if $r_1 + r_2 + r_3 \geq h_a + h_b + h_c = 1$.

Consequently, $r_1 + r_2 + r_3 \geq 1$.

Given the previous property, we only have to compute the space of classifiers that follow the condition that $r_1 + r_2 + r_3 \geq 1$ where $r_1 = \min(x_1, x_2)$, $r_2 = \min(x_3, x_4)$ and $r_3 = \min(x_5, x_6)$ to obtain the minimum volume corresponding to total absence of information.

More precisely, we have to compute the volume formed by this condition jointly with the valid classifier conditions, i.e.:

- $x_3 + x_5 <= 1$
- $x_1 + x_6 <= 1$
- $x_2 + x_4 <= 1$
- $r_1 + r_2 + r_3 \geq 1$ where $r_1 = \min(x_1, x_2)$, $r_2 = \min(x_3, x_4)$ and $r_3 = \min(x_5, x_6)$

This volume is more difficult to be obtained by a probability estimation, due to the min function. Let us compute this volume using a Monte Carlo method.

3.3 Monte Carlo Method for obtaining Max and Min VUS

Monte Carlo methods are used to randomly generate a subset of cases from a problem space and estimate the probability that a random case follows a set of conditions. These methods are particularly interesting to approximate volumes, such as the volume under the ROC curve we are dealing with.

For this purpose, we generate an increasing number of points in the 6D hypercube of length 1 (i.e., we generate six variables x_1, x_2, x_3, x_4, x_5, x_6 using a uniform distribution between 0 and 1) and then check whether or not they follow the previous maximum or minimum conditions. Since we are working with a 1-length hypercube, the proportion of cases following the conditions is exactly the volume we are looking for.

In particular, we have obtained the following results:

- Maximum: 0.12483 for 1,000,000 cases, that matches quite exactly our theoretical $VUS_3^{max} = 1/8$.
- Minimum: 0.00555523 for 1,000,000 cases, which is approximately 1/180.

However, although we have obtained the exact maximum, we have not obtained the exact minimum (although 1/180 is conjectured). How can we obtain the real VUS^{min} ? And, more importantly, how can we obtain the ROC polytopes that form these volumes?

4 Using Constraint Satisfaction HyperPolyhedra for obtaining the ROC Polytopes

In the previous section we have developed the conditions for the maximum and minimum VUS, given, respectively, when the best classifier is known (0, 0, 0, 0, 0, 0) and when no classifier is given (absence of information). However, we are interested in a way to obtain the border points of each space, i.e., the polytopes that represent both cases.

What we need is a way to compute these polytopes given the set of conditions. A general system able to do this was not available since quite recently. Such a system is HSA.

4.1 Hyperpolyhedron Search Algorithm (HSA)

Nowadays, many real problems can be efficiently modelled as Constraint Satisfaction Problems (CSP's). Most of these problems can be naturally modelled using non-binary constraints over continuous variables. However, researchers traditionally had focused on binary constraints and discrete variables, due to the simplicity of dealing with binary constraints and the fact that any non-binary CSP can be transformed into an equivalent binary one. However, this transformation may not be practical in many problems.

Traditional CSP techniques obtain the solution by searching systematically through the possible assignments of values to variables. The complexity of these techniques increases exponentially with the number of variables, number of constraints and domain length.

The Hyperpolyhedron Search Algorithm (HSA) [36][37] solves non-binary and continuous constraint satisfaction problems in a natural way as a non-binary CSP solver. HSA is a constraint propagation algorithm that carries out the search through a hyperpolyhedron that maintains in its vertices those solutions that satisfy all non-binary constraints. In HSA, the handling of the non-binary constraints (linear inequations) can be seen as a global hyperpolyhedron constraint. Initially, the hyperpolyhedron is created by the Cartesian Product of the variable domain bounds. For each constraint, HSA checks the consistency, updating the hyperpolyhedron (if the constraint is consistent), by means of linear programming techniques. This constraint is a hyperplane that is intersected to obtain the new hyperpolyhedron vertices. The resulting hyperpolyhedron is a convex set of solutions to the CSP.

A solution to a CSP is an assignment of a value from its domain to every variable such that all constraints are satisfied. The objective in our CSP solver may be determining:

- whether a solution exists, that is, if the CSP is consistent;
- one solution, several solution or the extreme solutions;
- the minimal variable domain;
- an optimal, or a good solution by means of an objective function defined in terms of certain variables.

In the ROC surface problem we focus in this paper, the objective of HSA will be to determine the extreme solutions in order to calculate the convex hull of the resulting hyperpolyhedron.

Since the HSA can handle open hyperpolyhedra (not forming a closed polytope), HSA does not compute the volume of the hyperpolyhedron. For this purpose, we are going to use QHull [2]. QHull is, among other things, an algorithm that implements a quick method for computing the convex hull of a set of points and the volume of the hull.

Now, we are ready to obtain the points that conform the max and min volumes.

4.2 Maximum VUS points for 3 classes

Let us recover the equations for the maximum volume (valid classifier conditions):

- $x_3 + x_5 <= 1$
- $x_1 + x_6 <= 1$
- $x_2 + x_4 <= 1$

We introduce these equations to the HSA and look for solutions for these six variables. We obtain the following 41 points:

(1): (1.00, 1.00, 0.50, 0.00, 0.50, 0.00)	(15): (1.00, 1.00, 1.00, 0.00, 0.00, 0.00)	(29): (0.00, 1.00, 0.00, 0.00, 0.00, 0.00)
(2): (1.00, 0.00, 0.50, 1.00, 0.50, 0.00)	(16): (1.00, 1.00, 0.00, 0.00, 1.00, 0.00)	(30): (0.00, 0.00, 1.00, 1.00, 0.00, 1.00)
(3): (1.00, 0.00, 0.50, 0.00, 0.50, 0.00)	(17): (1.00, 1.00, 0.00, 0.00, 0.00, 0.00)	(31): (0.00, 0.00, 1.00, 1.00, 0.00, 0.00)
(4): (0.00, 1.00, 0.50, 0.00, 0.50, 0.50)	(18): (1.00, 0.00, 1.00, 1.00, 0.00, 0.00)	(32): (0.00, 0.00, 1.00, 0.00, 0.00, 1.00)
(5): (0.00, 1.00, 0.50, 0.00, 0.50, 0.00)	(19): (1.00, 0.00, 1.00, 0.00, 0.00, 0.00)	(33): (0.00, 0.00, 1.00, 0.00, 0.00, 0.00)
(6): (0.00, 0.00, 0.50, 1.00, 0.50, 0.50)	(20): (1.00, 0.00, 0.00, 1.00, 1.00, 0.00)	(34): (0.00, 0.00, 0.00, 1.00, 1.00, 1.00)
(7): (0.00, 0.00, 0.50, 1.00, 0.50, 0.00)	(21): (1.00, 0.00, 0.00, 1.00, 0.00, 0.00)	(35): (0.00, 0.00, 0.00, 1.00, 1.00, 0.00)
(8): (0.00, 0.00, 0.50, 0.00, 0.50, 1.00)	(22): (1.00, 0.00, 0.00, 0.00, 1.00, 0.00)	(36): (0.00, 0.00, 0.00, 1.00, 0.00, 1.00)
(9): (1.00, 0.50, 1.00, 0.50, 0.00, 0.00)	(23): (1.00, 0.00, 0.00, 0.00, 0.00, 0.00)	(37): (0.00, 0.00, 0.00, 1.00, 0.00, 0.00)
(10): (1.00, 0.50, 0.00, 0.50, 1.00, 0.00)	(24): (0.00, 1.00, 1.00, 0.00, 0.00, 1.00)	(38): (0.00, 0.00, 0.00, 0.00, 1.00, 1.00)
(11): (1.00, 0.50, 0.00, 0.50, 0.00, 0.00)	(25): (0.00, 1.00, 1.00, 0.00, 0.00, 0.00)	(39): (0.00, 0.00, 0.00, 0.00, 1.00, 0.00)
(12): (0.00, 0.50, 1.00, 0.50, 0.00, 1.00)	(26): (0.00, 1.00, 0.00, 0.00, 1.00, 1.00)	(40): (0.00, 0.00, 0.00, 0.00, 0.00, 1.00)
(13): (0.00, 0.50, 0.00, 0.50, 1.00, 1.00)	(27): (0.00, 1.00, 0.00, 0.00, 1.00, 0.00)	(41): (0.00, 0.00, 0.00, 0.00, 0.00, 0.00)
(14): (0.00, 0.50, 0.00, 0.50, 0.00, 1.00)	(28): (0.00, 1.00, 0.00, 0.00, 0.00, 1.00)	

Whose volume is, as expected, 0.125 (1/8).

HSA generates a convex polytope but does not check whether some of the points can be eliminated. In this case, some of these points are exactly on the surface of the volume and can be removed without modifying the volume. The simplified set of 27 points is:

VUS$_3^{max}$ POINTS		
111000	011001	001000
110010	011000	000111
110000	010011	000110
101100	010010	000101
101000	010001	000100
100110	010000	000011
100100	001101	000010
100010	001100	000001
100000	001001	000000

whose volume is, once again, 1/8, and contains:

- The 1 point representing the best classifier (0, 0, 0, 0, 0, 0).
- The 3 trivial classifiers: everything is *a* (1, 1, 0, 0, 0, 0), everything is *b* (0, 0, 1, 1, 0, 0), and everything is *c* (0, 0, 0, 0, 1, 1).
- The 8 extreme everything-wrong classifiers:
  - (a->b), (b->a), (c->a) corresponds to (1, 1, 1, 0, 0, 0)
  - (a->b), (b->a), (c->b) corresponds to (1, 0, 1, 1, 0, 0)
  - (a->b), (b->c), (c->a) corresponds to (0, 1, 1, 0, 0, 1)
  - (a->b), (b->c), (c->b) corresponds to (0, 0, 1, 1, 0, 1)
  - (a->c), (b->a), (c->a) corresponds to (1, 1, 0, 0, 1, 0)
  - (a->c), (b->a), (c->b) corresponds to (1, 0, 0, 1, 1, 0)
  - (a->c), (b->c), (c->a) corresponds to (0, 1, 1, 0, 1, 1)
  - (a->c), (b->c), (c->b) corresponds to (0, 0, 0, 1, 1, 1)
- The 6 points representing the classifiers that always classify correctly two classes and the third class always wrong into another class.
  - (a->a), (b->b), (c->a) corresponds to (0, 1, 0, 0, 0, 0).
  - (a->a), (b->b), (c->b) corresponds to (0, 0, 0, 1, 0, 0).
  - (a->a), (c->c), (b->a) corresponds to (1, 0, 0, 0, 0, 0).
  - (a->a), (c->c), (b->c) corresponds to (0, 0, 0, 0, 0, 1).
  - (b->b), (c->c), (a->b) corresponds to (0, 0, 1, 0, 0, 0).
  - (b->b), (c->c), (a->c) corresponds to (0, 0, 0, 0, 1, 0).

- 3 points representing the classifiers that always classify correctly one class and the other two are mixed.
    - (a->a), (b->c), (c->b) corresponds to (0, 0, 0, 1, 0, 1)
    - (b->b), (a->c), (c->a) corresponds to (0, 1, 0, 0, 1, 0)
    - (c->c), (a->b), (b->a) corresponds to (1, 0, 1, 0, 0, 0)
- 6 other points representing the classifiers that always classify correctly one class and the other two are mixed in such a way that the one class is never predicted.
    - (a->a), (b->a), (c->b) corresponds to (1, 0, 0, 1, 0, 0)
    - (a->a), (b->c), (c->a) corresponds to (0, 1, 0, 0, 0, 1)
    - (b->b), (a->b), (c->a) corresponds to (0, 1, 1, 0, 0, 0)
    - (b->b), (a->c), (c->b) corresponds to (0, 0, 0, 1, 1, 0)
    - (c->c), (a->b), (b->c) corresponds to (0, 0, 1, 0, 0, 1)
    - (c->c), (a->c), (b->a) corresponds to (1, 0, 0, 0, 1, 0)

All these points can be summarised as the:

- 3^3 classifiers where each column has a 1 and two 0's.

4.3 Minimum VUS for 3 classes

From theorem 1, in order to compute the minimum VUS, we only have to compute the space of classifiers that follow the condition that $r_1 + r_2 + r_3 \geq 1$ where r_1 = min(X1, X2), r_2 = min(X3, X4) and r_3 = min(X5, X6) to obtain the minimum volume corresponding to total absence of information.

If we use this condition jointly with the hyper-cube conditions, we have:

- $x_3 + x_5 <= 1$
- $x_1 + x_6 <= 1$
- $x_2 + x_4 <= 1$
- $r_1 + r_2 + r_3 \geq 1$ where $r_1 = \min(x_1, x_2)$, $r_2 = \min(x_3, x_4)$ and $r_3 = \min(x_5, x_6)$

Since the min function is not directly handled by HSA, we convert this latter equation into these eight equations:

$$x_1 + x_3 + x_5 \geq 1$$

$$x_1 + x_3 + x_6 \geq 1$$

$$x_1 + x_4 + x_5 \geq 1$$

$$x_1 + x_4 + x_6 \geq 1$$

$$x_2 + x_3 + x_5 \geq 1$$

$$x_2 + x_3 + x_6 \geq 1$$

$$x_2 + x_4 + x_5 \geq 1$$

$$x_2 + x_4 + x_6 \geq 1$$

and now we obtain the following 25 points:

(0.6667, 0.6667, 1.00, 0.3333, 0.00, 0.00)	(1.00, 0.50, 0.50, 0.50, 0.00, 0.00)	(1.00, 1.00, 1.00, 0.00, 0.00, 0.00)
(1.00, 0.6667, 0.3333, 0.3333, 0.3333, 0.00)	(0.50, 0.50, 1.00, 0.50, 0.00, 0.50)	(1.00, 1.00, 0.00, 0.00, 1.00, 0.00)
(0.6667, 0.6667, 0.00, 0.3333, 1.00, 0.3333)	(0.6667, 0.6667, 1.00, 0.3333, 0.00, 0.00)	(1.00, 1.00, 0.00, 0.00, 0.00, 0.00)
(1.00, 0.50, 0.50, 0.50, 0.50, 0.00)	(0.00, 0.00, 0.50, 1.00, 0.50, 0.50)	(1.00, 0.00, 1.00, 1.00, 0.00, 0.00)
(0.50, 0.50, 0.00, 0.50, 1.00, 0.50)	(1.00, 1.00, 0.50, 0.00, 0.50, 0.00)	(0.00, 1.00, 0.00, 0.00, 1.00, 1.00)
(1.00, 0.50, 0.50, 0.50, 0.50, 0.00)	(0.00, 0.00, 0.50, 1.00, 0.50, 0.50)	(0.00, 0.00, 1.00, 1.00, 0.00, 1.00)

(0.50, 0.50, 0.00, 0.00, 1.00, 0.50) (1.00, 0.50, 0.50, 0.50, 0.00, 0.00)	(1.00, 0.50, 1.00, 0.50, 0.00, 0.00) (0.00, 0.50, 0.00, 0.50, 1.00, 1.00)	(0.00, 0.00, 1.00, 1.00, 0.00, 0.00) (0.00, 0.00, 0.00, 1.00, 1.00, 1.00) (0.00, 0.00, 0.00, 0.00, 1.00, 1.00)

whose volume is 0.0055, which is approximately 1/180 and matches the volume obtained by the Monte Carlo method.

Some of these points are exactly on the surface of the volume and can be removed without modifying the volume. The simplified set of 9 points is:

VUS_3^{min} POINTS		
111000	101100	001100
110010	010011	000011
110000	001101	000111

whose volume is 1/180 and contains:

- The 3 trivial classifiers: everything is *a* (1, 1, 0, 0, 0, 0), everything is *b* (0, 0, 1, 1, 0, 0), and everything is *c* (0, 0, 0, 0, 1, 1).
- 6 of the 8 extreme everything-wrong classifiers:
  - (a->b), (b->a), (c->a) corresponds to (1, 1, 1, 0, 0, 0)
  - (a->b), (b->a), (c->b) corresponds to (1, 0, 1, 1, 0, 0)
  - (a->b), (b->c), (c->a) corresponds to (0, 1, 1, 0, 0, 1). This is not included
  - (a->b), (b->c), (c->b) corresponds to (0, 0, 1, 1, 0, 1)
  - (a->c), (b->a), (c->a) corresponds to (1, 1, 0, 0, 1, 0)
  - (a->c), (b->a), (c->b) corresponds to (1, 0, 0, 1, 1, 0). This is not included.
  - (a->c), (b->c), (c->a) corresponds to (0, 1, 1, 0, 1, 1)
  - (a->c), (b->c), (c->b) corresponds to (0, 0, 0, 1, 1, 1)

We have detected two classifiers that are not included in the minimum polytope, and that we did include in the wrong extension of section 3.1.1.

In fact, we can express in a more clear way which are these six points:

- These 6 points correspond to the 3 trivial classifiers modified in such a way that classify everything into one class except from the original from that class, i.e.:
  - (a->b), (b->a), (c->a) corresponds to (1, 1, 1, 0, 0, 0)
  - (a->b), (b->a), (c->b) corresponds to (1, 0, 1, 1, 0, 0)
  - (a->b), (b->c), (c->b) corresponds to (0, 0, 1, 1, 0, 1)
  - (a->c), (b->a), (c->a) corresponds to (1, 1, 0, 0, 1, 0)
  - (a->c), (b->c), (c->a) corresponds to (0, 1, 0, 0, 1, 1)
  - (a->c), (b->c), (c->b) corresponds to (0, 0, 0, 1, 1, 1)

And, more importantly, the whole set of 9 classifiers can be defined exactly by:

- The 9 classifiers that have two 1's in the same row.

Summing up, we have the following VUS for 3 classes:

	Theoretical	MONTE CARLO (1,000,000 CASES)	exact (hSA)
VUS_3^{max}	1/8	0.12483	0.125
VUS_3^{min}	1/180 (conjecture)	0.00555523	0.0055555556

4.4 Computing the VUS of Any Classifier

Now it seems that we can obtain the VUS of any classifier just be adding the coordinates of the point it represents and adding them as a new point to the minimum and then computing the convex hull.

However, this would be again a hasty step. The surprise would come up if we take the minimum (9 points, 1/180) and add the origin (the best classifier with no error at all). In this case we obtain 10 points and 1/120 volume, which is a greater volume but it is not the maximum. But if we have the best classifier, we should obtain the maximum volume.

The reason is the following. When we have the perfect classifier, represented by the point (0, 0, 0, 0, 0, 0), any classifier that has a value equal or greater than 0 in any coordinate is discardable and, logically, this should give 1/8, not 1/120. The issue is that whenever we add a new classifier we have to consider the conditions it produces.

The perfect classifier generates the following discard equations:

$$x_1 \geq 0$$

$$x_2 \geq 0$$

$$x_3 \geq 0$$

$$x_4 \geq 0$$

$$x_5 \geq 0$$

$$x_6 \geq 0$$

These disequations are null, because all the values should be positive, and, hence, we only have the valid classifier conditions, and then we should have the maximum volume 1/8.

Now let us consider the same thing for any arbitrary classifier C1:

			Actual	
		a	*b*	*c*
Predicted	*a*	z_{aa}	z_{ba}	z_{ca}
	b	z_{ab}	z_{bb}	z_{cb}
	c	z_{ac}	z_{bc}	z_{cc}

What can be discarded?

Any classifier such that is improved by the classifier C1 (combined with the trivial classifiers) , i.e., any classifier that would have greater values for the 6 dimensions.

Consequently, given a new classifier C2:

			Actual	
		a	*b*	*c*
Predicted	*a*	v_{aa}	v_{ba}	v_{ca}
	b	v_{ab}	v_{bb}	v_{cb}
	c	v_{ac}	v_{bc}	v_{cc}

We have to look at all the classifiers constructed as a the linear combination of the three trivial classifiers and the classifier C1 and see whether C2 is worse than any of the constructed classifiers.

Formally, the linear combination is defined as:

$$h_a \cdot (1, 1, 0, 0, 0, 0) + h_b \cdot (0, 0, 1, 1, 0, 0) + h_c \cdot (0, 0, 0, 0, 1, 1) + h_d \cdot (z_{ba}, z_{ca}, z_{ab}, z_{cb}, z_{ac}, z_{bc})$$

And we can discard C2 when

$$\exists h_a, h_b, h_c, h_d \in \mathrm{R}^+ \text{ where } (h_a + h_b + h_c + h_d = 1) \text{ such that:}$$

$$v_{ba} \geq h_a + 0 + 0 + h_d \cdot z_{ba}$$

$$v_{ca} \geq h_a + 0 + 0 + h_d \cdot z_{ca}$$

$$v_{ab} \geq 0 + h_b + 0 + h_d \cdot z_{ab}$$

$$v_{cb} \geq 0 + h_b + 0 + h_d \cdot z_{cb}$$

$$v_{ac} \geq 0 + 0 + h_c + h_d \cdot z_{ac}$$

$$v_{bc} \geq 0 + 0 + h_c + h_d \cdot z_{bc}$$

This gives a system of disequations with 10 variables, that can be input to HSA, and then we can use the 6 obtained variables of the point C2.

This represents a way to obtain the VUS of any point. Let us see some examples:

	MONTE CARLO (100,000 cases)	exact (hSA)
(0, 0, 0, 0, 0, 0)	0.12467	0.125
(1, 1, 1, 0, 0, 0)	0.005547	0.0055556
(0.5, 0.5, 0, 0, 0, 0)	0.0327	0.0342
(0.5, 0, 0.5, 0, 0, 0)	0.0291	0.03105
(0.75, 0.75, 0.75, 0, 0, 0)	0.0076	0.0083279
(0.3333, 0, 0.3333, 0, 0, 0)	0.052186...	0.052214

Example of VUS_3 for some points

As we see, the two methods agree significantly, although HSA method gives the exact value much quicker than the Monte Carlo method. Moreover, the two first points match the maximum and minimum volumes, as expected.

4.5 Real VUS for More than One Classifier

In the same way as before, given a set of classifiers, we can compute the true VUS of the set, just generalising the previous formula.

Let us illustrate it for 4 classifiers *Z, W, Y* and *X*. In fact, what we have to do is to consider the linear combination of the three trivial classifiers with the four given classifiers, i.e.:

$$h_a \cdot (1, 1, 0, 0, 0, 0) + h_b \cdot (0, 0, 1, 1, 0, 0) + h_c \cdot (0, 0, 0, 0, 1, 1)$$

$$+ h_1 \cdot (z_{ba}, z_{ca}, z_{ab}, z_{cb}, z_{ac}, z_{bc})$$

$$+ h_2 \cdot (w_{ba}, w_{ca}, w_{ab}, w_{cb}, w_{ac}, w_{bc})$$

$$+ h_3 \cdot (x_{ba}, x_{ca}, x_{ab}, x_{cb}, x_{ac}, x_{bc})$$

$$+ h_4 \cdot (y_{ba}, y_{ca}, y_{ab}, y_{cb}, y_{ac}, y_{bc})$$

And now we can discard when:

$$\exists h_a, h_b, h_c, h_d \in \mathrm{R}^+ \text{ where } (h_a + h_b + h_c + h_1 + h_2 + h_3 + h_4 = 1) \text{ such that:}$$

$$v_{ba} \geq h_a + 0 + 0 + h_1 \cdot z_{ba} + h_2 \cdot w_{ba} + h_3 \cdot x_{ba} + h_4 \cdot y_{ba}$$

$$v_{ca} \geq h_a + 0 + 0 + h_1 \cdot z_{ca} + h_2 \cdot w_{ca} + h_3 \cdot x_{ca} + h_4 \cdot y_{ca}.$$

$$v_{ab} \geq 0 + h_b + 0 + h_1 \cdot z_{ab} + h_2 \cdot w_{ab} + h_3 \cdot x_{ab} + h_4 \cdot y_{ab}$$

$$v_{cb} \geq 0 + h_b + 0 + h_1 \cdot z_{cb} + h_2 \cdot w_{cb} + h_3 \cdot x_{cb} + h_4 \cdot y_{cb}$$

$$v_{ac} \geq 0 + 0 + h_c + h_1 \cdot z_{ac} + h_2 \cdot w_{ac} + h_3 \cdot x_{ac} + h_4 \cdot y_{ac}$$

$$v_{bc} \geq 0 + 0 + h_c + h_1 \cdot z_{bc} + h_2 \cdot w_{bc} + h_3 \cdot x_{bc} + h_4 \cdot y_{bc}$$

This gives a system with 9+4 variables that can be solved by HSA, from which we again retain just 6 variables to obtain the polytope.

4.6 Maximum VUS for *n* classes

Let us obtain the maximum VUS for any number of classes.

Given a point (x_{12}, x_{13}, ..., x_{1c}, x_{21}, x_{23}, x_{24}, ..., x_{2c}, ..., x_{c1}, ..., $x_{c(c-1)}$), then we have that it is a classifier if and only if:

Valid Classifier Conditions

- x21 + x31 + x41 + ... + xc1 <= 1
- x12 + x32 + x42 + ... + xc2 <= 1
- ...
- x1c + x2c + x3c + ... + x(c-1)c + <= 1

It is easy to see that the volume of the space determined by these equations is:

$$\mathrm{VUS}^{\max} = \prod_c [\ \mathrm{P}(\Sigma_{c-1} \mathrm{U}(0,1) \leq 1)\ ] = \mathrm{P}(\Sigma_{c-1} \mathrm{U}(0,1) \leq 1)^{c}.$$

However, the probability that the sum of *c*-1 random numbers under the distribution U(0,1) is less than 1 is difficult to be obtained.

In particular the probability density function of the sum of *n* uniform variables on the interval [0,1] can be obtained using the characteristic function of the uniform distribution.

$$d_n(x) = F^{-1}\left[\left(\frac{i - \cos t + \sin t}{t}\right)^{n}\right] = \frac{1}{2(n-1)!}\sum_{k=0}^{n}(-1)^{k}\binom{n}{k}(x-k)^{n-1}\operatorname{sgn}(x-k)$$

And the cumulative distribution function is given by:

$$D_n(x) = \int_{-\infty}^{x} d_n(x)dx$$

Consequently, the probability that the sum of *n* random numbers under the distribution U(0,1) is less than 1 is:

$$D_n(1) = \lim_{x \to 1}\int_{-\infty}^{x} d_n(x)dx = \frac{1}{2(n-1)!}\int_{-\infty}^{0.9999...}\sum_{k=0}^{n}(-1)^{k}\binom{n}{k}(x-k)^{n-1}\operatorname{sgn}(x-k)dx$$

For *n*=1 we have $D_1(1)$= 1, for *n*=2 we have $D_2(1)$= ½, for *n*=3, $D_3(1)$= 1/6 and:

$$\mathrm{VUS}_{\mathrm{c}}^{\max} = \left(D_{c-1}(1)\right)^{c}$$

And hence we have, $\mathrm{VUS}_2^{\max}$= 1, $\mathrm{VUS}_3^{\max}$= 1/8 and $\mathrm{VUS}_4^{\max}$= 1/1296. However, for *n*>3 D_n is complex. For such cases, we can approximate the sum of *n* random numbers under the distribution U(0,1) with a single variable (*Y*) under the normal distribution with μ=*n*/2 and σ= *n*/12 using the central limit theorem. Then:

$$P(Y \leq 1) \approx P\left(\frac{Y - \frac{n}{2}}{\sqrt{\frac{n}{12}}} \leq \frac{1 - \frac{n}{2}}{\sqrt{\frac{n}{12}}} \right) = P\left(Z \leq \frac{1 - \frac{n}{2}}{\sqrt{\frac{n}{12}}} \right) = \Phi\left(\frac{1 - \frac{n}{2}}{\sqrt{\frac{n}{12}}} \right)$$

Where Z is a standard normal distribution variable. Therefore, when c>3:

$$\mathrm{VUS}_{\mathrm{c}}^{\max} \approx \left[\Phi\left(\frac{1 - \frac{c-1}{2}}{\sqrt{\frac{c-1}{12}}} \right) \right]^{C}$$

4.7 Minimum VUS for *n* classes

The exact minimum for 3 classes was obtained through HSA, but we did not arrive to a direct theoretical value. By observing the results for 4 classes, we have the following conjecture for more than *n* classes:

$$\mathrm{VUS}^{\min} = \frac{(c-1)^{(c-1)}}{d!}$$

For c=2 (d=2) gives ½, for c=3 (d=6) gives 4/720= 1/180, for c=4 (d=12) gives 27/479001600 = 1/17740800.

The points of the minimum polytope can be defined by the following conjecture:

- **The *c*×*c* classifiers that have (c-1) 1's in the same row.**

This conjecture matches for 4 classes.

The following table shows a summary of results for the general case (conjectures) and the exact results for 2, 3 and 4 classes.

vus	GENERAL	2C	3C (6d)			4C (12d)		
			Theoret.	Monte Carlo (1,000,000)	Exact (HSA)	Theoret.	Monte Carlo (100,000,000)	Exact (HSA)
max	[1/(c-1)!] ^{c} (conject)	1	1/8	0.12483	0.125	1/1296	0.000769	0.0007716
min	[(c-1]$^{(c-1)}$)/d! (conject)	1/2	1/180 (conject)	0.005523	0.0055556	27/12! (conject)	0.0000000697	0.0000000564

5 Evaluation of Multi-class Approximations to the VUS

In the previous section we have developed a method (conditions + HSA) to obtain the real VUS of any classifier for an arbitrary number of classes. However, this exact computation, although quite efficient for 3 and 4 classes, must be impractical for a higher number of classes or higher number of classifiers.

In the literature, there have been several approximations for the extension of the AUC measure for multi-class problems. However, there is no appraisal or estimation, either theoretical or practical, of how good they are.

In this section we gather and remind the approximations for the AUC for more than two classes known to date: macro-average, 1-point trivial AUC extension and some Hand & Till variants. We are going to make a comparison with the measures we have presented in this work: the exact VUS (through the HSA method).

We will give the definitions for three classes, although this can be easily extended to more classes. For the following definitions consider a 3-class classifier:

		Actual		
		a	*b*	*c*
Predicted	*a*	v_{aa}	$v_{ba}=x_1$	$v_{ca}=x_2$
	b	$v_{ab}=x_3$	v_{bb}	$v_{cb}=x_4$
	c	$v_{ac}=x_5$	$v_{bc}=x_6$	v_{cc}

5.1 Macro-average

The macro-average is just defined as the average of the partial accuracies of each class, i.e.:

$$\mathrm{MAVG}_3 = (v_{aa} + v_{bb} + v_{cc}) \;/\; 3$$

This measure has been used as a very simple way to handle more appropriately unbalanced datasets (without using ROC analysis).

5.2 Macro-average Modified

This measure employs the average and geometric means. Concretelly, it is defined as follows:

$$\mathrm{MAVG}_3\text{-MOD} = 0.75 \cdot (v_{aa} + v_{bb} + v_{cc}) + 0.25(\sqrt[3]{(v_{aa} \cdot v_{bb} \cdot v_{cc})})$$

We modify the original definition of macro-average because this does not consider the standard deviation between the points. For instance, using two classes, the point (0.2, 0.2) obtains a AUC than the point (0.1, 0.3) although both points have identical macro-average.

5.3 Other Measures MSE and LogLoss

The AUC measure tells "how well separated are the estimated distributions of $p(x)$ for class 0 and class 1" (Hand & Till 2001). Why do not we develop other measures that try to approximate how well separated two distributions are?

One measure of this kind is the well-known Mean Squared Error measure.

$$MSE = \frac{\sum_{i=1..m}\sum_{j=1..c}(f(i,j) - p(i,j))^2}{m \cdot c}$$

Where $f(i,j)$ is the actual probability of example i to be of class j and $p(i,j)$ is the estimated probability of example i to be of class j. The denominator gives a normalised MSE between 0 and 1. For classification problems, $f(i,j)$ will be always 0 or 1, depending on the class.

We have also considered other measures derived from the original MSE:

$$MCE = \frac{\sum_{i=1..m}\sum_{j=1..c}(f(i,j) - p(i,j))^3}{m \cdot c}$$

$$MQE = \frac{\sum_{i=1..m}\sum_{j=1..c}\sqrt{(f(i,j) - p(i,j))}}{m \cdot c}$$

$$MPE = \frac{\sum_{i=1..m}\sum_{j=1..c}(f(i,j)-p(i,j))}{m \cdot c}$$

Another measure is the log-loss, which is claimed to be a measure of the goodness of probability estimates (Bernardo & Smith 1993) (Mitchell 1997).

$$LogLoss = \frac{-\sum_{i=1..m}\sum_{j=1..c}(f(i,j)\log_2 p(i,j))}{m}$$

Note that these two measures are closely related to MAUC. As we have said, the MAUC measure tells "how well separated are the estimated distributions of $p(x)$ for class 0 and class 1" (Hand & Till 2001). This is quite the same of what is measured by the expression $(f(i,j)-p(i,j))^2$ or by the expression $(f(i,j)\log p(i,j))$.

These two measures have the advantage that are much easier to be understood and computed. Obviously, they have no problems of generalisation for more than 2 classes.

5.4 1-point Trivial AUC Extension

Going back to two classes, the area for one point (v_{ba}, v_{ab}) (in our representation) can be defined as:

$$\text{AUC}_2 = \max(1/2, 1 - v_{ba}/2 - v_{ab}/2)$$

Extending trivially the previous formula, we have this extension for 1-point extension for 3-class problems:

$$\text{AUC-1PT}_3 = \max(1/3, 1 - (v_{ba} + v_{ca} + v_{ab} + v_{cb} + v_{ac} + v_{bc})/3$$

It is easy to see that this extension is quite the same as the macro-average since the columns of the matrix sum to 1. The only difference is that the $1PT_3$ measure is never lower than 1/3.

5.5 1-point Hand and Till Extension

Hand and Till have presented a generalisation of the AUC measure [16] for soft classifiers, i.e., classifiers that assign a different score, reliability or probability with each prediction. Although we will deal with soft classifiers later, let us adapt Hand and Till's formulation for crisp classifier, i.e., classifier that predict one of the possible classes, without giving any additional information about the reliability or probability of the predicted class, or the other classes.

Hand and Till's extension for more than two classes is based on the idea that if we can compute the AUC for two classes *i,j* (let us denote this by *A*(*i,j*)), then we can compute an extension of AUC for any arbitrary number of classes by choosing all the possible pairs. Since *A(i,j)* = *A(j,i)*, this can be simplified as shown in the following Hand and Till's *M* function:

$$M = \frac{1}{c(c-1)}\sum_{i \neq j}\hat{A}(i,j) = \frac{2}{c(c-1)}\sum_{i<j}\hat{A}(i,j)$$

Pursuing this idea we are going to introduce three variants for 3 classes.

The first variant is given if we consider the macro-average extension. Then we have:

$$\text{HT1a} = (\max(1/2, (v_{aa} + v_{bb})/2) + \max(1/2, (v_{aa} + v_{cc})/2) + \max(1/2, (v_{bb} + v_{cc})/2)) / 3$$

This turns out to be similar to the 1PT, so we are going to take failures into account instead of hits and we have:

$$HT1b= (\max(1/2\,,\ 1–(v_{ba}+v_{ab})/2) + \max(1/2,\ 1–(v_{ca}+v_{ac})\,/2) + \max(1/2,\ 1–(v_{cb}+v_{bc})\,/2))\ /\ 3$$

This measure is slightly different from the previous ones.

Other different way to obtain this is if we normalise. For instance, if we normalise only for classes *a* and *b*:

		Actual	
		a	b
Predicted	a	v_{aa} / (v_{aa} + v_{ab})	x= v_{ba} / (v_{ba} + v_{bb})
	b	y= v_{ab} / (v_{aa} + v_{ab})	v_{bb} / (v_{ba} + v_{bb})

we have:

$$\max(1/2, (x+y)/2)$$

And this would be the same for the rest of combinations. Namely:

$$\begin{aligned} HT2= (&\max(1/2\,,\ 1 – (v_{ba}\ /\ (v_{ba} + v_{bb}) + v_{ab}\ /\ (v_{aa} + v_{ab}))/2) \\ &+ \max(1/2,\ 1 – (v_{ca}\ /\ (v_{ca} + v_{cc}) + v_{ac}\ /\ (v_{aa} + v_{ac}))\ /2) \\ &+ \max(1/2,\ 1 – (v_{cb}\ /\ (v_{cb} + v_{cc}) + v_{bc}\ /\ (v_{bb} + v_{bc}))\ /2))\ \ /\ 3 \end{aligned}$$

Finally, we can define a third variant that instead of computing partial AUCs of pairs of classes, computes the AUC of each class against the rest and then average the results.

For instance, the AUC of class *a* and the rest (*b* and *c* joined) will be obtained from a condensed 2x2 matrix:

		Actual	
		a	rest
Predicted	a	v_{aa} / (v_{aa} + v_{ab}+ v_{ac})	(v_{ba} + v_{ca}) / (v_{ba} + v_{ca} + v_{bb} + v_{bc} + v_{cb} + v_{cc})
	rest	(v_{ab}+ v_{ac}) / (v_{aa} + v_{ab}+ v_{ac})	(v_{bb} + v_{bc} + v_{cb} + v_{cc}) / (v_{ba} + v_{ca} + v_{bb} + v_{bc} + v_{cb} + v_{cc})

Using cells (a,rest) and (rest,a) we have:

$$AUC_{a,rest} = \max(1/2, 1 - [(v_{ab}+ v_{ac}) / (v_{aa} + v_{ab}+ v_{ac})]/2 - [\ (v_{ba} + v_{ca}) / (v_{ba} + v_{ca} + v_{bb} + v_{bc} + v_{cb} + v_{cc})]/2$$

In the same way we can obtain $AUC_{a,rest}$ and $AUC_{a,rest}$. This allows us to define HT3:

$$HT3= AUC_{a,rest} + AUC_{a,rest}\ + AUC_{a,rest}\ /\ 3$$

5.6 Monte Carlo Estimation

As we saw in previous sections, the Monte Carlo method is quite exact for obtaining the maximum and minimum VUS. However, for an arbitrary classifier we had the condition:

$$\exists h_a,h_b,h_c,h_d \in R^+ \text{ where } (h_a + h_b + h_c + h_d = 1) \text{ such that:}$$

$$\begin{aligned} v_{ba} &\geq h_a + 0 + 0 + h_d \cdot z_{ba} \\ v_{ca} &\geq h_a + 0 + 0 + h_d \cdot z_{ca} \\ v_{ab} &\geq 0 + h_b + 0 + h_d \cdot z_{ab} \\ v_{cb} &\geq 0 + h_b + 0 + h_d \cdot z_{cb} \\ v_{ac} &\geq 0 + 0 + h_c + h_d \cdot z_{ac} \\ v_{bc} &\geq 0 + 0 + h_c\ + h_d \cdot z_{bc} \end{aligned}$$

This condition obliges to incorporate a new loop into the Monte Carlo method to "emulate" the existential quantifier (∃). Consequently, we have now an outer loop (number of instances) and an inner loop (which generates possible values for h_a, h_b, h_c, h_d according to a uniform distribution between 0 and 1, where $h_a + h_b + h_c + h_d = 1$). This makes it very slow to obtain good approximations. In any case, we will also compare (when possible) the Monte Carlo Estimation with an outer loop of 100,000 iterations and an inner loop of 500,000 iterations.

5.7 Experimental Evaluation

Once the previous approximations are presented, we are ready to evaluate them in comparison to the exact computation given by the HSA method.

First of all, we are going to examine the results of the methods for some arbitrary points, as is shown in the following table:

#	classifier point	MONTE CARLO	exact (hSA)	Macro-avg	1-p trivial	HT1B	HT2	HT3
1	(0, 0, 0, 0, 0, 0)	0.125...	0.125	1	1	1	1	1
2	(1, 1, 1, 0, 0, 0)	0.0055...	0.0055556	0	0.333	0.667	0.5	0.5
3	(1, 1, 0, 0, 0, 0)	0.0055...	0.0055556	0.333	0.333	0.667	0.5	0.5
4	(0.5, 0.5, 0, 0, 0, 0)	0.0327.	0.0342	0.666	0.666	0.833	0.833	0.75
5	(0.5, 0, 0.5, 0, 0, 0)	0.0291	0.03105	0.666	0.666	0.833	0.833	0.75
6	(0.75, 0.75, 0.75, 0, 0, 0)	0.0076	0.0083279	0.25	0.333	0.7083	0.7083	0.54167
7	(0.33, 0, 0.33, 0, 0, 0)	0.052186	0.05221362	0.777	0.777	0.8888	0.8888	0.8333
8	(0.1, 0.2, 0.05, 0.15, 0.25, 0.3)	0.029	COMPLETE	0.65	0.65	0.825	0.7952	0.7375
9	(0.1, 0.2, 0.05, 0.1, 0.25, 0.3)	0.0325	COMPLETE	0.667	0.667	0.833	0.80779	0.75
10	(0.1, 0.2, 0, 0.2, 0.25, 0.3)	0.0340	COMPLETE	0.683	0.683	0.84167	0.82110	0.7625

We can notice that if we take HSA and Monte Carlo as reference (as they behave quite similarly) we have the following order: 1, 7, 10, 4, 9, 5, 8, 6, 2-3. On the other hand, the five other approximations do not give the same ranking (specially because of the part 10, 4, 9, 5 in which some classifier are not discriminated or even the order is swapped). As expected, the approximations fail in some cases.

To evaluate which approximation is best, we are going to use the following experimental methodology:

- Let n= 100; // size of the experiment
- Initialise Matrices HSA[n×n], Monte[nxn], MAVG[n×n], AUC-1PT[n×n].
- Initialise Matrices HT1[n×n], HT2[n×n], HT3[n×n].
- We generate n valid classifiers randomly C[1..n].
    - The generation of each classifier is performed in the following way:
    - a= RandUniform(0, 1);
    - x3= RandUniform(0, 1-a);
    - x5= 1 – x3 – v;
    - b= RandUniform(0, 1);
    - x6= RandUniform(0, 1-b);
    - x1= 1 – x1 – b;
    - c= RandUniform(0, 1);

```
        o  x2= RandUniform(0, 1-c);
        o  x4= 1 – x2 – c;
•  FOR i=1 to n DO
        o  Compute exact VUS through HSA of C[i].
        o  Do the same for all the other methods...
•  ENDFOR
•  FOR i=1 to n DO
        o  FOR j=1 to n DO
                ▪  IF (i<j) THEN
                        •  Mark HSA[i,j] as   0  if the value for C[i] = C[j]
                                              -1 if the value for C[i] < C[j]
                                              1   if the value for C[i] > C[j]
                        •  Do the same for all the other methods...
                ▪  ENDIF
        o  ENDFOR
•  ENDFOR
```

Then we compare the matrices in the following way; given the matrices M_1 and M_2 of two different methods, we compare the discrepancy as:

$$disc = \frac{2\sum_{i=1}^{n} \sum_{j=1,i<j}^{n} \left| M_1(i,j) - M_2(i,j) \right|}{n(n-1)}$$

With this formula we can evaluate the discrepancy of the methods for 3 class problems with respect the real VUS computed with the HAS method.

The results are:

Accuracy	Macro-avg	Modif-Macro	1-p trivial	HT1B	HT2	HT3	MSE	MCE	MPE	MQE
0,08707	0,087071	0,06	0,09131	0,10404	0,14081	0,09677	0,27333	0,29434	0,25111	0,31172

Discrepancies between methods.

According to these results, the best approximation is the macro-average modified. This measure obtains the lower discrepancy among the studied approximations.

6 Computing the Real VUS for Soft Classifiers

Up to now, we have computed the VUS for one point or for a set of points that represent crisp classifiers, i.e., classifiers without prediction probabilities or scores. Nonetheless, it is increasingly more usual that machine learning methods are devised to obtain soft classifiers, i.e., classifiers that accompany each prediction with the reliability or, even better, with the estimated probabilities of each class.

For the two-class case a soft classifier can be used to construct infinite many classifiers as one is changing the predictions by using a threshold. The good thing about the two-class case is that given *n* predictions with different estimated probability vector, only *n*+1 classifiers have to be considered. However, this is not so for soft classifiers with more than two classes.

For instance, given a soft classifier of *c* classes, we can generate c^n classifiers, with *n* being the different probability vectors that it gives. In the case of a decision tree learner, e.g, *n* would be the number of leaf nodes of the tree, in other cases *n* may be number of examples.

For instance given a 3-class decision tree classifier with 10 nodes:

Node	ESTIMATED CLASS PROB. VECTOR			CRISP CLASSIFIER												
	a	b	c	Maj. Class	C_1	C_2	C_3	C_4							...	C_{59049}
1	0.3	0.6	0.1	b	a	a	a	a	a	a	a	a	a	a	...	c
2	0.5	0.25	0.25	a	a	a	a	a	a	a	a	a	a	a	...	c
3	0.2	0.4	0.4	b	a	a	a	a	a	a	a	a	a	a	...	c
4	0.1	0.2	0.7	c	a	a	a	a	a	a	a	a	a	a	...	c
5	0.8	0.1	0.1	a	a	a	a	a	a	a	a	a	a	a	...	c
6	0.6	0.3	0.1	a	a	a	a	a	a	a	a	a	a	a	...	c
7	0.2	0.6	0.2	b	a	a	a	a	a	a	a	a	a	a	...	c
8	0.1	0.3	0.6	c	a	a	a	a	a	a	a	a	a	b	...	c
9	0.3	0.3	0.4	c	a	a	a	b	b	b	c	c	c	a		c
10	0.7	0.2	0.1	a	a	b	c	a	b	c	a	b	c	a		c

We can generate $3^{10} = 59049$ classifiers that will form a polytope.

Theoretically, we can obtain the polytope using our method, by solving the system given by the c^n classifiers (the three trivial classifiers are included among them). In practice, however, this seems quite unfeasible for great values of *c* or *n*.

An open question is to see whether some assignments can be discarded in order to reduce complexity. Some preliminary ideas on this can be found in the Appendix B.

7 Approximations to the Area Under the ROC Curve for More than Two Classes for Soft Classifiers

Computing the real VUS for one point or a small set of points seems quite feasible in practice. However, the same is not true for soft classifiers, as we have seen, if we want to consider all the possible assignments.

It is now then when we are more interested in approximations and to perform an evaluation of how good these approximations are.

7.1 Hand and Till *M* Function

Hand and Till present a generalisation of a particular AUC measure [16]. It has been shown that for two dimensions the AUC measure is equivalent to the GINI measure (not that the GINI measure is not the GINI splitting criterion used in the CART algorithm).

The idea is that in the AUC measure for 2 dimensions, they use the estimated probabilities of an example x_i pertaining to the class 0, denoted by $p_0(\cdot)$, (estimated from the training set), to rank the pairs $\{g_k, f_k\}$ where g_k and f_k are defined as $g_i = p_0\ (x1_i)$ and $f_j = p_0\ (x0_j)$ where $x1_i$ are the examples from the test set of class 1 and $x0_j$ are the examples from the test set of class 0. Note that instead of ordering nodes they order examples.

For instance consider the following two nodes for the training set:

Node 1: (4,1) --> class 0 with prob= 4/5 = 0.8

Node 2: (2,3) --> class 1 with prob= 2/5 = 0.4

And now consider that the test set is distributed in the following way over the decision tree:

Node 1: (6,4)

Node 2: (4,11)

with n_0= 10 elements of class 0 and n_1= 15 elements of class 1. Then we have:

- 6 of class 0 with $p_0(\cdot)$ = 0.8
- 4 of class 1 with $p_0(\cdot)$ = 0.8
- 4 of class 0 with $p_0(\cdot)$ = 0.4
- 11 of class 0 with $p_0(\cdot)$ = 0.4

From here, we can rank them as described in [16]. Let us denote with r_i the rank of the i^{th} class 0 test set point. Let us denote $S_0 = \Sigma r_i$. The they derive the area as:

$$\hat{A} = \frac{S_0 - n_0(n_0+1)/2}{n_0 n_1}$$

This area, although is an AUC (and it has the equivalence Gini + 1 = 2 × A), has two main differences with respect to usual ROC curves and also to a similar proposal in [14]:

- It is a step-like (or a stairs-like) area (no diagonals between the points are computed)).
- It is not convex, because the order is given by the training set and the examples are given by the test set.

Apart from this, the most relevant novelty of Hand and Till paper is that they understand *A* as "an overall measure of how well separated are the estimated distributions of $p_0(\cdot)$ for class 0 and class 1", i.e., A(i,j) could be computed for whatever pair of classes *i* and *j*.

This interpretation allows what they call "a simple generalisation of the AUC for multiple class classification problems". They define a new measure M as:

$$M = \frac{1}{c(c-1)} \sum_{i \neq j} \hat{A}(i,j) = \frac{2}{c(c-1)} \sum_{i<j} \hat{A}(i,j)$$

7.2 Fawcett's Extension

This extension, presented in [13], is similar to Hand and Till's, but it includes a probability. Its definition is as follows:

$$AUCF = \sum_c p(i) \cdot \hat{A}(i, rest)$$

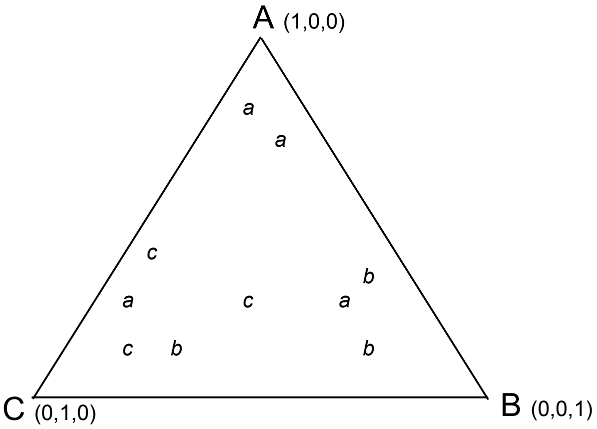
where A(i, rest) computes the AUC of one class against the other, i.e., each times considers all the other classes as one. This seems to be an optimisation of Hand and Till's M function. The main difference lies in the fact that AUCF includes the probability of each class as a factor and this might make AUCF less independent of class probabilities and biased datasets.

7.3 Mossman Trichotomous ROC Analysis

Mossman [27][9] has presented an extension for three classes that is based on a priorisation of one of them. In this way, the problem is simplified to an approximate problem with as many dimensions as classes. For the 3-class case, it even permits a graphical representation.

First of all, Mossman presents a very interesting representation of three class soft classifiers. Consider a soft classifier such that for each example it provides three estimated probabilities for each class p_a, p_b and p_c. If we consider that $p_a + p_b + p_c = 1$, then we can plot these vectors on a triangular "estimate plane", labelling each point with the true class that a validation or a test set has given.

For instance, the following picture shows 10 examples for which a classifier gives different class vector estimations:

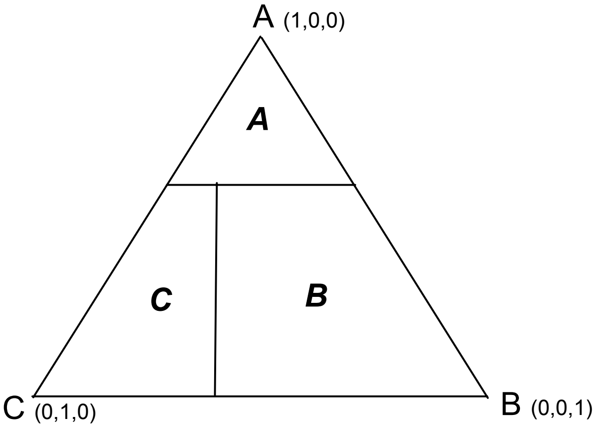


Although this is an interesting graph for 3 classes, the most remarkable proposal from Mossman is the idea of priorising one class under the following decision rule:

DECISION RULE I:

1) if (p_a ≥ α) then classify *a*
2) else if (p_b - p_c ≥ β) then classify *b*
3) else classify *c*.

For instance, if we fix α=0.6 and β= 0.2 we would have the following areas of the triangle:



The Mossman 3-dimensional plot is just based on taking the diagonal of the confusion matrix and completely ignoring the misclassifcations a->b, a->c, b->a, b->c, c->a, c->b. In this way it gives a different classifier and, consequently, a different 3D point for each value of α and β that we choose. Choosing infinite many different α and β, we have a 3D surface, and we can compute the Volume Under the Surface (VUS).

Computing this volume gives VUS1, a different extension for AUC for more than two classes. Depending on the priorities of the classes we have different VUS1ab, VUS1ac, VUS1ba, VUS1bc, VUS1ca, VUS1cb. We could define VUS1 as the mean of all them.

Maybe the most interesting thing of Mossman's paper is the interpretation of the AUC as a two-alternative forced-choice decision task, where two examples (one of each class) must be correctly classified:

1) Classify as "a" the one that has greater p_a.
2) Classify the other one as "b".

Mossman extends this to 3-class VUS, by equalling VUS to the probability that three examples (one of each class) would be correctly classified, according to the decision rule that we choose. **This is a three-alternative forced-choice decision task**. For instance if we use decision rule I, we could approximate this probability (just by selecting triplets of examples of three different classes), obtain the class vectors using the soft classifier and then applying the decision rule. This gives a probabilistic interpretation of VUS1 for any particular α and β, and for a wide range of different α and β.

But, additionally, we can change the decision rule. For instance, Mossman proposes a second decision rule:

DECISION RULE II:

1) Classify as "a" the one that has greater p_a.
2) Of the remaining two, classify as "b" the one that has greater p_b.
3) The last one is classified as "c".

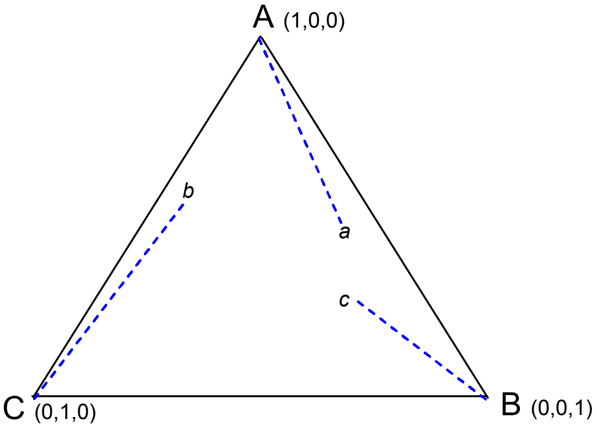
Depending on the variant of this rule we have more VUS: VUS2ab, VUS2ac, VUS2ba, VUS2bc, VUS2ca and VUS2cb. We could define VUS2 as the mean of all them.

Finally, Mossman also presented a third decision rule;

DECISION RULE III:

1) Plot the three points in the triangular estimate plane.
2) Assign each of the tree points to one class in such a way that the total length of the three segments is minimised.

This rule is illustrated in the following figure:



Note that this third decision rule gives another interpretation of VUS3, in this case is not graphical, but as a probability. In our opinion, this is the most reasonable one.

All these approximations could be evaluated. Nonetheless, some of them are not easy to be extended to more than three classes, especially VUS1 and VUS3 variants.

Consequently, in what follows, we will refer to VUS2.

7.4 Evaluation of Approximations

We have performed an experimental evaluation of the previous approximations in a similar way as was done in section 5.7 for crisp classifiers. In ROC analysis, there are some techniques such as stratification to generate a set of points from soft classifiers in order to compute a curve from the classifier.

Due to the limitations on the number of equations of the HAS method, we cannot easily apply this method over soft classifiers. Instead of this, we simulate a soft classifier by generating 4 points (in the way explained in Section 5.7) in the ROC space. Then, we compute the exact VUS with the HAS method. For the rest of approaches, we compute the VUS of each independent point, and the final VUS as the average of all the points. Finally, we construct a discrepancy matrix for these 100 different simulated soft classifiers. According to the matrices, the discrepancy of the approaches with respect to the exact VUS computed with the HAS method is:

Accuracy	Macro-avg	1-p trivial	HT1B	HT2	HT3	MSE	MCE	MPE	MQE
0,382222	0,373737	0,309091	0,346263	0,329293	0,280404	0,551919	0,642828	0,41899	0,373737

Discrepancies between methods.

Following these results, the best approximation is the HT3 approach. Note that the discrepancy in this case is much higher. We think that this is mainly owing to the employment of the average of the partial VUS to compute estimation. Probably, other strategies (more sophisticated) could help to improve the approaches.

7.5 Scattering factor

The VUS of a set of points is closely related to the space "scattering" of such points. Then, one way to estimate the VUS could be to compute how much scattered the points are in the space. Based on this idea we define a new *Scatt* factor defined as follows:

Given a set *n* points, of *d* dimensions, For each dimension, the *n* coordinates are ranked:

$p_1, p_2, \ldots, p_n,$

Then we compute the partial "dissemination" as:

$M_i = (p_1 + p_2 + \ldots + p_n)/n$, $V_i = ((p_n - p_{n-1}) * (p_{n-1} - p_{n-2}) * \ldots (p_2 - p_1))^{1/(n-1)} / M_i$, $1 \leq i \leq d$

The global dissemination is computed as the average for all the dimensions.

Scatt$=(M_1+M_2+..+M_d)/d$

Finally, we employ this factor in the final estimation as follows:

$VUS=(H_1+H_2+..+Hn)/n * (Scatt)^{exp}$

Where H is an approximation of the VUS for one point, *exp* is a correction value of the scattering factor. According to our experiments, 0.1 is the value where the best performance is obtained

We have applied this correction using the HT3 approach, and the discrepancy of the matrix computed with the same methodology as before with respect to the HAS method is:

HT3 + *Scatt*
0,259394

Therefore, the application of the scattering correction reduces 0,02 points approximately the discrepancy of HT3 with regard to the HAS method.

8 Conclusions

In this paper we have highlighted the limitations of current approaches for problems with more than two classes and proposing some extensions and alternatives.

We have identified the trivial classifiers and have neglected direct but wrong ways to extend the 2-class AUC into the general VUS. We have then derived the discard conditions, identified the maximum and minimum VUS and their polytopes, as well as the VUS for any arbitrary set of crisp classifiers. This is computed through the HSA algorithm. We have then compared experimentally the real VUS with several other approximations for crisp classifiers, showing which approximation is best. The best approximation seems to be HT3, which is a simplification of Hand & Till's *M* function for crisp classifiers.

For soft classifiers, we have shown how to compute the polytope form by all the possible assignments, although we have recognised the limitations of computing it in practice. In this case we have performed a limited comparison showing that the best approximation for soft classifiers is also HT3. In this case, we have introduced a correction based on the dissemination of the points in the space that allows to improve the HT3 approximation.

As future work, we would like to work further on the soft classifier case, trying to find out a way to simplify the set of c^n points in order to make a better evaluation or even derive a method to compute the real VUS for soft classifiers in a reasonable time. We also would like to prove some of the conjectures we have introduced in this paper.

References

1. Adams, N.M. and Hand, D.J. "Comparing classifiers when the misallocation costs are uncertain, *Pattern Recognition*, Vol. 32 (7) (1999) pp. 1139-1147.
2. Barber, C.B.; Huhdanpaa, H. "QHull", The Geometry Center, University of Minnesota, http://www.geom.umn.edu/software/qhull/.
3. Bernardo, J.M. & Smith, A.F. *Bayesian Theory*, John Wiley & Sons, 1993.
4. Blockeel, H.; Struyf, J. "Frankestein classifiers: Some experiments on the Sisyphus data set", ECML/PKDD2001 Workshop on *Integrating Aspects of Data Mining, Decision Support and Meta-Learning*, 2001.
5. Boissonat, J.D.; Yvinec, M. *Algorithmic Geometry*. Cambridge University Press, 1998.
6. Bradford, J.; Kunz, C.; Kohavi, R.; Brunk, C; and Brodley, C. "Pruning decision trees with misclassification costs" in *Proc. of the European Conference on Machine Learning*, pp. 131-136, 1998.
7. Breiman, L.; Friedman, J.H.; Olshen, R.A. and Stone, C.J. *Classification and regression trees*, Belmont, CA, Wadsworth, 1984.
8. Domingos, P. "Metacost: A general method for making classifiers cost-sensitive" *Proc. of the Fifth International Conference on Knowledge Discovery and Data Mining*, pp. 155-164, New York, ACM, 1999.
9. Dreisetl, S.; Ohno-Machado, L. ; Binder, M. "Comparing Trichotomous Tests by Three-Way ROC Analysis" *Medical Decision Making*, 1999.
10. Drummond, C.; Holte, R.C. "Exploiting the cost (in)sensitivity of decision tree splitting criteria", *Proc. of the Seventeenth International Conference on Machine Learning, pp. 239-246,* 2000.

11. Elkan, C. "The Foundations of Cost-Sensitive Learning", *Proc. of the Seventeenth International Joint Conference on Artificial Intelligence*, IJCAI'01, 2001.
12. Fan, W.; Stolfo, S.; Zhang, J. and Chan, P.H. "AdaCost: Misclassification Cost-sensitive Learning" Proceedings of the Sixteenth International Conference on Machine Learning (ICML'99), pp.97-105, Bled, Slovenia, June 1999.
13. Fawcett, T. "Using Rule Sets to Maximize ROC Performance" presented at the *2001 IEEE International Conference on Data Mining* (ICDM-01) .
14. Ferri-Ramírez, C.; Flach, P. Hernández-Orallo "Rocking the ROC Analysis with Decision Trees" Technical Report, Dep. of Computer Science, University of Bristol, 2001.
15. Hand, D.J. *Construction and assessment of classification rules.* Chichester: Wiley, 1997.
16. Hand, D.J.; Till, R.J. "A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems" *Machine Learning*, 45, 171-186, 2001.
17. Hanley, J.A.; McNeil, B.J. "The meaning and use of the area under a receiver operating characteristic (ROC) curve" *Radiology*. 1982: 143:29-36.
18. Kearns, M. and Mansour, Y. "On the boosting ability of top-down decision tree learning algorithms" *Proceedings of the Twenty-Eighth ACM Symposium on the Theory of Computing*, pp. 459-468, New York, ACM Press, 1996.
19. Knoll, U.; Nakhaeizadeh, G.; Tausend, B. "Cost-sensitive pruning of decision trees" *Proc. of the Eight European Conference on Machine Learning*, ECML-94, pp. 383-386, Berlin, Germany, Springer-Verlag, 1994.
20. Kukar, M.; Kononenko, I. "Cost-sensitive learning with neural networks" *Proc. of the Thirteenth Conference on Artificial Intelligence*, Chichester, NY, Wiley, 1998.
21. Lane, T. "Extensions of ROC Analysis to Multi-Class Domains", ICML-2000 Workshop on cost-sensitive learning, 2000.
22. Lavrac, N., Gamberger, D. and Turney, P. "Cost-sensitive feature reduction applied to a hybrid genetic algorithm." In Proc. Seventh International Workshop on Algorithmic Learning Theory, pp. 127-134, Springer, Berlin, 1996.
23. Margineantu, D.; Dietterich, T.G. "Learning Decision Trees for Loss Minimization in Multi-Class Problems", Technical Report 99-30-03, Department of Computer Science, Oregon State University, 1999.
24. Margineantu, D.; Dietterich, T.G. "Bootstrap Methods for the Cost-Sensitive Evaluation of Classifiers", Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000), pp.583-590, Morgan Kaufmann, San Francisco, CA, 2000.
25. McClish, D.K. "Comparing the areas under more than two independent ROC curves" *Med. Decis. Making*, 1987; 7:149-55.
26. Mitchell, T. *Machine Learning,* McGraw Hill, 1997.
27. Mossman, D. "Three-way ROCs" *Medical Decision Making*, 1999, 19: 78-89.
28. Mossman, D.; Somoza, E. "ROC curves, test accuracy, and the description of diagnostic tests" *J. Neuropsychiatr. Clin. Neurosci.* 1991, 3: 330-3.
29. Pazzani, M. J., Merz, C. J., Murphy, P., Ali, K., Hume, T., and Brunk, C. "Reducing Misclassification Costs" In Proceedings of the 11th International Conference of Machine Learning, Morgan Kaufmann, 217-225, 1994.
30. Provost, F.J. "Goal-directed inductive learning: Trading off accuracy for reduced error cost" *AAAI Spring Symposium on Goal-Driven Learning*, 1994.
31. Provost, F. and Fawcett, T. "Analysis and visualization of classifier performance: Comparison under imprecise class and cost distribution" in *Proc. of The Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, pp. 43-48, Menlo Park, CA: AAAI Press, 1997.
32. Provost, F.; Fawcett, T. and Kohavi, R. "The Case Against Accuracy Estimation for Comparing Induction Algorithms" presented at *ICML-98 (Fifteenth International Conference on Machine* Learning), July 1998.
33. Provost, F.; Fawcett, T. "Robust Classification for Imprecise Environments" Machine Learning 42, 203-231, 2001
34. Quinlan, J.R. *C4.5. Programs for Machine Learning*, San Francisco, Morgan Kaufmann, 1993.
35. Quinlan, J.R. "Improved use of continuous attributes in C4.5" *Journal of Artificial Intelligence Research*, 4, 77-90, 1996.

36. Salido, M.A.; Barber, F. “An Incremental and Non-binary CSP Solver: The Hyperpolyhedron Search Algorithm (Short Version)” in Proceedings of Seventh International Conference on Principles and Practice of Constraint Programming CP 2001. Springer Verlag, LNCS 2239, pp: 779-780, 2001.
37. Salido, M.A.; Giret, A. Barber, F. “Constraint Satisfaction by means of Dynamic Polyhedra” in Operations Research Proceedings 2001. Springer Verlag, ISBN: 3-540-43344-9, pp: 405-412, 2002.
38. Smith, S. P., and Jain, A.K. “Testing for Uniformity in Multidimensional Data” IEEE Trans. on Pattern Analysis and Machine Intelligence, 6 (1984), pp. 73-81.
39. Srinivasan, A. “Note on the Location of Optimal Classifiers in N-dimensional ROC Space” Technical Report PRG-TR-2-99, Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford.
40. Swets, J., Dawes, R., and Monahan, J. “Better decisions through science” *Scientific American*, October 2000, 82-87.
41. Turney, P.D. “Cost-Sensitive Classification: Empirical Evaluation of a Hybrid Genetic Decision Tree Induction Algorithm”, *Journal of Artificial Intelligence Research* 2, pp. 369-409, 1995.
42. Turney, P. “Types of Cost in Inductive Concept Learning” Proceedings Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning (WCSL at ICML-2000), 15-21,2000.
43. University of California, UCI Machine Learning Repository Content Summary, http://www.ics.uci.edu/~mlearn/MLSummary.html.
44. Weiss, G. and Provost, F. "The Effect of Class Distribution on Classifier Learning: An Empirical Study" Technical Report ML-TR-44, Department of Computer Science, Rutgers University, 2001.
45. Zweig, M.H.; Campbell, G. “Receiver-operating characteristic (ROC) plots: a fundamental evaluation tool in clinical medicine”, *Clin. Chem*, 1993; 39: 561-77.

APPENDICES

9 APPENDIX A. Graphical Alternatives for ROC Analysis with more than 2 classes

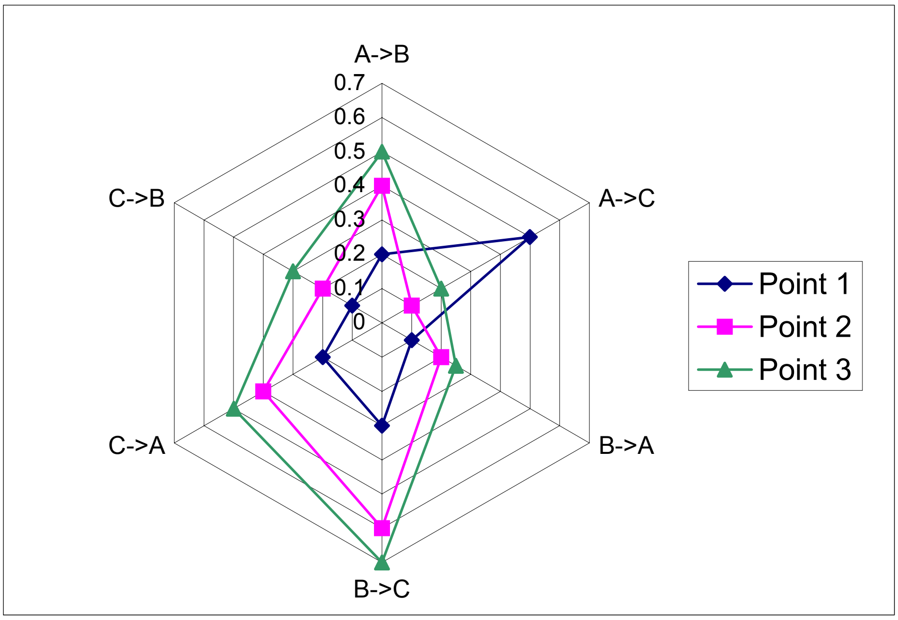
9.1 Cobweb Representation

Given several *d*-dimensional points on the ROC space corresponding to classifiers, we can draw them in a cobweb representation.

For instance, given three classifiers with these confusion ratio matrices:

		Point 1			Point 2			Point 3		
		Actual			Actual			Actual		
		A	B	C	A	B	C	A	B	C
	A	0.3	0.1	0.2	0.5	0.2	0.4	0.2	0.25	0.5
Predicted	B	0.2	0.6	0.1	0.4	0.2	0.2	0.5	0.05	0.3
	C	0.5	0.3	0.7	0.1	0.6	0.4	0.2	0.7	0.2

We can obtain d= (c·(c-1)) values from each classifier, in this case 6= 3·2, i.e. a point in a 6D space. The values of the different dimensions of a point are formed by the values in the confusion ratio matrix, ignoring the diagonal. This can be represented graphically using a cobweb representation as follows.



Note that point 3 can be discarded, because point 2 has all the coordinates below it.

The previous representation allows determining where each classifier has strong and weak points and, in simple cases, some of them can be discarded.

The previous representation, though, has two disadvantages:

- For many classifiers the representation gets highly confusing because the classifiers get very close.
- The convex hull cannot be drawn or recognised and consequently only classifiers where all the misclassification coordinates are below all the coordinates of another classifier can be discarded.

Moreover, it (graphically) ignores that each pair of classifiers can be combined to produce infinitely many classifiers stochastically combining their predictions. This is precisely why the convex hull has to be computed.

9.2 Lattice Representation

Another partially graphical representation, quite related to the previous one, is based on defining a partial order between classifiers, defined in the following way:

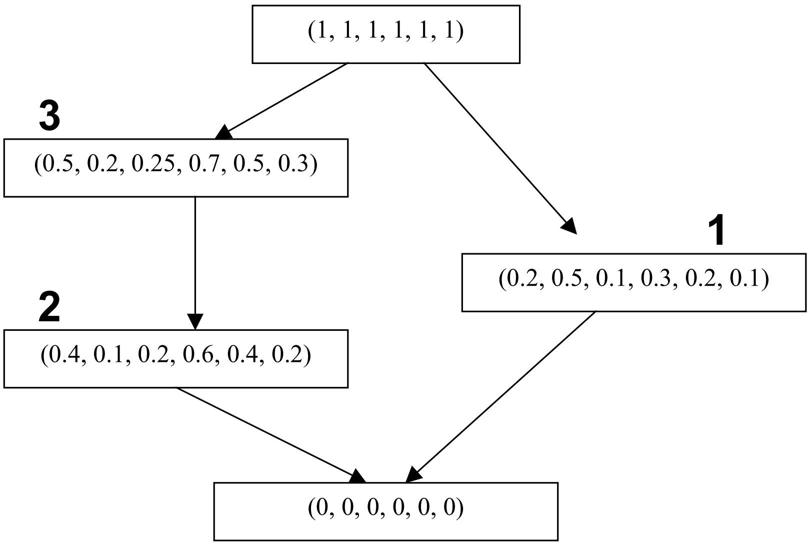
Given two d-dimensional points p and q,

$$p \leq q \text{ iff } \forall \text{i}: 1 \leq i \leq d\ ,\ p_i \leq q_i$$

If we just consider points from the confusion ratio matrices, then we have that for any point p, $\forall$i: $1 \leq i \leq d$, $0 \leq p_i \leq 1$.

Consequently, given a set of classifiers, if we always add the points (1,1,...,1) and (0,0,...,0) then we can construct a lattice.

For instance, for the previous example, we would have the following lattice:



The point C can be discarded because has other point below in the lattice (except 0,0,0,0,0,0)

This representation, as well as the cobweb representation, has two disadvantages:

- For many points the representation gets very large.
- The convex hull cannot be recognised and consequently only points where all the misclassification coordinates are below all the coordinates of another point can be discarded.

This representation and the cobweb representation are equivalent.

10 APPENDIX B. ARE ALL THE CLASS ASSIGNMENTS NEEDED?

We know that in the 2-class case, instead of the 2^n possible assignments, only n+1 assignments are strictly necessary (supposing the test set will behave similarly as the train set). However, can we do a similar reduction for more than 2 classes?

10.1 Class Assignment

If there is a cost matrix (derived from weights or provided by the user), the class of a node or rule should not be computed as the majority class, but as the class that minimises the cost of the examples that fall under that node using the current cost matrix.

More concretely, if we have $Cost_i$ as the cost if class i would be selected:

$$Cost_i = \sum_j n(j) \cdot C(i,j)$$

Where $n(j)$ is the number of training examples of class j.

Then the assigned class will be:

$$AssignedClass = \arg\min_i Cost_i$$

Apparently, there is no difference between two or more than two dimensions. However, some "strange assignments" can be given for more than two classes.

For instance, for two classes, if one node or rule has no elements of one class, it is supposed that it will not be selected, supposing that the cost matrix is reasonable, in the sense described in section 2.2.1. Let us see that this is not the case for more than two dimensions.

Let us just study the behaviour of a node/rule in a 3-class problem depending on a cost matrix. In this context we have a cost matrix:

		Actual		
		a	b	c
Predicted	a	0	A_2	A_3
	b	B_1	0	B_3
	c	C_1	C_2	0

Where all the costs are greater than 0. And the leaf is :

E_A	E_B	E_C

The leaf will be assigned A iff:

$$\begin{aligned} E_B \cdot A_2 + E_C \cdot A_3 &\le E_A \cdot B_1 + E_C \cdot B_3 \\ E_B \cdot A_2 + E_C \cdot A_3 &\le E_A \cdot C_1 + E_B \cdot C_2 \end{aligned}, \text{ and}$$

The leaf will be assigned B iff:

$$\begin{aligned} E_A \cdot B_1 + E_C \cdot B_3 &\le E_B \cdot A_2 + E_C \cdot A_3 \\ E_A \cdot B_1 + E_C \cdot B_3 &\le E_A \cdot C_1 + E_B \cdot C_2 \end{aligned}, \text{ and}$$

The leaf will be assigned C iff:

$$E_B \cdot C_2 + E_C \cdot C_3 \le E_B \cdot A_2 + E_C \cdot A_3$$
$$E_B \cdot C_2 + E_C \cdot C_3 \le E_A \cdot B_1 + E_C \cdot B_3 \text{, and}$$

We can express this problem more shortly, by defining the following parameters:

$$X_A = E_B \cdot A_2 + E_C \cdot A_3$$
$$X_B = E_A \cdot B_1 + E_C \cdot B_3$$
$$X_C = E_A \cdot C_1 + E_B \cdot C_2$$

As the following easy choice:

```
If $X_A \le X_B$ then
            If $X_A \le X_C$ then Class=A
            else             Class=C
else
            If $X_B \le X_C$ then Class=B
            else             Class=C
```

Let us suppose that there are no elements of one class, let's say A, then E_A=0. Now, we can show that a node will be assigned A iff:

$$E_B \cdot A_2 + E_C \cdot A_3 \le E_C \cdot B_3$$
$$E_B \cdot A_2 + E_C \cdot A_3 \le E_B \cdot C_2 \text{, and}$$

i.e.

$$E_C \cdot A_3 - E_C \cdot B_3 \le -E_B \cdot A_2$$
$$E_B \cdot A_2 - E_B \cdot C_2 \le -E_C \cdot A_3 \text{, and}$$

i.e.

$$\frac{A_3 - B_3}{A_2} \le \frac{-E_B}{E_C}$$
$$\frac{A_2 - C_2}{A_3} \le \frac{-E_C}{E_B} \text{,and}$$

Consequently, the node *will* be assigned A iff

$$\frac{B_3 - A_3}{A_2} \ge \frac{E_B}{E_C}$$
$$\frac{C_2 - A_2}{A_3} \ge \frac{E_C}{E_B} \text{,and}$$

Imagine we have this situation: $E_C=E_B=A_2=A_3=1$, and iff $C_2 \ge 2$ and $B_3 \ge 2$, then the node will be assigned A.

So it can happen that even without having no element of class A in one node, that node could be assigned class A.

The issue is that just in the case where E_B=0, E_C=0, then we can say that the leaf will be always assigned A and never to B or C. The proof is obvious, in this case it will be assigned to A iff:

$$E_B \cdot A_2 + E_C \cdot A_3 \le E_A \cdot B_1 + E_C \cdot B_3$$
$$E_B \cdot A_2 + E_C \cdot A_3 \le E_A \cdot C_1 + E_B \cdot C_2$$
, and

i.e.

$$0 \le E_A \cdot B_1$$
$$0 \le E_A \cdot C_1$$
, and

Since $E_A \geq 0$, $B_1 \geq 0$, $C_1 \geq 0$ it is true and that means that it is always assigned to A.