Redundant Feature Elimination for Multi-Class Problems

Annalisa Appice APPICE@DI.UNIBA.IT
Michelangelo Ceci CECI@DI.UNIBA.IT
Dipartimento di Informatica, Università degli Studi di Bari, Via Orabona 4, 70126 Bari, Italy

Simon Rawles SIMON.RAWLES@BRISTOL.AC.UK
Peter Flach PETER.FLACH@BRISTOL.AC.UK
Department of Computer Science, University of Bristol, Woodland Road, Bristol BS8 1UB, United Kingdom

Abstract

We consider the problem of eliminating redundant Boolean features for a given data set, where a feature is redundant if it separates the classes less well than another feature or set of features. Lavrač et al. proposed the algorithm REDUCE that works by pairwise comparison of features, i.e., it eliminates a feature if it is redundant with respect to another feature. Their algorithm operates in an ILP setting and is restricted to two-class problems. In this paper we improve their method and extend it to multiple classes. Central to our approach is the notion of a *neighbourhood* of examples: a set of examples of the same class where the number of different features between examples is relatively small. Redundant features are eliminated by applying a revised version of the REDUCE method to each pair of neighbourhoods of different class. We analyse the performance of our method on a range of data sets.

1. Introduction

Classification is one of the fundamental tasks in machine learning. In the usual classification setting, input or training data consists of multiple examples, each having multiple attributes or features X_i. Each example is tagged with a class label c_j. The goal is to learn the target concept associated with each class by finding regularities in examples of a class that characterize the class in question and discriminate it from the other classes. This problem has been extensively studied in machine learning. However, the growing importance of knowledge discovery and data mining in practical real world applications, where data consists of a large amount of records that may be stored in different tables of a relational database, requires increasingly sophisticated solutions for classification problems such as multi-relational data mining or propositionalisation. In particular, *propositionalisation* (Dzeroski & Lavrač, 2001; Krogel et al., 2003) is the process of transforming a multi-table representation of data into the form of a single table. The result can be used as input for attribute-value learning algorithms.

Propositionalisation tends to produce large numbers of features, many of which are highly correlated or even logically redundant. A simple example of a redundant feature is one which is never (or always) satisfied: e.g., 'a molecule having an atom which has a bond with itself'. While some forms of redundancy can be recognised at feature generation time, others can only come to light by examining the data. Pagallo & Haussler (1990) investigate the problem of discovering a subset of Boolean features to describe two-class Boolean concepts. using three different algorithms. FRINGE induces decision trees in which Boolean features are adaptively constructed at each node whereas GREEDY3 and GROVE use greedy accuracy-based heuristics to build decision lists. These methods select Boolean features as part of learning rather than according to any definition of logical redundancy. In contrast, (Lavrač et al., 1999) proposed a method to remove Boolean features that are logically redundant with respect to a two-class data set. They defined a feature *f* to be *redundant*[1] with respect to another feature *g* if *g* is true for at least the same positive examples as *f* and false for at least the same negative examples as *f*. The REDUCE algorithm operates by pairwise comparison of the features.

In this paper we improve and extend the REDUCE algorithm in two ways. First, we increase the number of redundant features detected, without losing the ability to find a complete and consistent theory with the reduced feature set. Secondly, we extend the method to multi-class problems. Central to our approach, which we call REFER (REdundant FEature Reduction) is the notion of a *neigh-*

Appearing in *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004.

[1](Lavrač et al., 1999) use the term 'irrelevant' rather than 'redundant'. However, we distinguish between relevance and redundancy (see below).

bourhood of examples: a set of examples of the same class where the number of different features between examples is relatively small. Redundant features are eliminated by applying a revised version of the REDUCE method to each pair of neighbourhoods of different class.

The paper is organized as follows. In the next section we discuss the background to our work and some related work. The method is presented in Section 3. Experimental results are reported in Section 4 and conclusions are drawn in Section 5.

2. Motivation and Background

It is useful to distinguish between feature selection and feature reduction. *Feature selection* is concerned with identifying a small subset of *relevant* features that are sufficient for learning the target concept. We define *feature reduction* as eliminating logically *redundant* features. In a certain sense, the two approaches are orthogonal: feature selection aims at increasing correlation with the class (relevance), feature reduction aims at reducing correlation among features (redundancy). This different focus may lead to different results: for instance, if we have several copies of a highly relevant feature, feature selection will select all of them, while feature reduction will eliminate all but one.

2.1 Feature Selection

The problem of feature selection has been widely explored in machine learning. Reasons include that irrelevant features may deteriorate the predictive performance of a learning algorithm (Langley, 1996; Rendell & Seshu, 1990), as well as reduce the comprehensibility of the learned model (Dash & Liu, 1997; Blum & Langley, 1997). Feature selection approaches may be categorised into wrapper, filter and embedded approaches (Kohavi et al., 1994; Blum & Langley, 1997), according to whether the method takes into account the characteristics of the data, the target concept or the learning algorithm.

In the *wrapper* approaches, the goal is to find a subset of features that maximizes accuracy. The wrapper approach implies that the selection algorithm searches for a good subset of features using the induction algorithm itself as a part of the evaluation function, the same algorithm that will be used to learn the final target concept. In the *filter* approaches, the goal is to filter the irrelevant and/or redundant features on the basis of the characteristics of the training data without involving any learning algorithm. Finally, in the *embedded* approaches the feature selection process is done inside the learning algorithm. For example, partitioning and divide-and-conquer methods implicitly select features for inclusion in a branch or rule in preference to other features that appear less relevant.

Embedded approaches are intrinsic to some learning algorithms and so only those algorithms designed with this characteristic can be used. However, embedded approaches such as Adaboost (Freund & Schapire, 1997) can often be quicker than wrapper or filter approaches since they perform feature selection and induction at the same time. Filtering approaches ensure a fairly good computational complexity, but the wrapper approaches with their higher complexity tend to produce higher resulting accuracy. Filtering approaches are very flexible, since any target learning algorithm can be used, while the wrapper approach is strictly dependent on the learner.

Among the filter approaches, it is also possible to distinguish between approaches based on probabilistic distance measures, probabilistic dependence measures, interclass distance measures or information theoretic measures such as the entropy. For instance, RELIEF (Kira & Rendell, 1992) and its extension to multi-class problems RELIEFF (Kononenko, 1994) estimate the relevance of each feature according to the difference between the selected example and the nearest examples of the same and different classes. However, RELIEF does not help with removing redundant features. As long as features are deemed relevant to the class concept, they will be selected even though many of them are highly correlated with each other (Kira & Rendell, 1992).

FOCUS (Almuallim & Dietterich, 1991) is a straightforward filtering algorithm for noise-free, multi-class Boolean data. It exhaustively examines all subsets in order of size to find the minimal subset necessary for concept learning. FOCUS has a time complexity of $O(n^p)$ for p relevant features among n total features and is therefore impractical for high-dimensional data sets.

SCRAP (Raman, 2003) implements a sequential search filter approach that is able to detect relevant features in multi-class problems. SCRAP divides the entire set of training examples into neighbourhoods that are groups of similar examples tagged with the same class label and identifies the features that are required to discriminate between adjacent neighbourhoods containing examples belonging to a different class. However, similarly to RELIEF, SCRAP does not consider dependencies among features. Furthermore, this method cannot guarantee that selected features are sufficient to discriminate among all classes (i.e., the completeness and consistency of the learned theory) since it analyses only the centres of pairs of adjacent neighbourhoods.

2.2 Feature Reduction

Feature reduction and elimination of redundant features have been studied to a much lesser extent. A definition of redundancy follows from reducts in rough sets theory (Modrzejewski, 1993), in which Boolean features are redundant if their removal does not change the set of example-pairs having the same value for each feature. In this paper our main inspiration comes from (Lavrač et al., 1999), who situate their work in the setting of inductive

logic programming (ILP). The primary aim of their REDUCE algorithm is to detect which features (or literals, in the ILP setting) are redundant for learning and exclude them in order to reduce the hypothesis space. They prove that their method achieves this without compromising the existence of a complete and consistent theory for the target concept, which is assumed to be Boolean. One of their conclusions is that "the method is more effective when there are many literals and a small number of examples".

The method we propose in this paper addresses exactly this issue. By using neighbourhoods similar to those of SCRAP, and calling a variant of REDUCE on pairs of neighbourhoods of different class, we achieve further feature reduction (with a factor of more than two on different propositionalised versions of the mutagenesis ILP benchmark). Furthermore, the use of pairwise comparison leads to a very natural upgrade to multi-class problems. In the next section we present our improved multi-class feature reduction method REFER.

3. The Method

The multi-class concept learning problem can be defined as follows. Given (1) a set of training examples $E = \{e_1, e_2, \ldots, e_n\}$, each of which is tagged with a class label in $C = \{c_1, c_2, \ldots, c_r\}$; (2) a background theory *BK*; and (3) a hypothesis language *LH* that defines the space of hypotheses *SH*: find a theory $T = \{H_1, \ldots, H_r\}$ comprising hypotheses in *SH* describing a concept for each class in *C*, that is complete and consistent with respect to each class. This means that for each i, $1 \le i \le r$, $BK \cup H_i \models e^+$ for each example e^+ of class i (*completeness* property) and $BK \cup H_i \not\models e^-$ for any example e^- not of class i (*consistency* property).

In this paper we consider training examples $e_i \in E$ that are described by *m* Boolean features $f_i \in F$ where a *feature* is a mapping from examples to Booleans. REFER (see Figure 1) is a two-stage process based on the iterative partitioning of the example set into neighbourhoods (REFER-N) followed by the reduction of features among these partitioned examples (REFER-R). We now describe these two stages in turn.

```
Algorithm. REFER feature elimination
REFER (E, F) → RF
Input:
E // a set of training examples
F // a set of features
Output:
RF // the set of reduced features
Begin
RF ← ∅;
𝑬 ← REFER-N(E);
∀ E_i ∈ 𝑬
    ∀ E_j ∈ 𝑬 such that c_i ≠ c_j
        RF ← RF ∪ REFER-R(E_i,E_j,,L,RF);
return RF.
End.
```

Figure 1. Top-level pseudocode for the REFER algorithm.

3.1 REFER-N: Neighbourhood Construction

REFER-N produces a set of disjoint neighbourhoods $E_1, \ldots, E_w$ ($w \le n$), forming a set partition of the training set *E*, such that each neighbourhood E_i contains a subset of examples belonging to one class $c_i \in C$ only. Each neighbourhood is uniquely identified by two examples. The first example is where the neighbourhood construction started and the second one is the termination point. Let $e_s \in E$ be a random starting example for the construction of a neighbourhood. REFER-N finds a corresponding termination point, the closest example in $e_t \in E$ tagged with a different class label, referred to here as the *point of class change*. The neighbourhood $E(e_s, e_t)$ contains the set of training examples $e_{s_1}, e_{s_2}, \ldots, e_{s_k}$ such that:

- $class(e_s) = class(e_{s_1}) = \ldots = class(e_{s_k})$,
- $distance(e_s, e_{s_h}) \le distance(e_s, e_t) \quad \forall h = 1, \ldots, k$,

where the distance between two examples is computed as the *Hamming distance*, that is, the number of features whose values differ between the two examples. The neighbourhood construction proceeds in $E \setminus E(e_s, e_t)$ by considering the last point of class change as the current starting point and the process is repeated until the entire set of training examples is partitioned in neighbourhoods. This process is illustrated in Figure 2 on a two-dimensional continuous instance space.

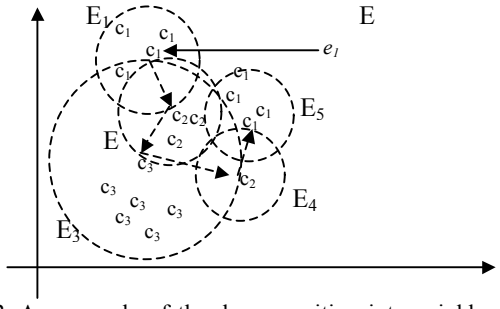

Figure 2. An example of the decomposition into neighbourhoods of the training set *E* starting from the example e_1.

3.2 REFER-R: Coverage-Based Feature Reduction

Let E_l and E_m be a pair of neighbourhoods in *E* of different classes ($c_l \ne c_m$). The goal is to detect which features $f \in F$ describing examples in $E_l \cup E_m$ are redundant for discriminating between the classes c_l and c_m. The definition of redundancy we use, following (Lavrač et al., 1999), is based on coverage among features.

Formally, a feature $f \in F$ *covers* a feature $g \in F$ with respect to $E_l \cup E_m$ if $T(g) \subseteq T(f)$ and $F(g) \subseteq F(f)$, where $T(f)$ ($T(g)$) is the set of all examples $e_i \in E_l$ such that f (g) has the value *true* for e_i and $F(f)$ ($F(g)$) is the set of all exam-

```
Algorithm. REFER-R feature elimination
REFER-R(E1, Em, F, R) → RF1,m
Input:
   E1   // a set of examples of class c1
   Em,  // a set of examples of class cm
   F    // a set of features
   R    // a set of features already detected
        // as non-redundant
Output:
RF1,m, the set of non redundant features
Begin
RF1,m ← F \ R;
∀ li ∈ RF1,m  {
  if fi has value false for each example ei∈E1
        then    eliminate fi from RF1,m;
  if fi has value true for each example ei∈Em
        then    eliminate fi from RF1,m;
  if fi is covered by any fj ∈ RF1,m
        then    eliminate fi from RF1,m;
}
∀ fi ∈ RF1,m
  if fi is covered by any fj ∈ R
        then    eliminate fi from RF1,m
return RF1,m
End.
```

Figure 3. REFER-R feature elimination algorithm.

ple $e_j \in E_m$ such that $f(g)$ has the value *false* for e_j. The intuition is that a feature f is better than another feature g for distinguishing c_l from c_m if f is true for at least the same c_l examples as g, and false for at least the same c_m examples as g. The implicit assumption is that class c_l is the positive class we are trying to describe.

This suggests the notion of *useless features*, those for which $T(f) = \emptyset$ or $F(g) = \emptyset$. Such features can be immediately removed from the set of features F regardless of the properties of other features. Furthermore, a feature $g \in F$ is defined to be *redundant* if there exists another feature $f \in F$ ($f \neq g$) such that f covers g. Therefore, redundant features can be eliminated in a preprocessing step *without compromising the existence of a complete and consistent theory* according to the theorem found in (Lavrač et al., 1999), the basis of the algorithm REDUCE.

REFER applies REFER-R (see Figure 3), a revised version of the algorithm REDUCE, to find a set of non-redundant features $RF_{l,m}$ between the neighbourhoods E_l and E_m. In particular, REFER-R takes into account the set of features R that have been already identified as non-redundant in previous neighbourhood-pair comparisons. This allows us to find a smaller set of non-redundant features by preferring features among those that have been already selected, instead of introducing new ones. REFER-R first identifies the non-redundant features between E_l and E_m, considering only the features in F not yet selected as non-redundant in previous neighbourhood-pair comparisons. Afterwards, it prunes the resulting set considering R.

3.3 Finding Non-Redundant Features

A neighbourhood decomposition $E_1, ..., E_w$ of the example set E can be represented by a *graph of neighbourhoods* G = (N, A), where N is the set of nodes n_i representing each neighbourhood E_i and A is the set of arcs connecting nodes in N. Each node $n_i \in N$ is connected only to *every other* node $n_j \in N$ that corresponds to a neighbourhood of a different class (see Figure 4a). Consequently, for each arc between a pair of nodes (n_i, n_j) in N tagged with a different class label, REFER detects the set of all non-redundant features discriminating between examples in n_i and n_j, and vice-versa, according to the REFER-R algorithm. We tag the arc with this set.

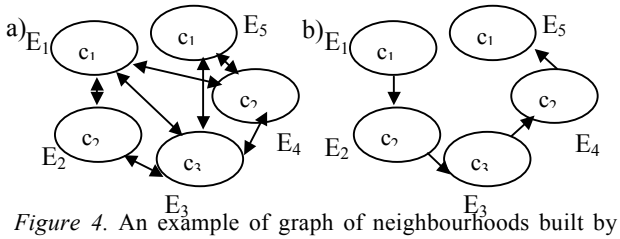

Figure 4. An example of graph of neighbourhoods built by (a) REFER and (b) SCRAP.

Where the features represent only positive literals, such as in some propositionalised data, the implicit assumption is that any negation is eventually introduced in learning. Then, there is an effective comparison between f and g as well as not-f and not-g. If negated features are supplied explicitly, all possible combinations are evaluated.

Example. Consider a pair of neighbourhoods, E_1 = {e_1, e_2, e_3} and E_2 = {e_4, e_5, e_6, e_7} as follows:

E	f_1	f_2	f_3	f_4	f_5	f_6	f_7	C	*Neighbourhood*
e_1	1	1	0	0	1	0	1	c_1	E_1
e_2	0	0	1	1	1	0	1	c_1	
e_3	0	1	1	0	1	0	0	c_1	
e_4	0	0	0	1	0	1	0	c_2	E_2
e_5	1	0	1	1	0	0	0	c_2	
e_6	1	1	1	1	1	1	0	c_2	
e_7	0	0	1	1	0	1	1	c_2	

By comparing E_1 against E_2, REFER-R discovers that only features f_5 and f_7 are non-redundant. In the opposite direction, when REFER-R compares E_2 against E_1, it identifies the features f_1, f_4 and f_6 as non-redundant. Therefore REFER-R discovers that features f_2 and f_3 are redundant in discriminating between E_1 and E_2.

In contrast, SCRAP builds a graph of neighbourhoods where each node is connected only to an *adjacent* node associated with a different class (see Figure 4b). Because of this, SCRAP does not guarantee that selected features are sufficient to discriminate among all classes. On the other hand, we can prove that given G = (N, A), the graph of neighbourhoods produced by REFER-N from a training set E and F', the union of all non-redundant features detected by applying the REFER-R algorithm to each pair of nodes in N with different class labels, we can learn a complete and consistent theory using only features in F'.

Theorem: Given L_H, an hypothesis language rich enough to allow for a theory T, that is complete and consistent for

each class, to be learned from a training set E, a set of features F and E_1, …, E_w a neighbourhood decomposition of E. A complete and consistent theory T can be found using only features from the set $F' \subseteq F$ if and only if for each possible pair of examples $(e_i, e_j) \in (E_l, E_m)$ with $c_l \neq c_m$, there exists at least one feature $f' \in F'$ such that $e_i \in T(f')$ and $e_j \in F(f')$.

Proof. *Necessity*: Suppose that a pair of examples $(e_i, e_j) \in (E_l, E_m)$ with $c_l \neq c_m$ exists such that there is no feature $f' \in F$ for which $e_i \in T(f')$ and $e_j \in F(f')$. This implies that no rule involving features in F' could discriminate between e_i and e_j and a description which is complete and consistent with respect to the class c_i cannot be found.

Sufficiency: Let $G = (N, A)$ be the graph corresponding to the neighbourhood decomposition of E. Consider an arbitrary example $e \in E$. We can build a description of e from all those features F_e appearing on all arcs of G connecting its containing node, since we have associated with these arcs the set of all features discriminating e against every other example of a different class, and vice-versa. This description is exactly the conjunction of all those features that are true for e with the negation of each of those features that are false for e. This description is then true for e and no other example. Now consider all those examples in the neighbourhood (and therefore of e's class). We build a description for those examples by constructing a disjunction of the descriptions for each example. This description is true for each of these examples and no other example of a different class. Similarly, we can build a description of each class as a whole by taking a disjunction of such descriptions for each neighbourhood. We now have a complete and consistent hypothesis for each class.

3.4 Computational Complexity

In this section we analyse the time complexity of the REFER algorithm. We consider the average case to be such that the example set E is decomposed into w equal disjoint subsets $E_1 \cup \ldots \cup E_w$, assuming E is distributed appropriately with respect to class. First we consider the average-case computational complexity of REFER-N for n training examples and m features. Consider the construction of the first neighbourhood E_1 with a random starting point e_s. This requires the calculation of the Hamming distance (complexity $O(m)$ per example) between e_s and each other example in E. This has a time complexity $O(m(n–1))$. REFER-N determines the changing class point e_c, and, without further calculation, extracts the first neighbourhood E_1. The example set for the next iteration is therefore $E \setminus E_1$, of size $n - (n/w)$. REFER-N iterates with the new example set. Therefore, the entire REFER-N process requires an average time complexity $O(nmw)$.

REFER-R is an extension of the algorithm REDUCE, but comparing each neighbourhood pair tagged with different class labels. For each comparison, REFER-R has a average time complexity of $O(m^2n/w)$. Therefore, REFER has an average combined time complexity of $O(nmw + m^2hn/w)$ with $h \leq w^2 - w$.

4. Experimental Results

REFER was implemented in Java and empirically evaluated in three different experimental settings.

4.1 Mutagenesis ILP dataset

This dataset concerns the problem of identifying mutagenic compounds (Muggleton et al., 1989) and has been extensively used as an ILP benchmark. We considered, following related experiments in the literature, the regression-friendly dataset of 188 molecules. The dataset consists of three relations, describing molecules, atoms and bonds, and the goal is to identify the mutagenic molecules. We preprocessed and propositionalised the dataset using SINUS (Krogel et al., 2003). We obtained four different datasets varying the background knowledge (Srinivasan et al., 1999) and SINUS parameters. The settings of the experiments are reported in Table 1.

The results obtained by ten-fold cross-validation are reported in Table 2. We furthermore compared REFER with REDUCE in order to show the advantages in the use of the neighbourhoods. Figure 5 presents a graphical comparison between the systems in terms of the number of features that are considered non-redundant with respect to the original set of features. In addition, Table 3 reports the average predictive accuracy on the test set (not used by REFER or REDUCE) obtained running different learning systems, specifically, C4.5 (Quinlan, 1993), Naïve Bayes, k-nearest neighbours, the RIPPER rule learner (Cohen, 1995) and a support vector machine (Keerthi et al. 2001). Results show that in general REFER selects half the number of features selected by REDUCE without accuracy decreasing. Not all the accuracy results are directly comparable with results reported in the literature. However, M2 is equivalent to the *BK2* setting (see Table 4). Despite the significant reduction in the number of features, the accuracy remains competitive with the performance of established multi-relational data mining algorithms for this domain.

Table 1. The background knowledge and SINUS settings used to generate four propositionalised versions of Mutagenesis.

Setting	*M1*	*M2*	*M3*	*M4*
Instances produced	1692	1692	1692	1692
Features produced	1016	2114	3986	13118
SINUS parameters (L, V, T)	3, 3, 20	3, 3, 20	3, 3, 20	4, 4, 20
inda and ind1	yes	yes	yes	yes
bonds	yes	yes	yes	yes
atom element and type	yes	yes	yes	yes
atom charge	no	yes	yes	yes
lumo and logp	no	yes	yes	yes
2D molecular structures	yes	no	yes	yes

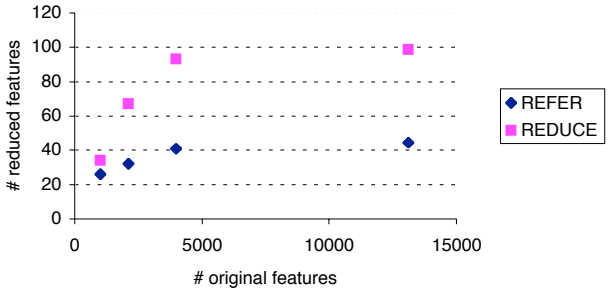

Figure 5. Comparing REFER and REDUCE regarding the average number of reduced features.

Table 2. Average numbers of features, neighbourhoods and running time performed by REFER for the Mutagenesis datasets.

	# features	*# features after reduction*	*# neighbourhoods*	*running time (s)*
M1	1016	25.9	16	1.10
M2	2114	32.1	17	9.33
M3	3986	40.9	26	40.30
M4	13118	44.4	27.1	608.17

Table 3. Accuracy for the Mutagenesis datasets reduced with REFER and REDUCE, respectively.

Dataset/System		JRIP	NB	KNN	C4.5	SVM
M1	REFER	85.58	87.22	82.98	84.53	86.14
	REDUCE	86.14	83.53	81.46	84.01	85.64
M2	REFER	87.28	87.25	87.77	89.91	89.88
	REDUCE	85.70	84.06	87.25	89.91	88.33
M3	REFER	85.08	84.06	84.53	84.56	86.66
	REDUCE	85.08	86.69	84.53	84.56	84.03
M4	REFER	80.98	82.19	82.09	83.30	86.19
	REDUCE	82.98	84.15	80.13	83.30	86.90

Table 4. Accuracy results on Mutagenesis by other learners (*BK2* setting, ten-fold cross-validation), from (Ceci et al., 2003).

System	PROGOL	FOIL	TILDE	MRDTL	1BC	MR-SBC
Accuracy	86	83	85	88	87.2	89.9

4.2 UCI Datasets

In the second experimental setting, 13 UCI datasets have been considered. In particular, we selected the following datasets containing only discrete attributes: Audiology, Bridge, Car, Flare-1066 (class C), Flare-1066 (class M), Flare-323 (class C), Flare-323 (class M), Mushroom, Nursery, Post-operative and Tic-tac-toe. In addition, we considered PimaF and YeastF, with continuous attributes discretised using equal width bins. Each dataset was transformed into a binary representation and evaluated by means of a ten-fold cross-validation. All experiments were performed using the same folds. We chose three feature selection methods for comparison with REFER, namely RELIEFF, the variant of RELIEF for multi-class problems, CFS and LVF. CFS uses a correlation-based

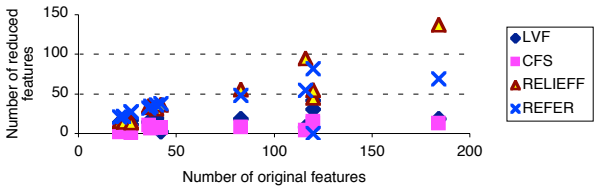

Figure 6. The average number of reduced features with respect to the original number of features on UCI datasets.

heuristic to evaluate features. This heuristic takes into account the utility of individual features for predicting the class along with the level of intercorrelation among them (Hall, 2000). LVF makes probabilistic choices to guide the search for the best subset of filtered features (Liu & Setiono, 1996). For RELIEFF, the parameter *K* is set to 5 (neighbours) and *M* is set to 30 (instances).

Figure 6 shows the number of selected features against the original number of features (before feature selection). It demonstrates that in general REFER is a conservative approach but is more selective than RELIEFF. On the other hand, REFER is considerably faster than any other feature selection method used in our comparison (see Table 5).

Table 5. Average running times of LVF, CFS, RELIEFF and REFER over ten folds. The results were recorded on an Intel Pentium IV 1.4 GHz PC running Windows XP.

Dataset	*# instances*	*# features*	*Time (s)*			
			LVF	*CFS*	*RELIEFF*	*REFER*
Audiology	398	184	3.37	0.80	3.84	**0.72**
Bridge	108	83	0.89	0.38	0.67	**0.22**
Car	1728	21	1.94	**0.44**	15.92	0.50
Flare1066/C	1066	40	2.62	**0.48**	11.51	0.61
Flare1066/M	1066	42	0.82	0.51	11.63	**0.20**
Flare323/C	323	37	0.72	0.38	1.19	**0.12**
Flare323/M	323	36	0.80	0.39	1.25	**0.21**
Mushroom	8124	116	29.48	5.30	1838.36	**1.66**
Nursery	12960	27	34.24	**1.64**	1038.31	20.38
Post-operative	90	23	0.33	0.30	0.32	**0.08**
Tic-tac-toe	950	27	1.03	0.37	5.49	**0.20**
Pima	768	120	12.2	**1**	14.1	2.6537
Yeast	1484	120	55	**19.1**	57.1	26.7132

In addition, we applied the same five learners used in the Mutagenesis experiment. We compared the percentage of correct classifications averaged over the ten folds in the cross-validation for each algorithm-dataset combination before and after feature selection or reduction. In Table 6 we report the accuracy results. We notice that the accuracy obtained with REFER's reduced feature sets is competitive with the feature selection approaches. Secondly, the accuracy of the algorithms on the feature sets produced by REFER is consistently close to that of the original feature set. These experiments have been conducted using the Weka environment (Witten & Frank, 2000).

Table 6. Average accuracy comparison on UCI datasets for four different learners and four different feature selection/reduction methods (none means the full feature set was available to the learner). Highest accuracy results per learner are indicated in bold.

	JRIP					NB					C4.5					SVM				
	LVF	CFS	RELIEF	REFER	*none*	LVF	CFS	RELIEF	REFER	*none*	LVF	CFS	RELIEF	REFER	*none*	LVF	CFS	RELIEF	REFER	*none*
Audiology	72.0	66.6	**74.9**	72.2	*74.9*	72.1	69.2	71.8	**74.9**	*71.8*	75.1	70.6	**77.9**	74.3	*74.4*	75.2	69.2	**81.6**	79.7	*81.6*
Bridge	58.7	58.0	58.8	**59.8**	*62.5*	62.7	55.9	63.5	**68.2**	*63.4*	57.6	**70.5**	64.2	65.1	*63.2*	61.8	61.8	66.0	**68.1**	*66.0*
Car	**94.9**	70.6	93.6	93.5	*94.1*	82.4	77.8	85.8	**86.9**	*86.9*	**96.8**	77.8	92.5	94.0	*94.0*	93.1	77.8	93.3	**93.6**	*93.6*
Flare1066 C	83.0	**82.7**	83.0	82.8	*82.7*	77.7	**80.8**	72.7	74.8	*74.2*	80.9	**81.5**	80.4	80.6	*80.6*	82.7	**82.9**	82.7	82.7	*82.7*
Flare1066 M	**96.6**	**96.6**	96.4	96.5	*96.4*	**96.3**	95.2	88.3	88.6	*88.2*	**96.5**	96.4	96.0	96.1	*96.1*	**96.5**	96.4	96.4	96.4	*96.4*
Flare323 C	**87.5**	88.0	88.1	88.1	*88.1*	**87.0**	86.2	82.0	81.4	*81.4*	87.2	87.7	**85.9**	85.9	*85.9*	88.4	88.4	**87.7**	**87.7**	*87.7*
Flare323 M	**89.1**	**89.7**	**89.1**	**89.1**	*89.1*	85.8	**86.9**	83.7	83.7	*83.7*	86.3	**88.5**	87.2	87.2	*87.2*	**90.0**	89.4	89.1	89.1	*89.1*
Mushroom	**100.0**	93.0	**100.0**	**100.0**	*100.0*	93.6	93.0	**94.3**	94.3	*94.3*	**100.0**	93.0	**100.0**	**100.0**	*100.0*	99.9	92.6	**100.0**	**100.0**	*100.0*
Nursery	97.3	36.3	91.1	**98.7**	*98.7*	86.2	66.2	89.4	**92.2**	*92.9*	**99.5**	66.2	91.1	98.1	*98.3*	**93.2**	66.2	93.1	93.1	*93.1*
Post-Op	**71.1**	68.9	**71.1**	**71.1**	*71.1*	66.7	66.7	**58.9**	57.8	*58.9*	62.2	**66.7**	65.6	62.2	*62.2*	**68.9**	**68.9**	67.8	67.8	*67.8*
Tic-tac-toe	95.4	69.9	85.3	**98.1**	*97.7*	68.8	69.9	**71.4**	68.4	*68.4*	91.6	69.9	83.5	**98.7**	*98.7*	75.7	69.9	**98.3**	**98.3**	*98.3*
Pima	69.8	72.6	**73.5**	72.6	*71.1*	71.2	**74.7**	74.0	**74.7**	*74.9*	63.9	**72.0**	65.1	67.9	*67.2*	**72.6**	73.5	72.1	74.1	*74.6*
Yeast	50.9	**56.1**	50.4	49.9	*50.6*	54.7	50.0	54.7	**56.0**	*55.7*	44.3	**48.0**	43.6	45.8	*45.8*	56.1	50.4	54.6	**57.6**	*57.8*

Finally, Figure 7 shows the number of both reduced features and neighbourhoods constructed by REFER using varying starting points. We may observe that the number of reduced features as well as the number of neighbourhoods is not greatly affected by a different starting point.

4.3 Reuters-21578

To evaluate the performance of REFER on large-scale datasets, we executed it on the Reuters-21578 dataset (Lewis, 1997) consisting of a set of 21,578 articles published by Reuters. This is a well-known benchmark dataset in the field of Information Retrieval and Document Categorisation. We evaluated it using the Mod-Apte split (Yang and Liu, 1999), where the dataset is cleaned and split into a training set (7769 articles) and a test set (3019 articles). The resulting data contains articles each belonging to one or more of 90 classes. In our representation, each feature represents a word in the article and each

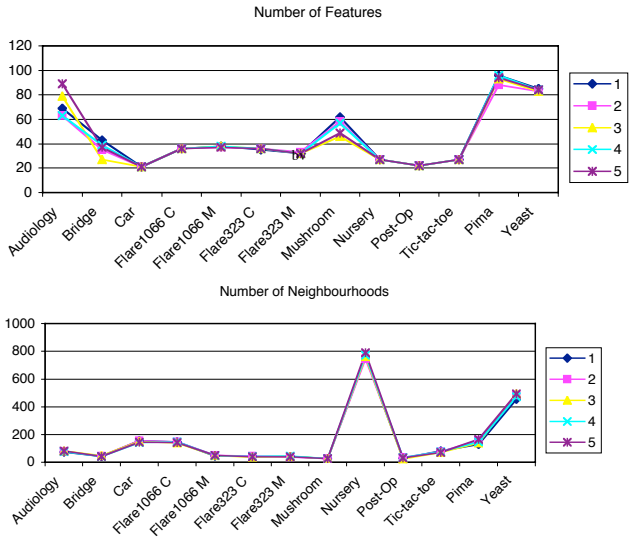

Figure 7. The number of features and neighbourhoods performed by REFER varying the starting point for 5 trials.

article has been represented in the form of a Boolean vector, recording for each word whether it occurs in the document. In order to adapt it to the single-label problem, we removed the articles with multiple classifications, as in (Schapire & Singer, 2000). We thus obtained a training set of 6577 articles and a test set of 2583 articles. After preprocessing, the total number of features was 16,582. Although the dataset could be represented in a sparse matrix representation, we did not use this representation in order to prove the applicability of our method even to large-scale datasets where the dataset is not sparse.

For this domain, the Boolean vector representation leads to a dataset of high dimensionality, giving a file size of hundreds of megabytes. One of the advantages of REFER is that it is possible to split a dataset into several smaller ones consisting of subsets of the features of the original set, run REFER on the subsets (possibly in parallel on different machines), combine them, and run REFER on the combination, without compromising the discovery of a complete and consistent theory, described in Section 3. In this way, we identified more than 90% of 16,582 features as redundant, resulting in a set of 1450 reduced features.

5. Conclusions and Future Work

In this paper we presented REFER, an efficient method for eliminating redundant Boolean features for multi-class classification tasks. The method is logically sound, in that it guarantees the existence of a complete and consistent theory using only the reduced feature set if it exists with the complete feature set. It also is efficient, requiring on average less time than the three feature selection methods we compared it with. We have demonstrated that the use of neighbourhoods increases the number of identified redundant features with more than a factor of two, in comparison with REFER's predecessor REDUCE. We have also shown that it is feasible to execute the method on a large and high-dimensional dataset, and that the method is amenable to parallel execution. REFER's computational efficiency derives from its heuristic nature and sets it

apart from exhaustive methods such as FOCUS, which do not scale up well to large sets of features.

Feature reduction, i.e., eliminating redundant features, and feature selection, i.e., identifying relevant features, are in some sense complementary approaches. In future work, we plan to study the interaction between these two approaches to obtain even smaller feature sets. We also plan to incorporate noise-handling mechanisms by allowing non-pure neighbourhoods.

References

Almuallim, H., & Dietterich, T.G. (1991). Learning with many irrelevant features. *Proc. 9th Nat. Conf. on Artificial Intelligence* (pp. 547-552). MIT Press.

Blum, A., & Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence*, *97*, 245-271.

Ceci, M., Appice, A., & Malerba, D. (2003). Mr-SBC: a Multi-Relational Naive Bayes Classifier. *Proc. 7th Eur. Conf. on Principles and Practice of Knowledge Discovery in Databases* (pp. 95-106). Springer-Verlag.

Cohen, W.W. (1995). Fast Effective Rule Induction. *Proc. 12th Int. Conf. on Machine Learning* (pp. 115-123). Morgan Kaufmann.

Dash, M., & Liu, H. (1997). Feature selection for classification. *Intelligent Data Analysis*, *1*, 131-156.

Dzeroski, S., & Lavrač, N., Eds. (2001). *Relational Data Mining*. Springer-Verlag.

Freund, Y. & Schapire, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *J. Computer and System Sciences*, *55(1)*, 119-139.

Hall, M. (2000). Correlation-based feature selection for discrete and numeric class machine learning. *Proc. 17th Int. Conf. on Machine Learning* (pp. 359-366). Morgan Kaufmann.

Keerthi, S.S., Shevade, S.K., Bhattacharyya, C., & Murthy, K.R.K. Improvements to Platt's SMO Algorithm for SVM Classifier Design. *Neural Computation*, *13(3)*, 637-649.

Kira, K., & Rendell, L.A. (1992). A practical approach to feature selection. *Proc. 9th Int. Conf. on Machine Learning* (pp. 249-256). Morgan Kaufmann.

Kohavi, R., John, G., & Pfleger, K. (1994). Irrelevant features and the subset selection problem. *Proc. 11th Int. Conf. on Machine Learning* (pp. 121–129). Morgan Kaufmann.

Kononenko, I. (1994). Estimating attributes: Analysis and extensions of RELIEF. *Proc. Eur. Conf. on Machine Learning* (pp. 171-182). Springer-Verlag.

Krogel, M., Rawles, S., Zelezny, F., Flach, P., Lavrač, N., & Wrobel S. (2003). Comparative evaluation of approaches to propositionalization. *Proc. 13th Int. Conf. on Inductive Logic Programming* (pp. 197-214). Springer-Verlag.

Langley, P. (1996). *Elements of Machine Learning*. Morgan Kaufmann.

Lavrač, N., Gamberger, D., & Jovanoski V. (1999). A study of relevance for learning in deductive databases. *J. Logic Programming*, *16*, 215-249.

Lewis, D.D. (1999). *Reuters-21578 text categorization test collection distribution 1.0*. Available at http://www.research.att.com/lewis.

Liu, H., & Setiono, R. (1996). A probabilistic approach to feature selection: A filter solution. *Proc. 13th Int. Conf. on Machine Learning* (pp. 319-327). Morgan Kaufmann.

Modrzejewski, M. (1993). Feature selection using rough sets theory. *Proc. Eur. Conf. on Machine Learning* (pp. 213-226). Springer-Verlag.

Muggleton, S. H., Bain, M., Hayes-Michie J., & Michie, D. (1989). An experimental comparison of human and machine learning formalisms. *Proc.. 6th Int. Workshop on Machine Learning*. Morgan Kaufmann.

Pagallo, G., & Haussler, D. (1990). Boolean feature discovery in empirical learning. *Machine learning*, *5(1),* 71-100.

Quinlan, J. (1993). *C4.5: Programs for machine learning*. Morgan Kaufmann.

Raman, B. (2003). *Enhancing inductive learning with feature selection and example selection.* Master thesis, Texas A & M University.

Rendell, A., & Sheshu, R. (1990). Learning hard concepts through constructive induction: Framework and rationale. *Computational Intelligence*, *6*, 247-270.

Schapire, R., & Singer, Y. (2000). A boosting-based system for text categorization. *Machine Learning*, *39(2/3)*, 135-168.

Srinivasan, A., King, R.D., & Muggleton, S. (1999). *The role of background knowledge: using a problem from chemistry to examine the performance of an ILP program*. Technical Report PRG-TR-08-99, Oxford University Computing Laboratory.

Yang, Y., & Liu, X. (1999). A re-examination of text categorization methods. *Proc. 20th ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval* (pp. 42-49). ACM Press.

Witten, I. & Frank, E. (2000). *Data mining: practical machine learning tools and techniques with Java implementations*. Morgan Kaufmann.