

Determining the Fitness of a Document Model by Using Conflict Instances

Ding-Yi Chen¹

Xue Li¹

Zhao Yang Dong¹

Xia Chen^{1,2}

School of Information Technology and Electrical Engineering,
University of Queensland, QLD 4072, Australia¹

School of Electronic and Information Engineering,
Tianjin University, P.R.China²

Email: {dingyi, xueli, zdong, chenxia}@itee.uq.edu.au

Abstract

Documents cannot be automatically classified unless they have been represented as a collection of computable features. A model is a representation of a document with computable features. However, a model may not be sufficient to express a document, especially when two documents have the same features, they might not be necessarily classified into the same category. We propose a method for determining the fitness of a document model by using conflict instances. Conflict instances are instances with exactly same features, but with different category labels given by human expert in an interactive document labelling process for training of the classifier. In our paper, we do not treat conflict instances as noises, but as the evidences that can reveal a distribution of positive instances. We develop an approach to the representation of this distribution information as a hyperplane, namely *distribution hyperplane*. Then the fitness problem becomes a problem of computing the distribution hyperplane.

Besides determining the fitness of a model, distribution hyperplane can also be used for: 1) acting as classifier itself; and 2) being a membership function of fuzzy sets. In this paper, we also propose the selection criteria of effectiveness measuring for a model in a process of fitness computations.

Keywords: Document Classification; Document Model; Document Model Fitness; Distribution Hyperplane; Conflict Instances.

1 Introduction

Document classification is a process of filing documents into different categories. It can benefit the works such as Web directory constructing (e.g., for Yahoo and Google search engines), business document routing, and email filtering. Researches on conventional classification tasks mainly concentrate on improving the effectiveness of classifiers. Several classification algorithms, such as linear least square fit (LLSF) (Yang & Chute 1994) and support vector machine (SVM) (Joachims 1998) do achieve good results. However, we noticed that effectiveness of classification is determined by not only the classification algorithms, but also the documents representation schemes, i.e., their models.

Copyright ©2005, Australian Computer Society, Inc. This paper appeared at the 16th Australasian Database Conference, University of Newcastle, Newcastle, Australia. Conferences in Research and Practice in Information Technology, Vol. 39. H.E. Williams and G. Dobbie, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

A *document model* determines how a computer recognises documents. Whenever a computer receives documents, it decomposes the documents into vectors of features. Then these vectors are used for further computations of the classification tasks. But what will happen if a document model is not good enough to differentiate the features of different documents? Let's see the following example: Suppose we are gathering travelling information of Java island. We have 10 documents that have the keyword "Java". While, some of them are about the computer language Java, the others are about travelling on Java island. In order to highlight the problem, we use a simplified document model, which only has two features: occurrence of "Sun" and occurrence of "Indonesia". Table 1 shows how these documents are modelled.

Document Serial No.	Features		Travel Info Label
	Sun	Indonesia	
1	0	1	+
2	1	0	-
3	1	1	+
4	0	1	+
5	1	0	-
6	1	1	-
7	0	1	+
8	1	0	-
9	1	1	-
10	0	1	+

Table 1: A simplified model which shows conflict instances, if the document should be filed to the "Travel Info" category, then the label '+' is assigned; otherwise, '-' is assigned.

Now, we group the records in the same document representation to generate Table 2. Both Table 1 and Table 2 show that document 3, 6 and 9 have the same representation, but their labels are different given by human expert during a training process. We call documents 3, 6, 9 *conflict instances*, for they are the same in modelling but different in their labels. A classifier learner might be confused to see those instances, for it considers they are exactly identical, but different classification decisions are expected. No matter what decision classifier makes, the mistake will be unavoidable.

What roles do conflict instances play in classification? They are the results of the incompleteness of model, because some essential features are missing from the model. Conventional approaches tend to either ignore the minor votes of conflict instances (Quinlan 1996) or use fuzzy membership function (Haruechaiyasak, Shyu, Chen & Li 2002) to obtain

Features		Label of Documents
Sun	Indonesia	
0	1	+ :1,4,7,10
1	0	- :2,5,8
1	1	+ :3 - :6,9

Table 2: Documents with the same presentation are grouped into the same row. the number after the ‘+’ or ‘-’ sign are document serial numbers.

the possibility of the document to be filed into certain categories. We, on the other hand, consider the conflict instances possess information of distribution, which can be utilised to estimate the fitness of model.

Since a document model might not be suitable for certain classification tasks, this leads to a question: “how good can the document model fit with the task?” Multiple models are available for document representation in classification tasks, it would be helpful to develop a fitness measurement to compare the suitability between different models. Although there are several measurements which are able to appraise the degree of fitness, the measurement which can reflect the upper bound of the effectiveness of classifier will be useful for classification benchmarking purposes. We choose the fitness as a measurement for following reasons: (1) The measurement shows the upper bound of the effectiveness, this implies that no matter how hard the classifiers try, it cannot exceed the upper bound score. It is unlikely to achieve the better results with the same model. (2) Even with the best model, it might still have some flaws which will lead to misclassification. In this situation, determining a good threshold will minimise the cost of misclassification, which means the better performance. So for a certain model after its fitness is known, a threshold will be able to tell a minimum expectation such that the classification is regarded as sensible.

We present an idea of *distribution hyperplane* to acquire the fitness of model. A hyperplane is a codimension 1 vector-subspace of a vector space. (Weisstein 2000). A distribution hyperplane is an artificial hyperplane used to separate the positive instances from negative ones in an artificial dimensional space. Each category has its own distribution hyperplane. Documents are decomposed into vectors, which can be represented as points in a high dimensional space with respect to the features of documents. The space is called a ‘*model space*’. Each point of the model space can be considered as a *bucket*, for it may contain multiple documents. Then we put the documents from instance set (i.e. human-labelled documents) into corresponding buckets. For each nonempty bucket, we know the *probability of positive* (POP). In other words, POP represents the probability of a document in the bucket that can be filed to the category. If there are no conflict instances in a bucket, the POP should be either 1 (the instances within the bucket are always filed into the category), or 0 (the documents with this representation are never filed into the category at all). If there are conflict instances, the POP of the corresponding buckets is a fraction number between 0 and 1. We add POP as a new dimension to the model space, make a *POP distribution space*. A bucket and its POP can be plotted in the POP distribution space as a *distribution point*. Distribution points show how positive instances and negative instances are distributed in corresponding document representations. A distribution hyperplane can be derived through the regression of a set of distribution points, which separates the positive instances and negative instances.

On the other hand the threshold in a distribution

space is also a hyperplane. It cuts the distribution space into two pieces. If a distribution point is above the the threshold, then the corresponding bucket will be filed to the category (i.e. a positive case). However, if the POP is not 1, it implies that the possibility of misclassification of the bucket is $1 - POP$. Similarly, distribution point that is below the threshold indicates a negative bucket. After the computation of misclassification, we can then get the fitness of document model. The details of the calculation are given in Section 4

Our goal is to determine the fitness of a model for a given labelled corpus. The fitness is defined as the maximum effectiveness that a classifier can achieve. In addition to model selection, the research also contributes to the following fields:

1. *Model dimensionality reduction*: dimensionality reduction techniques such as Latent Semantic Index (LSI) (Marcus & Maletic 2003) and Porter stemming algorithm (Porter 1997) not only can help to reduce the computational cost, but also avoid the over fitting problem (Sebastiani 2002). Our research on the model fitness will help to determine how much effectiveness is improved after the dimensionality reduction.
2. *Mistake-driven learning*: mistake-driven learning approaches such as Winnow (Littlestone 1991), Perceptron (Dagan, Karov & Roth 1997) and Prediction-Learning-Distillation (PLD) framework (Chen & Li 2004) rely on the identification of misclassified items to learn. However, mistake-driven learning is very sensitive to conflict instances. So if we can deal with conflict instances by reducing the uncertainty that they cause, the effectiveness of mistake-driven learning algorithms can be improved.

Moreover, our approach of distribution hyperplane has other interesting effects, for instance: being utilised as a classifier or as a fuzzy set membership function to evaluate the buckets in a model space.

The rest of this paper is organised as follows: The background information and the problem definition are given in Section 2 The related work is discussed in Section 3. Our proposed approach of the distribution hyperplane is discussed in Section 4. Section 5 gives some experimental results and discussions. The conclusion is presented in Section 6.

2 Problem Statement

2.1 Preliminary Definitions

The classification task can be considered as assigning Boolean values to a Cartesian product of a set of documents D and a set of categories C , which can be represented as:

$$\Phi : D \times C \rightarrow \{1, 0\}, \quad (1)$$

where Φ denotes the true classification function that assigns either 1 or 0 to a point (d_i, c_j) , $d_i \in D$, $c_j \in C$ in $D \times C$. If d_i should be filed to c_j , then Φ assigns a value 1 to (d_i, c_j) ; otherwise 0 is assigned (Sebastiani 2002).

An instance for classifier training is a row η in the Cartesian product $D \times C$. It can also be represented as: $\eta_i = (d_i, \omega_i)$, where ω_i is the set of categories that d_i belongs to.

In order to make a document computable for its features, we transform document d into its representation \tilde{d} of model M . The representation \tilde{d} is a vector of weighted features $\tilde{d} = [f_1, f_2, \dots, f_k]$, where k is the

total number of features. So the transformation can be expressed as:

$$d \xrightarrow{M} \tilde{d} = [f_1, f_2, \dots, f_k] \quad (2)$$

According to Formula 2, \tilde{d} is a point (f_1, f_2, \dots, f_k) in a k -dimensional model space. A point in the model space is called a ‘bucket’, for it may contain several documents. For any nonempty bucket, if it has two instances η_x and η_y , which $\tilde{d}_x = \tilde{d}_y$, but $\omega_x \neq \omega_y$, then η_x, η_y and all other instances in the same bucket are conflict instances.

For each bucket, the POP is defined as:

$$POP = \frac{\text{Number of positive instances in the bucket}}{\text{Number of instances in the bucket}} \quad (3)$$

2.2 Problem Definition

Given a document set D , a category set C , and a document model M , M can be regarded as a function defined over $D \times C$ written as $M : D \times C$. In general the document set D can be grouped into a bucket set B by M :

$$D \xrightarrow{M} B \quad (4)$$

Where for any $b_i \in B$ and $b_j \in B$, $b_i \cap b_j \neq \phi$. So a document set D is grouped as a set of overlapping subsets B based on model M . A subset b_i of B can be obtained by $b_i = \{d_j | d_j \in D, j = 1, 2, \dots, k, M(d_j) = c_k, c_k \in C\}$.

Now the problem is stated as that for a given bucket set B , we need to:

1. Calculate the POP based on the Cartesian product $B \times C$ for each $(b_i, c_j) | b_i \in B, c_j \in C$. The results will form up a distribution hyperplane by regression.
2. Determine the positive and negative ratio that a classifier can make by applying a threshold based on the $B \times C$.

The fitness of model M can then be seen as a maximum value of correctness that a classifier can achieve based on M .

3 Related Work

Document models such as binary independence retrieval (a.k.a bag of words) (Robertson & Jones 1976), vector space model (Salton, Wong & Yang 1975), Darmstadt indexing approach (Fuhr 1989), and LSI (Deerwester, Dumais, Furnas, Landauer & Harshman 1990) have been used to represent documents. However, none of them guarantees that it can sufficiently cover all of the relevant features, therefore, conflict instances may be the case.

Different approaches handle conflict instances in different ways. In rule induction such as C4.5 (Quinlan 1996), conflict examples are simply removed. In the approach of rough sets (Pawlak, Grzymala-Busse, Slowinski & Ziarko 1995), the lower approximation (non-conflict examples) can be used to build ‘‘certain rules’’, and the rules induced from upper approximation (conflict examples) can be used to build ‘‘possible rules’’. Lower approximation completely discards conflict examples, while upper approximation only deals with the conflict examples that win votes for a label. Lau (Lau, Bruza & Song 2004) utilises AGM belief to deal with data uncertainty. Liu (Liu & Song 2003) uses fuzzy sets to

avoid the ambiguity of features and the membership function to make classification decisions.

In those approaches, conflict instances are used to show the uncertainty of classification results. We, however, found a novel way to explain and utilise conflict instances. The next section introduces our ideas of using conflict instances as an indicator to separate the positively classified examples from the rest.

4 Proposed Approach

In terms of measuring the fitness of document model over a given document corpus, the distribution of positive instances and negative instances are essential. The distribution hyperplane, to be introduced in this section, is developed for this purpose.

4.1 Distribution Hyperplane

A distribution hyperplane is the hyperplane that splits positive and negative instances in a distribution space. The following example illustrates the brief idea of how the distribution hyperplane would look like.

Assume two features, x and y , are necessary and sufficient to determine whether an email is a spam or not. Feature x stands for the spam likelihood through the key-words filtering of the content, while y stands for the score of checking-up with a senders’ mail server. A positive decision is made if an email filter considers the email is a spam, otherwise a negative decision is made. Table 3 shows the agreement between user judgement and classifier decision (Sebastiani 2002):

Should document d be filed to c		User Judgements	
		Positive	Negative
Classifier Decision	Positive	TP	FP
	Negative	FN	TN

Table 3: The contingency table of user judgements and classifier decisions.

Where

1. TP (True Positive): Classifier correctly makes positive decisions.
2. TN (True Negative): Classifier correctly makes negative decisions.
3. FP (False Positive): Classifier mistakenly makes positive decisions.
4. FN (False Negative): Classifier mistakenly makes negative decisions.

If, at the time of classification, only x were recognised, y could not be seen by classifier (i.e., the model were insufficiently defined as $\tilde{d} = \{x\}$), then no matter how hard would have the classifier tried, the misclassification would be unavoidable. In this case, the correct classifications would be those points within the areas of TP and TN. The incorrect classifications would be those points within the areas of FP and FN.

Figure 1 illustrates the impact of incompleteness of a document model. The gray area illustrates the positive instances (i.e., spams), while white area shows the negative instances. The *actual separating hyperplane* is the hyperplane which actually separates the positive and negative instances. In this example, the actual separating hyperplane is $x + y = 1$. The actual decision function is:

$$\Phi = \begin{cases} 1 & , \text{ if } 1 - x - y \geq 0 \\ 0 & , \text{ otherwise} \end{cases} \quad (5)$$

However, since the classifier can only see the feature x , the best result of the classifier is the separating hyperplane $x = 0.5$. Therefore, the classifier uses following function to make classification decisions:

$$\Psi = \begin{cases} 1 & , \text{ if } 0.5 - x \geq 0 \\ 0 & , \text{ otherwise} \end{cases} \quad (6)$$

To shorten the mathematical expression, we use $\Phi : \langle \text{formula} \rangle$ and $\Psi : \langle \text{formula} \rangle$ to represent the decision functions of actual distribution and classifier. If the value of *formula* is greater or equals 0, positive decision will be made, otherwise, negative decision.

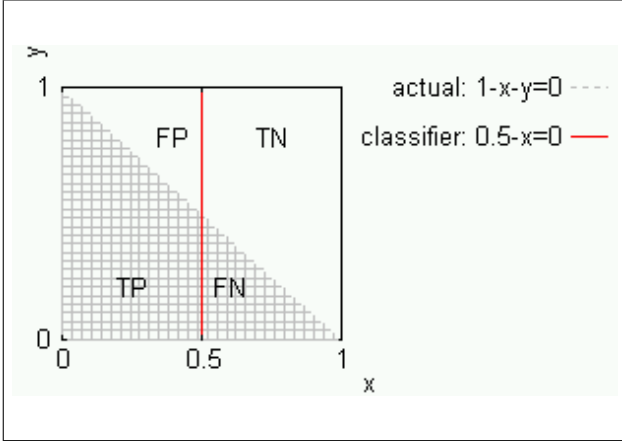


Figure 1: Actual decision function $\Phi : 1 - x - y$ and classifier-generated decision function $\Psi : 0.5 - x$ with model $\{x\}$

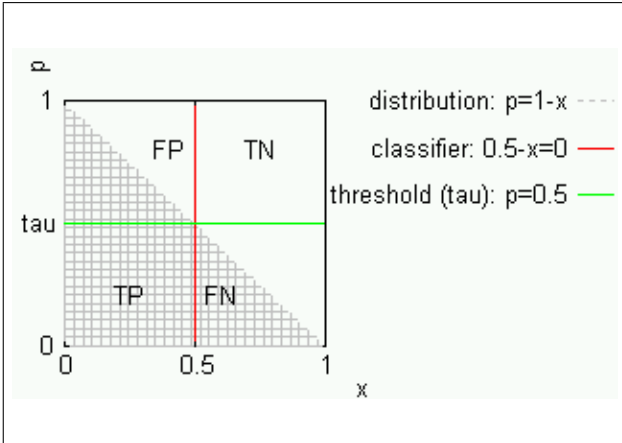


Figure 2: Distribution hyperplane of model $\{x\}$. Distribution hyperplane $p = 1 - x$ separates the positive instances and negative instances. With the threshold 0.5, classifier can generate the most effective separating hyperplane: $0.5 - x = 0$.

Figure 2 illustrates the distribution hyperplane approach. Suppose we can not access y , but we can get the POP value p , that is, the possibility that positive instances occur for given x . Based on POP, we can generate the distribution hyperplane $p = 1 - x$, which separates the region of positive instances (spams) for the gray area, and the region of negative ones (not spams) for the white area.

A threshold τ demonstrates the user's preference. If $p \geq \tau$, then a positive decision will be made. In order to obtain better outcome, the threshold

$\tau : \tau \in R, 0 \leq \tau \leq 1$ should be set to:

$$\tau = \frac{\lambda_{FP} - \lambda_{TN}}{(\lambda_{FN} - \lambda_{TP}) + (\lambda_{FP} - \lambda_{TN})}, \quad (7)$$

where $\lambda_{TP}, \lambda_{FP}, \lambda_{TN}, \lambda_{FN}$ are the penalty for TP, FP, TN and FN (Cooper 1973, Lewis 1995). For example, in email filtering, FP (identify legitimate mail as spam) is worse than FN (identify spam as legitimate mail), in this case, we should set λ_{FP} greater than λ_{FN} . Nevertheless, if we show no special interests among TP, TN, FP , and FN , which implies $\lambda_{FP} = \lambda_{FN}, \lambda_{TP} = \lambda_{TN} = 0$, therefore, $\tau = 0.5$. We use this value as default if not mentioned.

Please note that the distribution hyperplane is not always equivalent to the actual separating hyperplane.

4.2 Calculating the Distribution Hyperplane

A document model can be represented as a k -dimensional space (a.k.a. model space), and each document as a point in model space is mapped to a bucket. We can consider the POP is the "height" of the bucket, and plot it in a $(k + 1)$ -dimensional space, namely the distribution space. Let B be a set of nonempty buckets (with document instances), then we have $|B|$ number of points in distribution space, where $|B|$ stands for the number of nonempty buckets in B . The distribution hyperplane can be generated with $|B|$ by regression methods such as Linear Least Square Fit (LLSF) or SVM (Support Vector Machine) regression (Chang & Lin 2002). After regression, the distribution hyperplane can be expressed as follows:

$$p = \psi(f_1, f_2, \dots, f_k) \quad (8)$$

where $f_i, i = 1, 2, \dots, k$ are features.

In the email filter example, feature x might be split into say, 20 buckets. The reason to split feature x is to get the statistically significant data. It is very unlikely that the number of instances is enough to provide the statistically significant data for every single real number. We compute the POP for every bucket, and use the LLSF to get the distribution hyperplane $p = \psi(x) = 1 - x$ with the method stated above.

4.3 Fitness of Document Model

We regard the fitness as the maximum effectiveness of a classifier, the effectiveness measurement of a classifier can be achieved. Commonly used effectiveness measurements are: accuracy, precision-recall break even point, F_1 function, and Pearson correlation. Here we choose the Pearson correlation as the measurement of fitness, which can be calculated by:

$$\rho = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(FP + TN)(FN + TN)}}, \quad (9)$$

Correlation $\rho = 1$ indicates that the model is perfectly fit, and $\rho = 0$ indicates that the model is not fit at all, while $\rho = -1$ means that the model is misleading.

To calculate the effectiveness, we need a document corpus, a model and a value τ as threshold. Firstly, we utilise the model to decompose the documents to the sets of features, then put documents with the same features into the same bucket. After that, we compute p_b , the POP for bucket b for each nonempty bucket in bucket set (model space) B . We use following formulae to obtain the values of TP, TN, FP, and FN:

$$\begin{aligned} TP &= \sum_{b \in B} (p_b | p_b \geq \tau) \\ TN &= \sum_{b \in B} (1 - p_b | p_b < \tau) \\ FP &= \sum_{b \in B} (1 - p_b | p_b \geq \tau) \\ FN &= \sum_{b \in B} (p_b | p_b < \tau) \end{aligned} \quad (10)$$

Finally, we can put TP, TN, FP, and FN into our effectiveness measurement function, Pearson correlation, to obtain the score of model fitness.

In the email filter example, the fitness of model is 0.5.

4.4 An Example of Fitness Computation Procedures

We use the following example to illustrate the calculation of the fitness of a model.

There are five documents in document corpus $D = d_1, d_2, \dots, d_5$, and three predefined categories $C = c_1, c_2, c_3$. Table 4 shows the classification of $(d_i, c_j), d_i \in D, c_j \in C$, filed by human expert in a training process. If d_i should be filed into c_j , then 1 is assigned, otherwise, 0 is assigned. These numbers are regarded as the POP values as well, for 1 stands for 100% chance that d_i belongs to c_j , while 0 stands for there is no chance that d_i belongs to c_j .

Document	c_1	c_2	c_3	ω
d_1	1	0	1	$\omega_1 : \{c_1, c_3\}$
d_2	1	0	0	$\omega_2 : \{c_1\}$
d_3	0	1	1	$\omega_3 : \{c_2, c_3\}$
d_4	0	0	1	$\omega_4 : \{c_3\}$
d_5	1	0	1	$\omega_5 : \{c_1, c_3\}$

Table 4: Cartesian product of $D \times C$, with the POP for each (d_i, c_j) pairs

Assume we utilise the document model: $\tilde{d} = [f_1, f_2]$, Table 5 shows how documents are represented in the model:

Document	f_1	f_2
\tilde{d}_1	1	1
\tilde{d}_2	0	1
\tilde{d}_3	1	1
\tilde{d}_4	1	0
\tilde{d}_5	1	1

Table 5: Documents represented by model $\tilde{d} = [f_1, f_2]$

Step 1: Map documents to buckets

Since $\tilde{d}_1 = \tilde{d}_3 = \tilde{d}_5$, but $\omega_1 \neq \omega_3 \neq \omega_5$, therefore, d_1, d_3, d_5 are conflict instances. We put all documents into buckets, then compute the POP of the $(b_m, c_j), b_m \in B$ pairs from the average of the POP of the $(d_i, c_j), d_i \in b_m$ pairs. Coordinate is the position of bucket in model space, i.e. (f_1, f_2) . Details is shown in Table 6.

Bucket	Coordinate	c_1	c_2	c_3
$b_1 = \{\tilde{d}_2\}$	(0,1)	1	0	0
$b_2 = \{\tilde{d}_4\}$	(1,0)	0	0	1
$b_3 = \{\tilde{d}_1, \tilde{d}_3, \tilde{d}_5\}$	(1,1)	0.66	0.33	1

Table 6: $B \times C$ with the POP for each $(b_m, c_j), b_m \in B$ pair. POP for each (b_m, c_j) pair is counted by average of the POP of the $(d_i, c_j), d_i \in b_m$ pairs.

Step 2: Add POP as a new dimension to plot distribution points

We add POP as a new dimension to buckets, then, the distribution points for each category is shown in Table 7:

Bucket	c_1	c_2	c_3
b_1	(0,1,1)	(0,1,0)	(0,1,0)
b_2	(1,0,0)	(1,0,0)	(1,0,1)
b_3	(1,1,0.67)	(1,1,0.33)	(1,1,1)

Table 7: Distribution points for each category.

Step 3: Generate distribution hyperplane by regression

One distribution hyperplane for each category can be obtained by applying regression method over these points. We utilise the linear least square fit regression to compute the coefficient of the distribution hyperplane for each category. The distribution hyperplane for c_1 is $p = 0.67f_1 - 0.33f_2 + 0.33$, for c_2 is $p = 0.33f_1 + 0.33f_2 - 0.33$, for c_3 is $p = f_2 - 1$.

Step 4: Fill the POP of empty buckets by using distribution hyperplane

In our example, bucket (0,0) has never appeared, which implies it is empty. However, they might appear later. So we use the distribution hyperplane to estimate the POP of the bucket (0,0). POP should be always in the range between [0,1], therefore, if $\text{POP} > 1$, then we set $\text{POP} = 1$; likewise, if $\text{POP} < 0$, then we set $\text{POP} = 0$. After we calculate the POP of bucket (0,0), we can get Table 8.

Bucket	Coordinate	c_1	c_2	c_3
$b_1 = \{\phi\}$	(0,0)	0.33	0	0
$b_1 = \{\tilde{d}_4\}$	(0,1)	0	0	1
$b_2 = \{\tilde{d}_2\}$	(1,0)	1	0	0
$b_3 = \{\tilde{d}_1, \tilde{d}_3, \tilde{d}_5\}$	(1,1)	0.67	0.33	1

Table 8: $B \times C$ with the POP for each $(b_m, c_j), b_m \in B$ pair. POP for each (b_m, c_j) pair is counted by average of the POP of the $(d_i, c_j), d_i \in b_m$ pairs.

Step 5: Calculate the TP, TN, FP and FN

According to Table 8 and Function 10, if the threshold $\tau = 0.5$ is accommodated, then we can calculate TP, TN, FP, FN for each category, then compute overall scores by adding up the values of every category. Table 9 lists the results.

	c_1	c_2	c_3	Overall
TP	1.67	0	2	3.67
TN	1.67	3.67	2	5.37
FP	0.33	0	0	0.33
FN	0.33	0.33	0	0.66

Table 9: Results of TP, TN, FP, FN calculations.

Step 6: Using the effectiveness measurement to determine the fitness of model

With Formula 9, we can determine that fitness of the model is 0.80. Other effectiveness measurements such

as accuracy or F_β can also be accommodated by using similar methods.

4.5 Other Usages of Distribution Hyperplane

The distribution hyperplane can be used in following ways:

4.5.1 Classifier

For each category c , classification decision can be made by the decision function $\Psi_c(d)$:

$$\Psi_c(d) = \begin{cases} 1 & , \text{ if } \psi(d) \geq \tau \\ 0 & , \text{ otherwise} \end{cases} \quad (11)$$

If $\Psi_c(d) = 1$, then d should be filed to c ; otherwise, d should not be filed to c .

4.5.2 Fuzzy Set Membership Function

When making classification decisions, classifier computes the likelihood of a given document being filed in to a given category, The categorisation status value (CSV) is used to describe likelihood which is a number between 0 and 1 (Sebastiani 2002). The meaning of CSV is exactly identical to POP. In order to obtain CSV, we can consider buckets as fuzzy sets, then POP is the degree of fuzzy membership. Therefore, $\psi(\tilde{d})$ is the membership function for the bucket corresponding to d .

5 Experiments and Results

In our approach, distribution hyperplane plays an important role of fitness estimation. There are two methods to estimate a distribution hyperplane:

1. Empirical Observation: Empirical observation is a data preparation technique. By this method, values of features are grouped according to class intervals. Class mark (average of the value of the two ends of a class interval) for each class interval represents the value of feature of bucket. Each distribution point from empirical observation provides an example of real data distribution, however, in high dimensional space, the empirical observation may not cover whole model space, hence, the conclusion from empirical observation might not expendable for whole model space.
2. Regression: Regressed function is generated by applying regression method on instances. The regressed function can cover whole model space, but, may not reflect the actual separating hyperplane.

In order to compare these methods, we calculate two fitness scores from each method, and compare them with the fitness score of actual separating hyperplane. If the fitness score from one method is closer to the fitness score of actual separating hyperplane, the method estimates the distribution hyperplane better.

Currently available real document corpora, such as Reuters-21578 (Lewis 2000), do not show actual separating hyperplane. Therefore, we generate two artificial instance sets – ‘Linear’ and ‘Circular’ – as measurement benchmarks.

5.1 Experiments Design

5.1.1 Instance Sets

All of the artificial instance sets have two relevant features, f_1 and f_2 , which are generated by random numbers between 0 and 1. A label of each instance in the instance set namely Linear is determined by the decision function $\Phi_{linear} = 1 - f_1 - f_2$. If $\Phi_{linear} > 0$, then positive label is assigned. Instances without positive labels are negative instances. The decision function for Circle instance set is $\Phi_{circular} = 0.5^2 - (f_1 - 0.5)^2 - (f_2 - 0.5)^2$. Positive label is assigned if $\Phi_{circular} > 0$. Instances without positive labels are negative instances. Each instance set has 40,000 instances. The original Linear instance set is plotted as Figure 3; while the Circle instance set is plotted in Figure 4. The light colour points are positive instances, while dark colour points are negative ones.

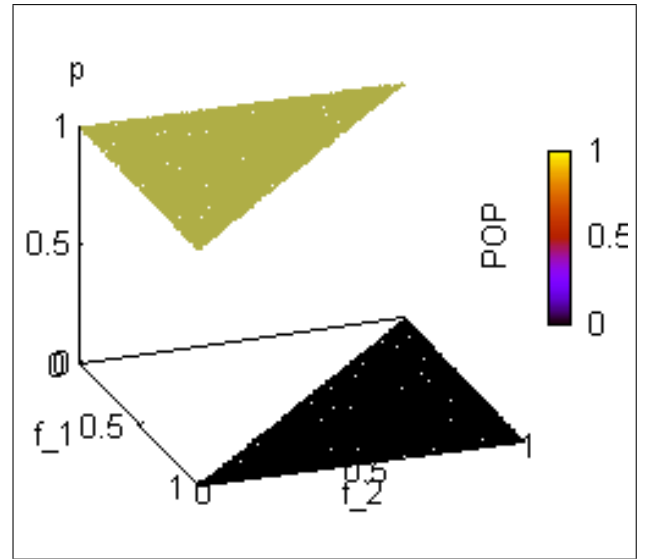


Figure 3: Original instance distribution of instance set namely ‘Linear’ in distribution space. The colour indicates POP values, the lighter the colour is, the higher value the POP is.

5.1.2 Models

Two models are used to decompose instances into features. One is an incomplete model M_1 , which contains only f_1 , the other is a perfect model M_2 , which contains both f_1 and f_2 .

5.1.3 Threshold

Since we have no special interest on one of the TP, TN, FP, and FN, threshold τ is set to 0.5 according to Formula 7.

5.1.4 Distribution Hyperplane Estimation

Practically, empirical observation are ‘binned’ instances. ‘Bin’ is an interval into which a given data point does or does not fall (Weisstein 1999). Since features f_1 and f_2 are the random real numbers, this implies the number of buckets will be infinite if we don’t group them together. Hence, for the reason of statistical significance, values of feature are binned into 10 bins. In other words, 40,000 instances are merged into 10 buckets in incomplete model, or 100 buckets in perfect model.

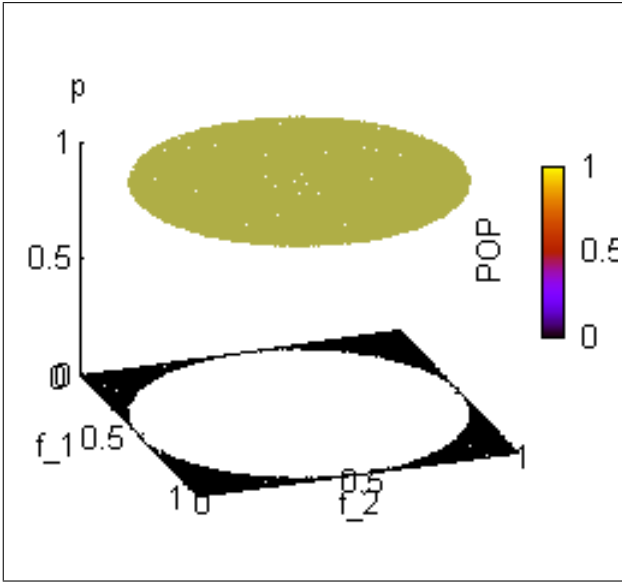


Figure 4: Original instance distribution of instance set namely ‘Circular’ in distribution space. The colour indicates POP values, the lighter the colour is, the higher value the POP is.

Please note that ‘bin’ and ‘bucket’ are not the same concept. ‘Bin’ is a statistics term which denotes the discretisation process of continuous value for a feature; while ‘bucket’ is the discretisation process for feature combination.

As for regressed function, we used LibSVM (Chang & Lin 2001) ν -SVR with RBF kernel to generate the regressed functions from the instance sets.

5.2 Results of Experiments

We group the value of each feature into 10 bunches. The reasons are: firstly, the number of buckets will be greatly reduced to a computable amount; secondly, the sizes of buckets (the number of instances in the bucket) are large enough to reach statistical significance.

5.2.1 Model 1: Incomplete Model

Figure 5 and 6 compare the estimated distribution hyperplanes to the actual separating hyperplane for the instance set ‘Linear’. Figure 7 and 8 show the comparisons for the instance set ‘Circular’. All of them are quite close to the actual separating hyperplane.

5.2.2 Model 2: Perfect Model

Figure 9 and 10 compare the estimated distribution hyperplanes to the actual separating hyperplane for the instance set ‘Linear’, while Figure 11 and 12 show the comparisons for the instance set ‘Circular’. According to these figures, regressed distribution hyperplanes indicate that there will be some possibility that model 2 will lead to some misclassification because some points are within the range $[0,1]$. This phenomenon shows that a perfect model’s distribution hyperplane may be distorted by a regression algorithm.

5.3 Experiments Summary

With incomplete models, both regressed function and empirical observation can make distribution hyperplanes that are very close to actual separating hyperplane of POP values. However, with perfect models,

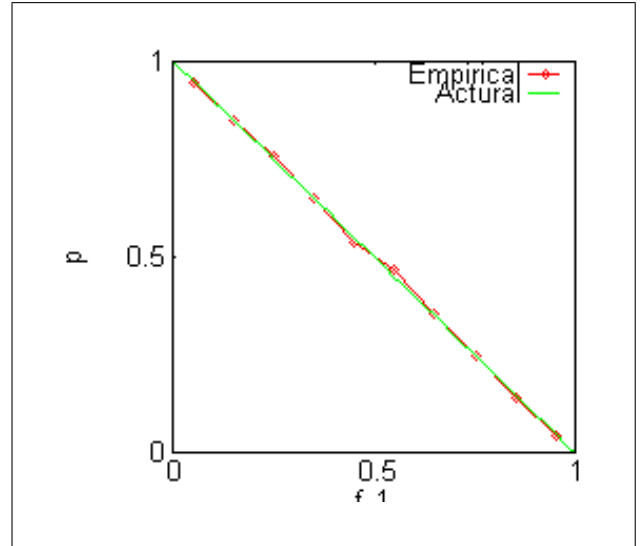


Figure 5: ‘Linear’ instances with respect to Model 1: Empirical observation vs. actual separating hyperplane.

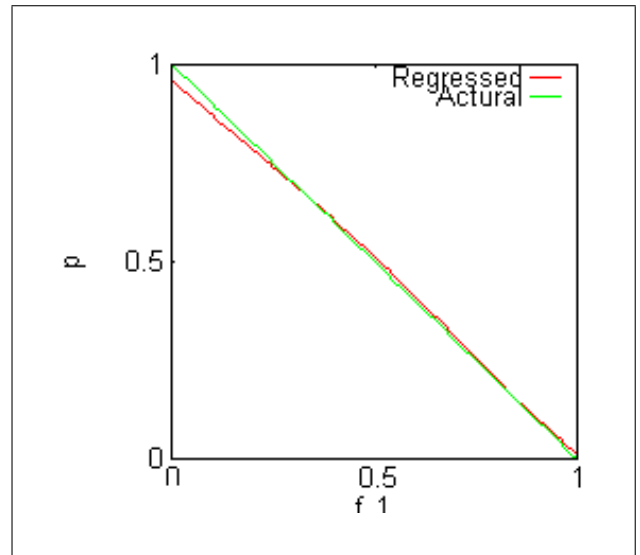


Figure 6: ‘Linear’ instances with respect to Model 1: Regressed function vs. actual separating hyperplane.

regressed function cannot show advantages over empirical observation. Those results suggest that empirical observation might be a better way to provide a feasible approximation of the actual separating hyperplane.

6 Conclusion

In this paper, we have discussed the fitness measurement of document models in classification tasks. Our proposed approach considers with the conflict instances – a symptom of weakness of a model. The measurement is used to tell not only the suitability of a document model for classification purposes, but also the upper-bound effectiveness of a classifier.

Moreover, our approach of distribution hyperplane can be used for different purposes that are not just for the fitness measurement. Firstly, it is classifier independent, so we don’t have to use multiple classifiers to verify the fitness. Secondly, we can predict the correctness of a given point in model space even the point has never shown in instance set before. Finally, the distribution hyperplane can serve as a clas-

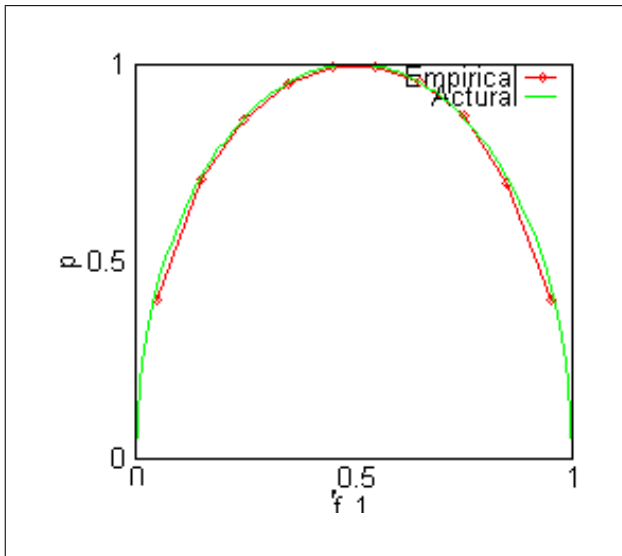


Figure 7: ‘Circular’ instances with respect to Model 1: Empirical observation vs. actual separating hyperplane.

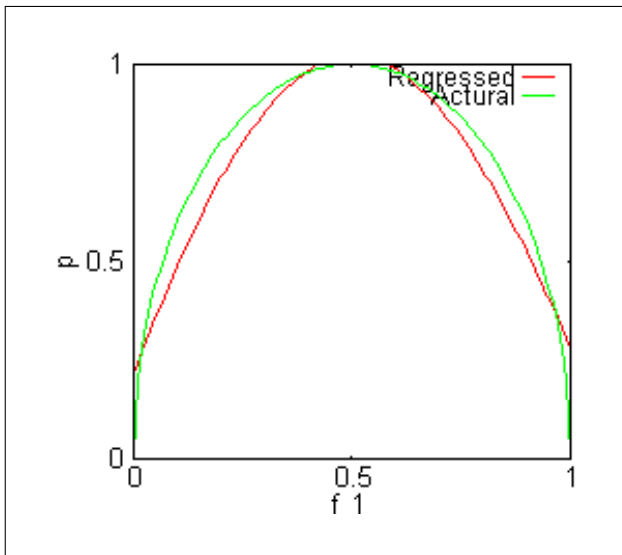


Figure 8: ‘Circular’ instances with respect to Model 1: Regressed function vs. actual separating hyperplane.

sifier, which is capable of evaluating how confident it is about the classification decisions.

Our experiments suggest that the empirical observation is better than regressed function when we have a great amount of instances (e.g. be 400 instances per bucket). However, we might not able to obtain enough instances for each bucket, so we are examining and comparing the empirical observation, interpolation methods, and regressed function in order to further improve the ability to predict the POP of empty buckets.

Furthermore, generally complex models tend to get higher score on fitness, but cannot provide significant advantages over the performances and effectiveness (Dumais, Platt, Heckerman & Sahami 1998). We are currently applying our techniques to address the overfilling problem in order to provide a fair way to compare the fitness of different types of models.

References

Chang, C.-C. & Lin, C.-J. (2001), *LIBSVM: a library for support vector machines*.

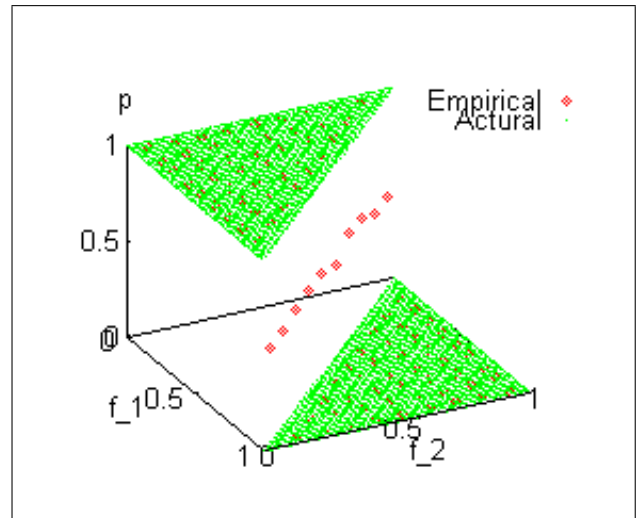


Figure 9: ‘Linear’ instances with respect to Model 2: Empirical observation vs. actual separating hyperplane.

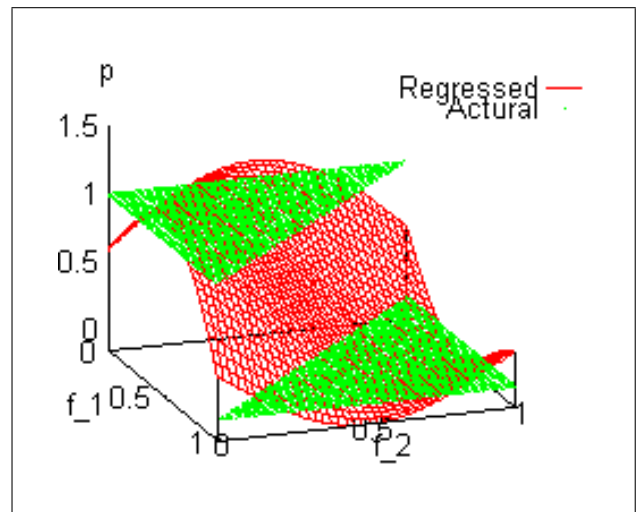


Figure 10: ‘Linear’ instances with respect to Model 2: Regressed function vs. actual separating hyperplane.

<http://www.csie.ntu.edu.tw/~cjlin/libsvm>.

Chang, C.-C. & Lin, C.-J. (2002), ‘Training nu-support vector regression: theory and algorithms’, *Neural Computation* **14**, 1959–1977.

Chen, D.-Y. & Li, X. (2004), PLD: A distillation algorithm of misclassified documents, in ‘Advances in Web-Age Information Management (WAIM), Lecture Note in Computer Science (LNCS 3219)’, Springer, pp. 499–508.

Cooper, W. S. (1973), ‘On selecting a measure of retrieval effectiveness’, *Journal of the American Society for Information Science* **24**, 87–100.

Dagan, I., Karov, Y. & Roth, D. (1997), Mistake-driven learning in text categorization, in ‘Proceedings of EMNLP-97, 2nd Conference on Empirical Methods in Natural Language Processing’, Association for Computational Linguistics, Morristown, US, Providence, US, pp. 55–63.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K. & Harshman, R. (1990), ‘Indexing by latent semantic analysis’, *Journal of the American Society for Information Science and Technology* **41**(6), 391–407.

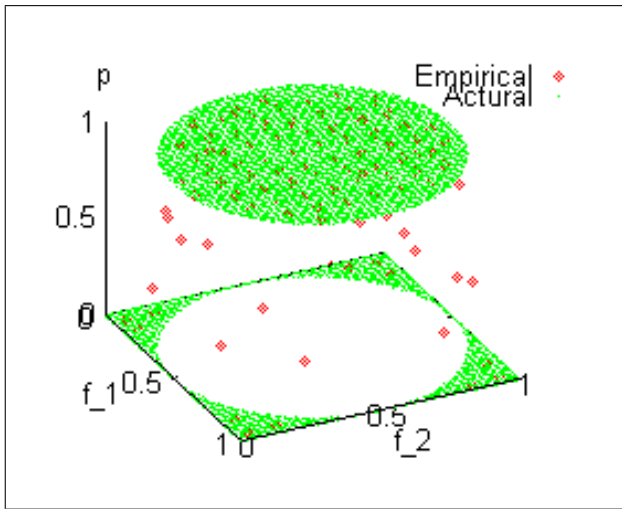


Figure 11: ‘Circular’ instances with respect to Model 2: Empirical observation vs. actual separating hyperplane.

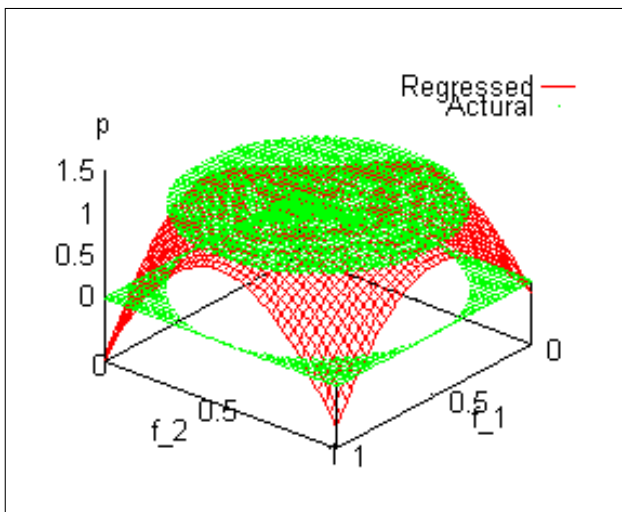


Figure 12: ‘Circular’ instances with respect to Model 2: Regressed function vs. actual separating hyperplane.

Dumais, S., Platt, J., Heckerman, D. & Sahami, M. (1998), Inductive learning algorithms and representations for text categorization, in ‘Proceedings of the seventh international conference on Information and knowledge management’, ACM Press, Bethesda, Maryland, United States, pp. 148–155.

Fuhr, N. (1989), ‘Models for retrieval with probabilistic indexing’, *Information Processing and Management* **25**(1), 55–72.

Haruechaiyasak, C., Shyu, M.-L., Chen, S.-C. & Li, X. (2002), Web document classification based on fuzzy association, in ‘Proceedings of the 26th IEEE Computer Society International Computer Software and Applications Conference (COMP-SAC)’, Oxford, England, pp. 487–492.

Joachims, T. (1998), Text categorization with support vector machines: learning with many relevant features, in C. N. Rouveirol & Celine, eds, ‘Proceedings of ECML-98, 10th European Conference on Machine Learning’, Springer Verlag, Heidelberg, DE, Chemnitz, DE, pp. 137–142.

Lau, R. Y., Bruza, P. D. & Song, D. (2004), Belief revision for adaptive information retrieval, in ‘ACM SIGIR’.

Lewis, D. D. (1995), Evaluating and optimizing autonomous text classification systems, in ‘Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval’, ACM Press, Seattle, Washington, United States, pp. 246–254.

Lewis, D. D. (2000), ‘Reuters corpus (21578)’.

Littlestone, N. (1991), Redundant noisy attributes, attribute errors, and linear-threshold learning using winnow, in ‘Proceedings of the fourth annual workshop on Computational learning theory’, Morgan Kaufmann Publishers Inc., Santa Cruz, California, United States, pp. 147–156.

Liu, W. Y. & Song, N. (2003), ‘A fuzzy approach to classification of text documents’, *Journal of Computer Science and Technology* **18**(5), 640–647.

Marcus, A. & Maletic, J. I. (2003), Recovering documentation-to-source-code traceability links using latent semantic indexing, in ‘Proceedings of the 25th international conference on Software engineering’, IEEE Computer Society, Portland, Oregon, pp. 125–135.

Pawlak, Z., Grzymala-Busse, J., Slowinski, R. & Ziarko, W. (1995), ‘Rough sets’, *Communications of the ACM* **38**(11), 88–95.

Porter, M. F. (1997), ‘An algorithm for suffix stripping’.

Quinlan, J. R. (1996), ‘Improved use of continuous attributes in c4.5’, *Journal of Artificial Intelligence Research* **4**, 77–90.

Robertson, S. E. & Jones, K. S. (1976), ‘Relevance weighting of search terms’, *Journal of the American Society for Information Science* **27**, 129–146.

Salton, G., Wong, A. & Yang, C. (1975), ‘A vector space model for automatic indexing’, *Communications of the ACM* **18**(11), 613–620.

Sebastiani, F. (2002), ‘Machine learning in automated text categorization’, *ACM Computing Surveys (CSUR)* **34**(1), 1–47.

Weisstein, E. W. (1999), ‘Bin’. From MathWorld—A Wolfram Web Resource <http://mathworld.wolfram.com/Bin.html>.

Weisstein, E. W. (2000), ‘Hyperplane’. From MathWorld—A Wolfram Web Resource <http://mathworld.wolfram.com/Hyperplane.html>.

Yang, Y. & Chute, C. G. (1994), ‘An example-based mapping method for text categorization and retrieval’, *ACM Transactions on Information Systems (TOIS)* **12**(3), 252–277.