Feature Selection via Coalitional Game Theory

Shay Cohen
scohen@cs.cmu.edu
School of Computer Sciences, Tel-Aviv University, Tel-Aviv, Israel

Gideon Dror
gideon@mta.ac.il
Department of Computer Sciences, Academic College of Tel-Aviv-Yaffo, Tel-Aviv, Israel

Eytan Ruppin
ruppin@post.tau.ac.il
School of Computer Sciences, Tel-Aviv University, Tel-Aviv, Israel, and Department of Computer Sciences, Academic College of Tel-Aviv-Yaffo, Tel-Aviv, Israel

We present and study the contribution-selection algorithm (CSA), a novel algorithm for feature selection. The algorithm is based on the multiperturbation shapley analysis (MSA), a framework that relies on game theory to estimate usefulness. The algorithm iteratively estimates the usefulness of features and selects them accordingly, using either forward selection or backward elimination. It can optimize various performance measures over unseen data such as accuracy, balanced error rate, and area under receiver-operator-characteristic curve. Empirical comparison with several other existing feature selection methods shows that the backward elimination variant of CSA leads to the most accurate classification results on an array of data sets.

1 Introduction

Feature selection refers to the problem of selecting input variables, otherwise called features, that are relevant to predicting a target value for each instance in a data set. Feature selection can be used to rank all potentially relevant input variables or to build a good classifier, and each task may lead to a different methodological approach (Blum & Langley, 1997; Kohavi & John, 1997). Feature selection has several potential benefits: defying the curse of dimensionality to enhance the prediction performance, reducing measurement and storage requirements, reducing training and prediction times, providing better understanding of the process that generated the data, and allowing data visualization. This letter focuses on the first issue: selecting input variables in an attempt to maximize the performance of a classifier on previously unseen data. Clearly, this would not necessarily produce the most compact set of features.

Feature selection is a search problem, where each state in the search space corresponds to a subset of features. Exhaustive search is usually intractable, and methods to explore the search space efficiently must be employed. These methods are often divided into two main categories: filter methods and subset selection methods. The algorithms in the first category rank each feature according to some measure of association with the target, such as mutual information, Pearson correlation or χ^2 statistic, and features with high-ranking values are selected. A major disadvantage of filter methods is that they are performed independent of the classifier, and it is not guaranteed that the same set of features is optimal for all classifiers. In addition, most filter methods disregard the dependencies between features, as each feature is considered in isolation.

The second category, the subset selection methods, has two types of algorithms. Embedded algorithms select the features through the process of generating the classifier, such as regularization methods such as grafting (Perkins, Lacker, & Theiler, 2003), Gram-Schmidt methods (e.g., Stoppiglia, Dreyfus, Dubois, & Oussar, 2003; Rivals & Personnaz, 2003), or methods specific for support vector machines (e.g., Weston et al., 2000; Guyon, Weston, Barnhill, & Vapnik, 2002). Wrapper algorithms treat the induction algorithm as a black box and interact with it in order to perform a search for an appropriate features set using search algorithms such as genetic algorithms or hill climbing (Kohavi & John, 1997). Although wrapper methods are successful in feature selection, they may be computationally expensive, because they require retraining a classifier on data with a large number of features. (For a survey of the current methods in feature selection, see Guyon & Elisseeff, 2003.)

In this letter, we recast the problem of feature selection in the context of coalitional games, a notion from game theory. This perspective yields an iterative algorithm for feature selection, the contribution-selection algorithm (CSA), intent on optimizing the performance of the classifier on unseen data. The algorithm combines the filter and wrapper approaches, where the features are reranked on each step by using the classifier as a black box. The ranking is based on the Shapley value (Shapley, 1953), a well-known concept from game theory, to estimate the importance of each feature for the task at hand, specifically taking into account interactions between features. Due to combinatorial constraints, the Shapley value cannot be calculated precisely and is estimated by the multiperturbation Shapley analysis (MSA) (Keinan, Sandbank, Hilgetag, Meilijson, and Ruppin, 2004, 2006). Furthermore, since the classifier is trained and tested extensively, the classifier used by CSA must be fast in both training and testing phases. This requirement can be moderated by parallel processing.

Throughout the letter, we use the following notations. Three disjoint sets containing independently and identically distributed (i.i.d.) sampled instances of the form $(\mathbf{x}_k, y_k)$ are denoted by T, V, and S, representing the training set, validation set, and test set, respectively, where $\mathbf{x}_k \in \mathbf{R}^n$ denotes the kth instance and y_k is the target class value associated with it. Given an induction algorithm and a set of features $S \subseteq \{1, \ldots, n\}$, $f_S(x)$ stands for a classifier constructed from the training set using the induction algorithm, after its input variables were narrowed down to the ones in S. Namely, $f_S(x)$ labels each instance of the form $(x_{i_1}, \ldots x_{i_{|S|}})$, $i_j \in S$, $1 \leq j \leq |S|$ with an appropriate class value. The task of feature selection is to choose a subset S of the input variables that maximizes the performance of the classifier on the test set. In what follows, we focus on optimizing classifier accuracy, although we could as easily optimize other performance measures such as the area under the receiver operator chracteristic or balanced error rate.

The rest of this letter is organized as follows. Section 2 introduces the necessary background from game theory and justifies the use of game theory concepts for the task of feature selection. It also provides a detailed description of the CSA algorithm. Section 3 provides an empirical comparison of CSA with several other feature selection methods on artificial and real-world data sets, accompanied by an analysis of the results, showing that the backward elimination version of CSA is significantly superior to other feature selection methods considered. Section 4 discusses the empirical results and provides further insights into the success and failure of the backward elimination version of the CSA algorithm. Section 5 summarizes the results.

2 Classification as a Coalitional Game

Cooperative game theory introduces the concept of coalitional games in which a set of players is associated with a real function that denotes the payoff achieved by different subcoalitions in a game. Formally, a coalitional game is defined by a pair (N, v) where $N = \{1, \ldots, n\}$ is the set of all players and $v(S)$, for every $S \subseteq N$, is a real number associating a worth with the coalition S. Game theory further pursues the question of representing the contribution of each player to the game by constructing a value function, which assigns a real value to each player. The values correspond to the contribution of the players in achieving a high payoff.

The contribution value calculation is based on the Shapley value (Shapley, 1953). An intuitive example of the potential use of the Shapley value is a scenario of a production machine in a factory composed of numerous components. During its operation, the machine undergoes various malfunctions from time to time. In each such malfunction, the normal activity of a subset of its components may be shut down, resulting in reduction in the machine's productivity and output. Based on annual data of these multicomponent failures and their associated production drops, the Shapley value provides a fair and efficient way to distribute the responsibility for the machine's failure among its individual components, identifying the ones needing the most attention and maintenance, while considering their possible intricate functional interactions. In reference to feature selection, the machine is analogous to the predictor and its components to the classification features. The task of feature selection then involves identifying the features contributing most to the classification in hand.

The Shapley value is defined as follows. Let the marginal importance of player i to a coalition S, with $i \notin S$, be

$$\Delta_i(S) = v(S \cup \{i\}) - v(S). \tag{2.1}$$

Then the Shapley value is defined by the payoff

$$\Phi_i(v) = \frac{1}{n!} \sum_{\pi \in \Pi} \Delta_i(S_i(\pi)), \tag{2.2}$$

where Π is the set of permutations over N and $S_i(\pi)$ is the set of players appearing before the ith player in permutation π. The Shapley value of a player is a weighted mean of its marginal value, averaged over all possible subsets of players.

Transforming these game theory concepts into the arena of feature selection, in which one attempts to estimate the contribution of each feature in generating a classifier, the players are mapped to the features of a data set and the payoff is represented by a real-valued function $v(S)$, which measures the performance of a classifier generated using the set of features S.

The use of Shapley value for feature selection may be justified by its axiomatic qualities:

Axiom 1 *(normalization or Pareto optimality).* *For any game (N, v) it holds that $\sum_{i \in N} \Phi_i(v) = v(N)$.*

In the context of feature selection, this axiom implies that the performance on the data set is divided fully between the different features.

Axiom 2 *(permutation invariance or symmetry).* *For any (N, v) and permutation π on N, it holds that $\Phi_i(v) = \Phi_{\pi(i)}(\pi v)$.*

This axiom implies that the value is not altered by arbitrarily renaming or reordering the features.

Axiom 3 *(preservation of carrier or dummy property).* *For any game (N, v) such that $v(S \cup \{i\}) = v(S)$ for every $S \subseteq N$, it holds that $\Phi_i(v) = 0$.*

This axiom implies that a dummy feature that does not influence the classifier's performance indeed receives a contribution value 0.

Axiom 4 *(additivity or aggregation).* *For any two games (N, v) and (N, w), it holds that $\Phi_i(v + w) = \Phi_i(v) + \Phi_i(w)$ where $(v + w)(S) = v(S) + w(S)$.*

This axiom applies to a combination of two different payoffs based on the same set of features. For a classification task, these may be, for example, accuracy and area under the receiver operator characteristic curve or false-positive rate and false-negative rate. In this case, the Shapley value of a feature that measures its contribution to the combined performance measure is the sum of the corresponding Shapley values. The linearity of the Shapley value is a consequence of this property. If the payoff function v is multiplied by a real number α, then all Shapley values are scaled by α: $\Phi_i(\alpha v) = \alpha \Phi_i(v)$. In other words, multiplying the performance measure by a constant does not change the ranking of the features, a vital property for any scheme that ranks features by their importance.

Since it was introduced, the Shapley value has been successfully applied to many fields. One of the most important applications is with cost allocation, where the cost of providing a service should be shared among the different receivers of that service (Shubik, 1962; Roth, 1979; Billera, Heath, & Raanan, 1978). This use of the Shapley value has received recent attention in the context of sharing the cost of multicast routing (Feigenbaum, Papadimitriou, & Shenker, 2001). In epidemiology, the Shapley value has been used as a means to quantify the population impact of exposure factors on a disease load (Gefeller, Land, & Eide, 1998). Other fields where the Shapley value has been used include politics (starting from the strategic voting framework introduced by Shapley and Shubik, 1954), international environmental problems, and economic theory (see Shubik, 1985, for discussion and additional references).

2.1 Estimating Features Contribution Using MSA. The calculation of the Shapley value requires summing over all possible subsets of players, which is impractical in typical feature selection problems. Keinan et al. (2005) have presented an unbiased estimator for the Shapley value by uniformly sampling permutations from Π.[1] Still, the estimator considers both large and small features sets to calculate the contribution values. In our feature selection algorithm, we use the Shapley value heuristically to estimate the contribution value of a feature for the task of feature selection. Since in most realistic cases, we assume that the size d of significant interactions between features is much smaller than the number of features, n, we will limit

[1] The estimation in Keinan et al. (2005) is used in a different context: to analyze the functional contribution in artificial and biological networks. However, the method to estimate the Shapley value is valid for our purpose as well.

ourselves to calculating the contribution value from permutations sampled from the whole set of players, with d being a bound on the permutation size,

$$\varphi_i(v) = \frac{1}{|\Pi_d|} \sum_{\pi \in \Pi_d} \Delta_i(S_i(\pi)), \tag{2.3}$$

where Π_d is the set of sampled permutations on subsets of size d.

Limiting the extent of interactions taken into account is not uncommon in feature selection methods. Most filter methods are equivalent to using $d = 1$, where no feature interactions are taken into account. Explicit restriction on the level of interactions also characterizes several ensemble methods, for example, Random Forests (Breiman, 2001), where $d \simeq \sqrt{n}$ is usually suggested.

The use of bounded sets, coupled with the method for the Shapley value estimation, yields an efficient and robust way to estimate the contribution of a feature to the task of classification. (For a detailed discussion of the MSA framework and its theoretical background see Keinan et al., 2004, 2005. The MSA toolbox can be downloaded online from http://www.cns.tau.ac.il/msa/.)

2.2 The Contribution-Selection Algorithm. The CSA is iterative in nature and can adopt a forward selection or backward elimination approach. Its backward elimination version, which overall yields better prediction accuracy (see section 3.2) is described in detail in Figure 1. In the backward elimination version, using the subroutine contribution, the CSA ranks each feature according to its contribution value and then eliminates e features with the lowest contribution values (using the subroutine elimination). It repeats the phases of calculating the contribution values of the currently remaining features and eliminating new features, until the contribution values of all candidate features exceed a contribution threshold Δ. Forward selection CSA works in a similar manner, selecting on each iteration s features with highest contribution values, as long as their contribution values exceed some threshold.

The algorithm, without further specification of the contribution subroutine, is a known generalization of filter methods. However, the main idea of the algorithm is that the contribution subroutine, unlike common filter methods, returns a contribution value for each feature according to its role in improving the classifier's performance, which is generated using a specific induction algorithm and in conjunction with other features. Using the notation in section 2 and assuming that one maximizes the accuracy level of the classifier, the contribution subroutine for backward selection calculates the contribution values φ_i by equations 2.1 and 2.3, where the payoff function $v(S)$ is simply the validation accuracy of the base classifier $f_S(x)$

```
Contribution-Selection-Algorithm-Backward-Elimination($F$ ; $t, d, \Delta, e$)
  1. $c := F$
  2. for each $f \in F \backslash c$
      2.1. Sample permutations set $\{\pi_1, ..., \pi_t\}$ over $F \backslash c$
      2.2. $C_f :=$ contribution($f$, $c$ ; $d$) $= \frac{1}{t}\sum_{j=1}^{t} \Delta_f(S_f(\pi_j))$
  3. if $\max_f C_f < \Delta$
      3.1. $c := c \setminus$ elimination($\{C_f\}$ ; $e$, $\Delta$)
      3.2. goto 2
     else
      3.3. return c
```

Figure 1: The contribution-selection algorithm in its backward elimination version. F is the input set of features. t, d, Δ, and e are hyperparamters: $t = |\Pi_d|$ is the number of permutations sampled (see equation 2.3), d is the maximal permutation size for calculating the contribution values, Δ is a contribution value threshold, and e is the number of features eliminated in each phase. The variable c represents the set of candidate features for selection. The contribution subroutine calculates the contribution value of feature f according to equation 2.3, where f corresponds to the ith player. The elimination subroutine eliminates at most e features with lowest contribution values that do not exceed Δ. In the forward selection version, the elimination subroutine is replaced with a selection subroutine, which selects s features in each phase, and the halting criterion is changed accordingly.

trained on the training set T,

$$v(S) = \frac{|\{x | f_S(x) = y, (x, y) \in V\}|}{|V|}.$$

The case $S = \phi$ is handled by returning the fraction of majority class instances. The maximal permutation size d has an important role in deciding the contribution values of the different features and should be selected in a way that ensures that different combinations of features that interact together are inspected. Its impact is demonstrated in section 3.

The number of eliminated features e for the elimination subroutine controls the redundancies of the eliminated features; the higher e is, the more likely it is that correlated features with redundant contribution will be eliminated. Although $e = 1$ minimizes the redundancy dependencies of the features, increasing e accelerates the algorithm's convergence and provides some regularization, as has been verified experimentally. The algorithm's halting criterion depends on Δ, which designates a trade-off between the number of selected features and the performance of the classifier on the validation set. With the backward elimination version, choosing $\Delta = 0$ means that CSA eliminates features as long as there exist features that are unlikely to improve the classifier's performance. Increasing Δ has the opposite effect on the size of the final set of features. The naive halting criterion that terminates feature elimination when no further performance gain is expected—when the exclusion of no feature enhances the performance—is entirely different. For example, during CSA backward elimination, there are occasionally features with negative contribution values: once they are eliminated, there is no performance improvement. Still the removal of such features tends to increase the generalization of the classifier by the mere reduction of its complexity. Indeed, testing this naive halting criterion has verified that it leads to considerably inferior performance levels.

3 Results

3.1 Experiments with Artificial Data. In order to demonstrate the algorithm's behavior, we generated a data set that consists of nine features. The first three features are binary. The target labels are taken as the parity function of these three features. The other six features are correlated with the target by setting them to the target values and adding a random value taken from the normal distribution with expectancy 0 and standard deviation $\sigma = 1$. A simple calculation shows that the correlation between each of these six features and the target is $\sqrt{(1+\sigma^2)^{-1}} \simeq 0.71$, and the correlation between the sum of these features and the target is $\sqrt{(1+\sigma^2/6)^{-1})} \simeq 0.92$. The mean accuracy of a classifier that outputs the sign of the sum of these six features is 0.84, which will be considered a baseline accuracy (the accuracy of C4.5 classifier that uses all nine features is 0.82). The data set consists of 200 training examples and 100 test examples.

Using this data set, we inspected the features selected and the performance of several feature selection methods. We used Random Forests (Breiman, 2001), mutual information, Pearson correlation, regular backward wrapper method (eliminating one feature at a time), regular forward wrapper method (selecting one feature at a time), and CSA in both backward elimination and forward selection using C4.5 as base learner. CSA was run with $s = 1$, $e = 1$, $d = 3$, and $t = 20$.[2] No optimization was performed on these hyperparameters. In all ranking methods (Random Forests, mutual

[2] While CSA evaluated 540 permutations, this toy example can be solved by an exhaustive search requiring only $2^9 - 1 = 511$ evaluations.

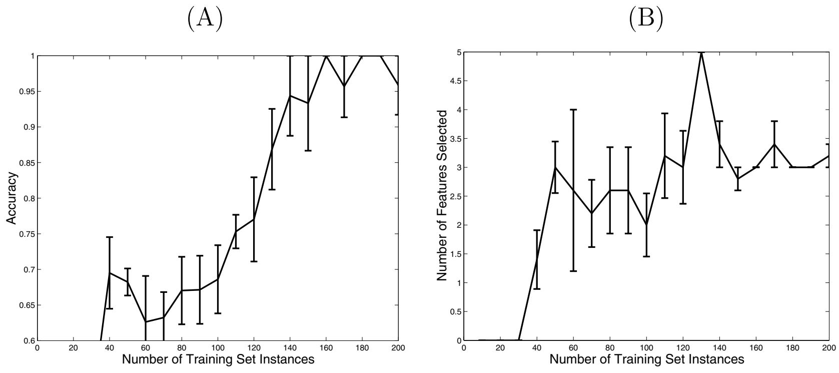

Figure 2: Behavior of backward selection CSA on the artificial data set. (A) The accuracy of the backward selection CSA changes with respect to *n*, the number of training set instances. (B) The average number of features selected as a function of *n*.

information, and Pearson correlation), the first three features were ranked as least informative features. As expected, the accuracy obtained by using these methods did not exceed the baseline accuracy 0.84. Forward selection CSA together with backward wrapper and forward wrapper did not select any of the first three most relevant features. Yet backward elimination CSA chose the three features for classification and achieved 100% accuracy. The reason for this behavior is that the other six features were too confusing for the classifier with most algorithms. With decision trees, the first three features will always be used in deep nodes of the tree due to the greedy criterion used by decision tree algorithm to select a feature for node splitting. Therefore, the inclusion or removal of any one of these features will have less influence on the classifier's accuracy than any of the six features linearly correlated with the target.

We further tested the behavior of CSA with respect to the number of training instances, *n*. To this end, we produced training sets of different values of *n* and ran the algorithm on each sample. For each value of *n*, five different training sets were sampled and scored, and the average accuracy and average number of features selected were computed. The results for the backward selection CSA are described in Figure 2. As can be seen, for $n \geq 140$, many instances of the backward elimination CSA identify the three salient features and achieve perfect categorization of the test examples.

3.2 Experiments with Real-World Data. To test CSA empirically, we ran a number of experiments on seven real-world data sets with number of features ranging from 278 to 20,000 (see Table 1):

Table 1: Description of Data Sets Used.

Name	Number of Classes	Number of Features	Training Set Size	Testing Set Size
Reuters1	3	1579	145	145
Reuters2	3	1587	164	164
Arrhythmia	2	278	280	140
Internet Ads	2	1558	2200	800
Dexter	2	20,000	300	300
Arcene	2	10,000	100	100
I2000	2	2000	40	22

- Following Koller and Sahami (1996), we constructed the Reuters1 data set from the Reuters-21578 document collection (Reuters, 1997). The data set consists of articles from the categories *coffee* (99 documents), *iron-steel* (137 documents), and *livestock* (54 documents). These topics do not have many overlapping words, making the task of classification easier. As a preprocessing step, we removed all words that appeared fewer than three times. Each article was then encoded into a binary vector, where each element designates whether the word appeared in the document.

- The Reuters2 data set was constructed, similar to the Reuters1 data set, from the categories *gold* (68 documents), *gross national product* (124 documents), and *reserves* (136 documents). These topics are more similar and contain many overlapping words, making the task of classification harder. For both of the Reuters data sets, our splits are not identical to the one in Koller and Sahami (1996) and contain fewer documents because we could not obtain the exact same data set.

- The Arrhythmia database from the UCI repository (Blake & Merz, 1998). The task for this database is to distinguish between normal and abnormal heartbeat. We used a version of the data that was slightly modified by Perkins et al. (2003): features that were missing in most of the instances were removed. The data set contains 237 positive instances and 183 negative instances. It can be found online at http://nis-www.lanl.gov/~simes/data/jmlr03.

- The Internet Advertisements database from the UCI repository (Blake & Merz, 1998) was collected for research on identifying advertisements in web pages (Kushmerick, 1999). The features in the database describe different attributes in a web page, such as the domain it was downloaded from, the domain that referred to it, and its size. There are two classes: each instance is either an advertisement or not an advertisement. The data set contains 2581 positive instances and 419 negative instances.

- The Dexter data set from the NIPS 2003 workshop on feature selection (Guyon, 2003). The Dexter data set is a two-class text categorization data set constructed from a subset of the Reuters data set, using documents from the category in corporate acquisitions. The data set is abundant with irrelevant features. (For exact details on how the data set was constructed, see Guyon, 2003.) The data set contains 300 positive instances and 300 negative instances. As a preprocessing step, we binarized each of the instances that originally contained the word *frequency* in each document and removed words that appeared fewer than three times. The validation set of the data served in the following experiments as T, the test set.
- The Arcene data set from the NIPS 2003 workshop on feature selection (Guyon, 2003) is a two-class categorization data set, describing mass spectrometry analysis of the blood serum of patients with a certain kind of cancer and without it. It is affluent with features and poor with data instances. (For details on how the dataset was constructed, see Guyon, 2003.) The data set contains 88 positive instances and 112 negative instances. The validation set of the data served in the following experiments as T, the test set.
- I2000, a microarray colon cancer data set by Alon et al. (1999) is a two-class categorization data set for discriminating between healthy and ill tissues in colon cancer. The data contain the expression of 2000 genes with highest minimal intensity across 62 tissues. The data set contains 31 positive instances and 31 negative instances. It has a very high features-to-instances ratio, making the task of feature selection harder.

In principle, CSA can work with any induction algorithm L. However, due to computational constraints, we focused on fast induction algorithms, or algorithms that may be efficiently combined into CSA. We experimented with Naive Bayes, C4.5, and 1NN. For each of the data sets, we measured the training set accuracy of each classifier using tenfold cross-validation on the whole set features. For each data set, all subsequent work used the induction algorithm L that gave the highest cross-validation accuracy, as detailed in Table 2.

Nine different classification algorithms were then compared on the data sets described above:

- Classification using the induction algorithm L without performing any feature selection.

Table 2: Parameters and Classifier Used with the CSA Algorithm for Each Data Set.

Data Set	Induction Algorithm (L)	s (Forward)	e (Backward)	d	t
Reuters1	Naive Bayes	1	100	20	1500
Reuters2	Naive Bayes	1	100	20	1800
Arrhythmia	C4.5	1	50	20	500
Internet Ads	1NN	1	100	20	1500
Dexter	C4.5	50	50	12	3500
Arcene	C4.5	100	100	5	10,000
I2000	C4.5	100	100	3	2000

Notes: s is the number of features selected in forward selection in each phase, e is the number of features eliminated in backward elimination in each phase, d is the permutation size, and t is the number of permutations sampled to estimate the contribution values. For an explanation of how hyperparameters were chosen, see the description of the backward CSA algorithm.

- Classification using soft margin linear SVM with the SVM^{light} package (Joachims, 1999). Data sets that had more than two classes were decomposed to a few one-versus-all binary classification problems.
- Classification using L after performing feature selection by estimation of the Pearson correlation coefficient. The number of features was selected by performing tenfold cross validation on the training set and averaging the results, each time adding more features with the highest correlation value to the current features set. After this process, the set that obtained the best result was selected.
- Classification using L after performing feature selection by estimation of mutual information. For data sets with continuous domain (Arrhythmia, Internet Ads, Arcene, and I2000), we used binning to estimate the mutual information. The number of features selected was optimized, as with the Pearson correlation coefficient.
- Classification using L after performing feature selection with Random Forests (Breiman, 2001). We used the randomForest library implementation for the R environment (Bengtsson, 2003). The number of features selected was optimized as with the Pearson correlation coefficient.
- Classification using L after performing feature selection with backward elimination CSA with $d = 1$. The number of permutations selected was large enough that each feature is sampled with high probability. This is equivalent to regular wrapper technique, in which backward elimination is used to eliminate the features that most degrades the accuracy of the classifier. This algorithm is chosen to check

whether it is sufficient to examine each feature separately for performing feature selection on the data set.

- Classification using L after performing feature selection with forward selection CSA with $d = 1$. The number of permutations selected was large enough that each feature is sampled with high probability. This is equivalent to the regular wrapper technique, in which forward selection is used to select the features that most improve the accuracy of the classifier. This algorithm is similar to that of the backward wrapper.
- Classification using L after performing feature selection with backward elimination CSA and parameters as described in Table 2. The parameters d and t were chosen such that the expected number of times that each feature is sampled is higher than five. This number was chosen according to error analysis considerations of MSA (Keinan et al., 2004) and following preliminary experimentation with an artificial data set. When computation times allowed, the number of permutations sampled was much larger than the minimal value. The contribution value threshold for stopping elimination was $\Delta = 0$. No hyperparameter selection was performed on d, t, or Δ.
- Classification using L after performing feature selection with forward selection CSA and parameters as described in Table 2. The parameters d and t were chosen such that the expected number of times that each feature is sampled is higher than five. The termination of feature selection was fixed by choosing a contribution value threshold $\Delta = 0$. No hyperparameter selection was performed on d, t, or Δ.

CSA is prone to overfitting on the validation set. When the classifier's performance is always evaluated on a possibly small validation set, the curse of dimensionality appears, and irrelevant features are selected, even if the classifier itself is trained using techniques that avoid overfitting. It might seem at first that evaluating the classifier each time on a different training set and validation set split can solve the problem. However, this leads to another problem: the classifier's performance depends on the split, so the marginal contributions of the different features do not reflect their real value. In order to avoid both of these problems, we used tenfold cross validation; the training set was split into several parts, and the payoff function was evaluated by averaging a classifier's performance on the whole training set.

3.3 Feature Selection and Classification Results. Table 3 summarizes the classifiers' performance on the test set and the number of features selected in each of the experiments. The accuracy levels are the fraction of correctly classified test set instances.

Table 3: Accuracy Levels and Number of Features Selected in the Different Data Sets.

Data Set	No FS	SVM	Corr	MI	RF
Reuters1	84.1%	94.4%	90.3% (20)	94.4% (20)	96.3% (6)
Reuters2	81.1%	91.4%	88.4% (20)	90.2% (5)	87.2% (21)
Arrhythmia	76.4%	80%	71.4% (20)	70% (20)	80% (40)
Internet Ads	94.7%	93.5%	94.2% (15)	95.75% (70)	95.6% (10)
Dexter	92.6%	92.6%	92.6% (1240)	**94**% (230)	93.3% (800)
Arcene	83%	83%	83% (6600)	81% (5600)	82% (6000)
I2000	86.3%	72.7%	81.8% (260)	**90.9**% (1060)	86.3% (100)

Data Set	Wrapper Bwd	Wrapper Fwd	CSA Bwd	CSA Fwd
Reuters1	94.4% (35)	92.4% (7)	**98.6**% (51)	96.5% (10)
Reuters2	**95.7**% (53)	91.4% (5)	93.2% (109)	90.1% (14)
Arrhythmia	77.8% (17)	70% (5)	**84.2**% (21)	74.2% (28)
Internet Ads	95% (62)	-	**96.1**% (158)	95.6% (8)
Dexter	92.6% (653)	80% (10)	93.3% (717)	92.6% (100)
Arcene	82% (6800)	58% (7)	**86**% (7200)	81% (600)
I2000	86.3% (1600)	86.3% (550)	**90.9**% (1100)	86.3% (500)

Notes: Upper table: No FS: no feature selection; SVM: linear soft margin SVM without feature selection; Corr: feature selection using Pearson correlation; MI: feature selection using mutual information; RF: feature selection using Random Forests. Bottom table: Wrapper Bwd and Wrapper Fwd: wrapper with backward and forward selection, respectively; CSA Bwd and CSA Fwd: CSA with backward elimination and forward selection, respectively, with parameters from Table 2. Accuracy levels are calculated by counting the number of misclassified parentheses. The number of features selected is given in parentheses. Notice that the accuracies obtained by our algorithms on the Dexter and Arcene data sets are inferior to those of the winners of NIPS 2003 feature selection competition.

3.3.1 Reuters1 Data Set. Feature selection using CSA with backward elimination did best, yielding an accuracy level of 98.6% with 51 features. Koller and Sahami (1996), for example, report that the Markov Blanket algorithm yields approximately 600 selected features with accuracy levels of 95% to 96% on this data set.

3.3.2 Reuters2 Data Set. Wrapper with backward elimination did best, yielding accuracy level of 95% with 53 features. For comparison, Koller and Sahami (1996) report that the Markov Blanket algorithm yields approximately 600 selected features with accuracy levels of 89% to 93% on this data set.[3]

[3] The data sets used in Koller and Sahami (1996) are not identical to the data sets we used. We were unable to obtain the same data sets and had to reconstruct them from the original Reuters-21578 collection.

3.3.3 Arrhythmia Data Set. This data set is considered a difficult one. CSA with backward elimination did best, yielding an accuracy level of 84% with 21 features. Forward selection with higher depth value ($d = 20$) did better than with $d = 1$, implying that one should consider many features concomitantly to perform good feature selection for this data set. For comparison, the grafting algorithm (Perkins et al., 2003) yields an accuracy level of approximately 75% on this data set.

3.3.4 Internet Ads Data Set. All the algorithms did approximately the same, leading to accuracy levels between 94% and 96%, with CSA slightly outperforming the others. Interestingly enough, with $d = 1$, the algorithm did not select any feature. In the first phase, the 1NN algorithm had neighbors from both classes with the same distance for each feature checked, leading to arbitrary selection of one of the classes, and the classifier's performance was constant through all the phase, yielding zero contribution values. However, when selecting the higher depth levels, the simple 1NN algorithm was boosted up to outperform classifiers such as SVM.

3.3.5 Dexter Data Set. For the Dexter data set, we used algorithm L (*C4.5* decision trees) only for the process of feature selection and linear *SVM* to perform the actual prediction on the features selected. This was done because *C4.5* did not give satisfying accuracy levels for any of the feature selection algorithms, and it is impractical to use SVM with CSA for large data sets. To overcome the difference between the classifiers performing feature selection and the classifier used for the actual classification, we added an optimization phase for the forward selection algorithm after it stopped. In this phase, a tenfold crossvalidation is performed on the data set in a similar way to the one used to optimize filter methods. The simple mutual information feature selection performed best, followed closely by the CSA in its backward elimination version and by Random Forests. This implies that in Dexter, the contribution of single features significantly outweighs the contribution of feature combinations for the task of classification. The forward selection algorithm did as well as linear SVM without feature selection, but with a significantly lower number of features.

3.3.6 Arcene Data Set. Here, just as in the case of Dexter, we use *C4.5* for the process of feature selection and linear SVM to perform the actual prediction on the features selected. The CSA with backward elimination obtained better performance than the rest of the algorithms.

3.3.7 I2000 Data Set. CSA with backward elimination, together with feature selection using mutual information, yielded the best results. The poor performance of CSA with forward selection can be explained by the poverty

Table 4: Wilcoxon Signed-Rank Test p-Values.

	CFwd.	WBwd.	WFwd.	RF	MI	No FS	SVM	Corr.
CBwd.	0.015	0.078	0.031	0.093	0.015	0.015	0.015	0.015
CFwd.	-	0.672	0.219	0.625	0.218	0.625	0.687	0.156
WBwd.		-	0.016	0.031	0.094	0.016	0.016	0.016
WFwd.			-	0.156	0.046	0.625	0.437	0.937
RF				-	0.562	0.156	0.156	0.156
MI					-	0.078	0.437	0.109
No FS						-	0.812	0.812
SVM							-	0.109

Notes: This table specifies the Wilcoxon signed-rank test p-values related to the results in Table 3. The entry on row i and column j specifies the p-value related to testing whether method i is superior to method j. CBwd stands for CSA with backward elimination and WBwd stands for Wrapper with backward elimination. CFwd and WFwd follow similar naming. p-values were calculated using the exact distribution for $n = 7$ tests, which can easily be calculated by enumeration. It can be seen that the backward elimination CSA is better than the other methods tried with significance level 0.05, except for Random Forests and Wrapper backward elimination, where only a marginal significance is achieved. No other feature selection method was found to be significantly better than the majority of the remaining methods.

of data compared to the number of features. The algorithm selected in the first phases features that explain well the training data by coincidence and avoided selecting features that truly contribute to the task of classification. This phenomenon is explained section 3.5.2.

3.4 Significance of the Results. The McNemmar test (Gillick & Cox, 1989) on the results summarized in Table 3 identified no significant superiority of any feature selection method on any of the data sets. The accuracies are too close to each other compared to the size of the test sets.

However, in five of the seven data sets, CSA with backward elimination achieved the highest accuracy. In the other case, it achieved the second-best accuracy. So although the results are not significant for each data set, the overall picture may suggest otherwise. To test whether the backward elimination version of CSA is indeed superior to the other feature selection algorithms, we performed a one-sided Wilcoxon signed-rank test (Kanji, 1994). This test takes into account the ranking of feature selection methods across all data sets and tests whether the set of rankings significantly deviates from the H_0 distribution that assumes that all methods are equal. Table 4 lists the p-values of these tests. As can be seen, the backward elimination version of CSA has significantly higher performance than most of the other methods tried.

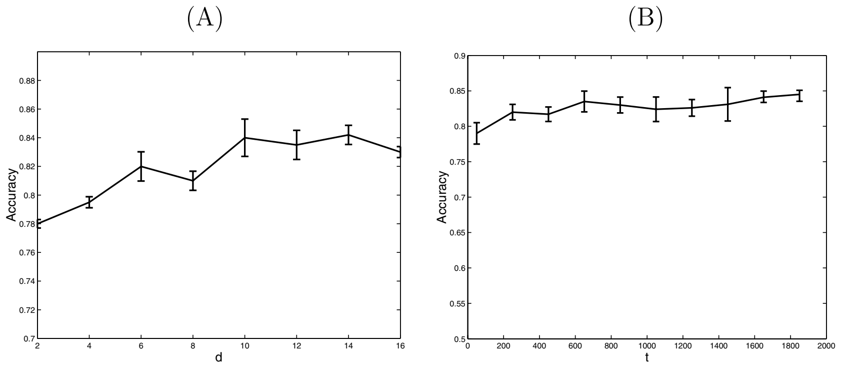

Figure 3: Dependency of CSA performance on its hyperparameters. (A) The accuracy of the backward elimination version of CSA for various values of d, size of subsets. As expected, for small values of d, there is a substantial improvement of performance as d is increased. But beyond a certain point (here, around $d = 10$), the accuracy stays stable around 83%. (B) The accuracy of backward elimination of CSA for various values of t, the number of permutations sampled. The overall picture is compatible with the fact that the estimate of the contribution value becomes more robust as more samples are taken.

3.5 A Closer Inspection of the Results

3.5.1 Behavior of the Algorithm with Different Parameters. In order to examine the effect of different parameter values on the algorithm, we ran the CSA on the Arrhythmia data set with different values of d (size of subsets analyzed) and values of t (number of permutations examined in each phase of eliminating new features). The results were averaged over five experiments for each value of d and t.

Figure 3 describes the result. Figure 3A implies that there are optimal values of d for which the performance achieved is highest. For small values of d, not enough interactions between the different features are considered. As d increases, the performance on the data set increases until it reaches a critical value. For values of d larger than that critical value, the performance stays stable around the critical value's performance.

Figure 3B implies that the algorithm is rather robust to the number of permutations analyzed in each phase. For very small t values, the algorithm's performance is limited. But as t grows to values a little higher, the performance grows as well, until it stays rather stable.

3.5.2 The Distribution of the Contribution Values. The MSA, intent on capturing correctly the contribution of elements to a task, enables us to examine the distribution of the contribution values of the features. Figure 4 depicts a

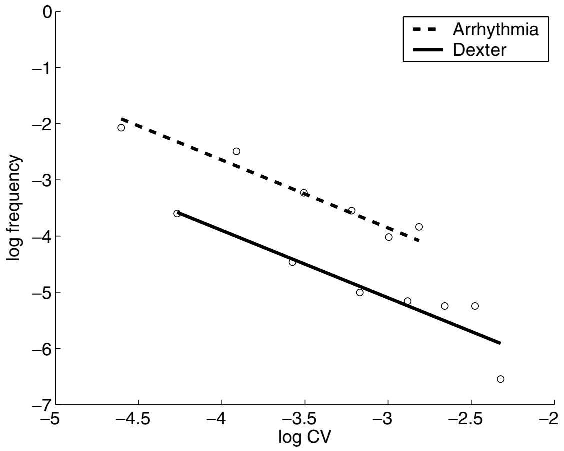

Figure 4: Power law distribution of contribution values. This log-log plot of the distribution of the contribution values (absolute value) in the first phase for Arrhythmia and Dexter, prior to making any feature selection, demonstrates a power law behavior. For both axes, natural logarithms are used. The p-values for the regression were 0.0047 (Arrhythmia) and 0.0032 (Dexter). The corresponding plots for the other data sets show power law characteristics with different slopes and were eliminated for clarity.

log-log plot of the distribution of the contribution values in the first phase for Arrhythmia and Dexter, prior to making any feature selection. This distribution follows a scale-free power law, implying that large contribution values (in absolute value) are very rare, while small ones are quite common, justifying quantitatively the need of feature selection. The other data sets were also observed to possess a similar power law characteristic.

The behavior of the algorithm through the process of feature selection and feature elimination is displayed in Figure 5. After the forward selection algorithm identifies the significant features in the first few phases, there is a sharp decrease in the contribution values of the features selected in the following phases, while with backward elimination, there is a gradual and rather stable increase in the contribution values of the noneliminated features. The peaks in the graph of the contribution values in Figure 5A demonstrate that the contribution values change as the CSA iterates. In this case, the selection of a single feature considerably increased the contribution value of another feature, pointing at intricate dependencies between features.

Figures 4 and 5 also assist in explaining why backward elimination usually outperforms several feature selection methods, including forward selection. Due to the high dimensionality of the data sets, a feature that assists in prediction merely by coincidence may be selected on account of other

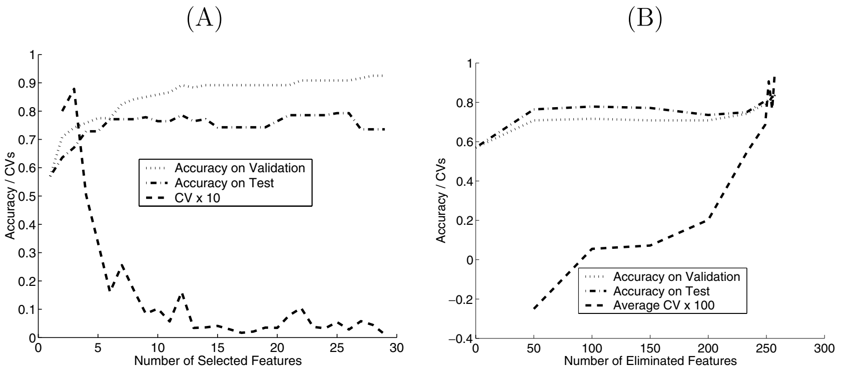

Figure 5: Prediction accuracy and feature contribution during forward selection (A) and backward elimination (B) for the Arrhythmia data set. Both parts of the figure show how the performance of the C4.5 classifier improves on the validation set as the algorithm selects (eliminates) new features, while the contribution values of the selected features decrease (increase). The backward elimination generalizes better on the test set through the algorithm's progress. The behavior for the other data sets is similar.

truly informative features. Forward selection is penalized severely in such case: among the few significant features, some will not be chosen. However, backward elimination always maintains the significant features in the noneliminated set; a feature that truly enhances the classifier's generalization will do so for the validation set as well and will not be eliminated. This leads to a more stable generalization behavior for backward elimination on the test set through the algorithm's progress (see Figure 5).

4 Discussion

CSA evaluates on each phase t feature sets for each coalition size in the range 1 to d, leading to $O(\frac{td}{e}n)$ sets being evaluated. However, in order to obtain reliable estimates of the contribution values, td should scale linearly with n. Therefore, in practice, CSA requires $O(n^2)$ evaluations, like standard forward selection or backward elimination (Wrappers). The filter methods, ranking features by their Pearson correlation (Corr) or by by their mutual information (MI) with the targets, both have linear time complexity. The time complexity of Random Forest (RF) is difficult to assess, since in order to get a reliable estimate of feature importance, one should increase the number of trees; the resulting trees are usually deeper, and with each node, $O(\sqrt{n})$ features are being considered.

The $O(n^2)$ behavior of CSA poses a real challenge when using it with nontrivial problems. To cope with it, fast induction algorithms such as naive Bayes, KNN, or decision trees must be used. Running times can be further reduced by parallelizing, an advantage not shared by wrapper algorithms, which use search methods such as hill climbing; At each phase, the permutations can be computed in parallel and combined on completion to obtain an estimate of contribution values. Furthermore, as the algorithm progresses, the number of candidate features for either selection (forward selection) or elimination (backward elimination) decreases. Consequently, the number of permutations sampled may be reduced, speeding up the algorithm significantly. The restriction in selecting the learning algorithm for CSA does not apply to the prediction once the features are selected. After a set of features is found by the CSA, it may be used by any induction algorithm, as demonstrated in section 3.3 with the Dexter and Arcene data sets.

Several learning algorithms, such as KNN with Euclidean metric, Naive Bayes classifier and fisher's linear discriminant, allow an optimization that dramatically reduces the time spent in the contribution subroutine. Without such optimizations, running times can be considerable: For example, using C4.5 as a base learning algorithm, the total running time for the Arrhythmia data set was on average 41 minutes on a 1.73 Ghz Pentium 4 for backward elimination and 34 minutes for forward selection. The standard deviations of running times were approximately 8 minutes and 6 minutes, respectively. For comparison, the running times for the filter algorithms (mutual information and Pearson correlation) were less than 4 minutes. Forward selection and backward elimination wrapper methods took 28 minutes and 33 minutes, respectively, and the running time of Random Forests on the same data set was 12 minutes.

Since the contribution value is based on extensive sampling of feature sets, the CSA algorithm is capable of identifying intricate dependencies between features and the target. We therefore expect that CSA will be effective for data sets where feature independence is strongly violated, as demonstrated in section 3.1. However, CSA may fail in certain circumstances, for example, when a large coalition should be formed to aid in prediction. In such a case, it may well be that this coalition will not be sampled—and hence, the contribution value of the corresponding features will not be increased. Obviously, forward selection CSA is more prone to this pitfall. Furthermore, when the data are scarce, overfitting could pose a real problem for CSA. The significance of contribution estimates can be rather low, and the resulting noise can play a substantial role in driving the algorithm. Further incorporation of regularization into CSA may help to deal with such a situation.

5 Conclusion

The contribution-selection algorithm presented in this letter views the task of feature selection in the context of coalitional games. It uses a wrapper-like technique combined with a novel ranking method based on the Shapley contribution values of the features to the classification accuracy. The CSA works in an iterative manner, each time selecting new features (or eliminating them) while taking into account the features that were selected (or eliminated) so far.

We verified that the feature sets selected by CSA are significantly different from those selected by other filter methods. It turns out that the first strong features are selected by most methods. But within a few iterations, CSA selects entirely different features from other methods due to the fact that the contribution values of the candidate features are modified along the run of the algorithm, sometimes drastically, according to the features already selected.

The CSA was tested on number of data sets, and the results show that the algorithm can improve the performance of the classifier and successfully compete with an existing array of filter and feature selection methods, especially in cases where the features interact with each other. In such cases, performing feature selection with a permutation size higher than one, namely, not using the common greedy wrapper approach, can enhance the classifier's performance significantly.

The results successfully demonstrate the value of applying game theory concepts to feature selection. While the forward selection version of the algorithm is competitive with other feature selection methods, our experiments show that overall, the backward elimination version is significantly superior to them and produces features sets that can be used to generate a high-performing classifier.

Acknowledgments

We thank David Deutscher and Alon Keinan for their valuable comments and assistance. The MSA toolbox was developed and is maintained by David Deutscher, Alon Keinan, and Alon Kaufman.

References

Alon, U., Barkai, N., Notterman, D. A., Gish, K., Ybarra, S., Mack, D., & Levine, A. J. (1999). Broad patterns of gene expression revealed by clustering of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc. Natl. Acad. Sci., 96*, 6745–6750.

Bengtsson, H. (2003, March). The R.oo package—object-oriented programming with references using standard R code. In K. Hornik, F. Leisch, & A. Zeileis (Eds.), *Proceedings of the 3rd International Workshop on Distributed Statistical Computing (dsc 2003).* Vienna, Austria. Available online at http://www.ci.tuwien.ac.at/Conferences/DSC-2003/.

Billera, L. J., Heath, D., & Raanan, J. (1978). Internal telephone billing rates—a novel application of non-atomic game theory. *Operations Research, 26*, 956–965.

Blake, C., & Merz, C. (1998). *UCI repository of machine learning databases.* Irvine University of California, Irvine, Department of Information and Computer Sciences. Available online at http://www.ics.uci.edu/~mlearn/MLRepository.html.

Blum, A., & Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence, 97*(1–2), 245–271.

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32.

Feigenbaum, J., Papadimitriou, C. H., & Shenker, S. (2001). Sharing the cost of multicast transmissions. *Journal of Computer and System Sciences, 63*(1), 21–41.

Gefeller, O., Land, M., & Eide, G. E. (1998). Averaging attributable fractions in the multifactorial situation: Assumptions and interpretation. *J. Clin. Epidemiol., 51*(5), 437–441.

Gillick, L., & Cox, S. (1989). Some statistical issues in the comparison of speech recognition algorithms. *ICASSP, 1*, 532–534.

Guyon, I. (2003). *Design of experiments for the NIPS 2003 variable selection benchmark*. Tutorial at the NIPS 2003 Workshop on Feature Extraction and Feature selection.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research, 3*, 1157–1182.

Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning, 46*(1–3), 389–422.

Joachims, T. (1999). Making large-scale SVM learning practical. In B. Schölkopf, C. Barges, & A. Smola (Eds.), *Advances in—support vector learning*. Cambridge, MA: MIT Press.

Kanji, G. (1994). *100 statistical tests*. Thousand Oaks, CA: Sage.

Keinan, A., Sandbank, B., Hilgetag, C., Meilijson, I., & Ruppin, E. (2004). Fair attribution of functional contribution in artificial and biological networks. *Neural Computation, 16*(9), 1887–1915.

Keinan, A., Sandbank, B., Hilgetag, C., Meilijson, I., & Ruppin, E. (2006). Axiomatic scalable neurocontroller analysis via the shapley value. *Artificial Life, 12*, 333–352.

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence, 97*, 273–324.

Koller, D., & Sahami, M. (1996). Toward optimal feature selection. In *Proceedings of the 13th International Conference on Machine Learning (ML)* (pp. 284–292). San Francisco: Morgan Kaufmann.

Kushmerick, N. (1999). Learning to remove Internet advertisements. In *Proceedings of the 3rd International Conference on Autonomous Agents*. New York: Association for Computing Memory.

Perkins, S., Lacker, K., & Theiler, J. (2003). Grafting: Fast, incremental feature selection by gradient descent in function space. *Journal of Machine Learning Research, 3*, 1333–1356.

Reuters. (1997). *Reuters collection. Distribution for research purposes by David Lewis*. Available online at http://www.daviddlewis.com/resources/testcollections/reuters21578/.

Rivals, I., & Personnaz, L. (2003). MLPS (mono-layer polynomials and multilayer perceptrons) for nonlinear modeling. *Journal of Machine Learning Research, 3*, 1383–1398.

Roth, A. E. (1979). *Axiomatic models of bargaining*. Berlin: Springer-Verlag.

Shapley, L. S. (1953). A value for n-person games. In H. W. Kuhn, & A. W. Tucker (Eds.), *Contributions to the theory of games* (Vol. 2, pp. 307–317). Princeton, NJ: Princeton University Press.

Shapley, L. S., & Shubik, M. (1954). A method for evaluating the distribution of power in a committee system. *American Political Science Review, 48*(3), 787–792.

Shubik, M. (1962). Incentives, decentralized control, the assignment of joint costs and internal pricing. *Management Science, 8*, 325–343.

Shubik, M. (1985). *Game theory in the social sciences*. Cambridge, MA: MIT Press.

Stoppiglia, H., Dreyfus, G., Dubois, R., & Oussar, Y. (2003). Ranking a random feature for variable and feature selection. *Journal of Machine Learning Research, 3*, 1399–1414.

Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T., & Vapnik, V. (2000). Feature selection for SVMs. In T. K. Leen, T. G. Dietterich, & K.-R. Müller (Eds.), *Advances in neural information processing systems, 13* (pp. 668–674). Cambridge, MA: MIT Press.

Received August 26, 2005; accepted October 14, 2006.