Anytime Algorithm for Feature Selection

Mark Last[1], Abraham Kandel[1], Oded Maimon[2], Eugene Eberbach[3]

[1]Department of Computer Science and Engineering, University of South Florida, 4202 E. Fowler Avenue, ENB 118, Tampa, FL 33620 USA
{mlast,kandel}@csee.usf.edu
[2]Department of Industrial Engineering, Tel-Aviv University, Tel-Aviv 69978, Israel
maimon@eng.tau.ac.il
[3]Jodrey School of Computer Science, Acadia University, Wolfville, NS B0P 1X0, Canada
eugene.eberbach@acadiau.ca

Abstract. Feature selection is used to improve performance of learning algorithms by finding a minimal subset of relevant features. Since the process of feature selection is computationally intensive, a trade-off between the quality of the selected subset and the computation time is required. In this paper, we are presenting a novel, anytime algorithm for feature selection, which gradually improves the quality of results by increasing the computation time. The algorithm is interruptible, i.e., it can be stopped at any time and provide a partial subset of selected features. The quality of results is monitored by a new measure: fuzzy information gain. The algorithm performance is evaluated on several benchmark datasets.

Keywords: feature selection, anytime algorithms, information-theoretic network, fuzzy information gain

1 Introduction

Large number of potential features constitutes a seriously obstacle to efficiency of most learning algorithms. Such popular methods as k-nearest neighbors, C4.5, and backpropagation do not scale well in the presence of many features. Moreover, some algorithms may be confused by irrelevant or noisy attributes and construct poor classifiers. A successful choice of features provided to a classifier can increase its accuracy, save the computation time, and simplify its results.

In practical applications, like data mining, there is no better solution than using the knowledge of a domain expert, who can identify manually all relevant predictors of a given variable. However, in many learning problems, such an expert is not available, and we have to use *automated methods* of feature selection that choose an optimal subset of features according to a given criterion. A detailed overview of feature selection methods is presented by Liu and Motoda (1998).

Since classification accuracy is an important objective of learning algorithms, the most straightforward method (called the *wrapper model*) is to evaluate each subset of features by running a classifier and measuring its validation accuracy. Obviously, this approach requires a considerable computation effort. Another approach (the *filter model*) uses indirect performance measures (like information, distance, consistency, etc.), but it still requires training and validation of a learning algorithm on the final subset of selected features.

Whether the wrapper model or the filter model is applied to a set of features, the user may have to stop the execution of the algorithm, because there is no more time left for continuing the computation. Moreover, the time constraints may be unknown in advance and they can vary from seconds in real-time learning systems to hours or days in large-scale knowledge discovery projects. In both cases, we may be interested to find a good, but not necessarily the optimal, set of features as quickly as possible. However, as appears from (Liu and Motoda, 1998), the existing methods of feature selection do not consider the trade-off between time and performance.

Anytime algorithms (e.g., Dean and Boddy, 1988, Horvitz, 1987, Russell and Wefald, 1991, Zilberstein, 1996) offer such a trade-off between the solution quality and the computational requirements of the search process. The approach is known under a variety of names, including flexible computation, resource bounded computation, just-in time computing, imprecise computation, design-to-time scheduling, or decision-theoretic metareasoning. All these methods attempt to find the best answer possible given operational constraints. A formal model for anytime algorithms is provided by \$-calculus (Eberbach, 2000), which is a higher-order polyadic process algebra with a utility (cost) allowing to capture bounded optimization and metareasoning typical for distributed interactive AI systems.

In this paper, we present an information-theoretic algorithm for feature selection, which is shown theoretically and empirically to have the basic properties of anytime algorithms. We define a criterion for measuring the quality of the algorithm results and study the algorithm performance profile on several benchmark datasets. We also evaluate the performance of the algorithm, when it is run to its completion, by comparing the classification accuracy, using the selected features, to the accuracy obtained with the full set of features.

In section 2, we present a brief overview of existing feature selection methods. The information-theoretic connectionist method of feature selection is described by us in section 3. We also analyze the theoretical properties of the method and compare them to the desired properties of anytime algorithms. Section 4 reports initial experiments that study the performance of the information-theoretic method and suggests possible enhancements using $-calculus. Finally, in section 5 we summarize the benefits and the limitations of our approach and discuss some directions for future research in the field of resource-bounded feature selection.

2 Overview of Feature Selection Methods

Most feature selection techniques are limited to certain types of data. Thus, Littlestone (1988) has studied the problem of learning Boolean functions in the presence of irrelevant attributes. Littlestone's algorithm is limited to datasets with binary classes. The primary advantage of this algorithm is that the number of errors grows only logarithmically with the number of irrelevant attributes.

A number of linear dimension reducers have been developed over years. The linear methods of dimensionality reduction include *projection pursuit* (Friedman and Tukey, 1974), *factor analysis* (see Kim and Mueller, 1978), and *principal components analysis* (see Dunteman, 1989). These methods do not deal directly with eliminating irrelevant and redundant variables, but are rather concerned about transforming the observed variables into a small number of "projections", or "dimensions". The underlying assumptions are that the variables are numeric and the dimensions can be expressed as linear combinations of the observed variables (and vice versa). The methods assume each discovered dimension to represent an unobserved factor and thus provide a new way of understanding the data (similarly to the curve equation in the regression models). However, the linear methods are not able to reduce the number of original features as long as all the variables have non-zero weights in the linear combination.

John *et al.* (1994) distinguishes between two models of selecting a "good" set of features under some objective function. The *feature filter* model assumes filtering the features *before* applying an induction algorithm, while the *wrapper* model uses the induction algorithm itself to evaluate the features. The possible search strategies in the space of feature subsets include *backward elimination* and *forward selection*. The performance criterion of the wrapper model in (John *et al.*, 1994) is the prediction accuracy of the induction algorithm, estimated by *n*-fold cross validation.

A new book on feature selection by Liu and Motoda (1998) suggests a unified model of the feature selection process. Their model includes four parts: feature generation, feature evaluation, stopping criteria, and testing. In addition to the "classic" evaluation measures (accuracy, information, distance, and dependence) that can be used for removing irrelevant features, they mention important *consistency* measures (e.g., *inconsistency rate)*, required to find a *minimum* set of relevant features. By decreasing the inconsistency rate of data, both irrelevant and redundant features are removed. As indicated by Liu and Motoda (1998), consistency measures are only suitable for selecting discrete features.

Caruana and Freitag (1994) present an enhanced greedy algorithm, based on the wrapper model. Again, the metric used is the generalization performance of the learning algorithm (its accuracy over the validation data set), which significantly increases the computation time of the entire process.

Almuallim and Dietterich (1992) make use of the information theory (see Cover, 1991) for selecting relevant features. Their Mutual-Information-Greedy (MIG) Algorithm defined for Boolean noise-free features, selects a feature if it leads to the minimum conditional entropy of the classification attribute. Since the algorithm assumes noiseless data, no significance testing is required (any non-zero entropy is significant). The assumption of noise-free data leaves the MIG algorithm at quite a distance from most practical problems of feature selection.

A statistical approach to feature selection from numeric / ordinal attributes is presented by Liu and Setiono (1997). The adjacent intervals are merged until the chi-square statistic exceeds a pre-determined threshold value. The attributes discretized into a single interval can be removed as irrelevant or redundant. The algorithm is limited to first-order attribute-class correlations.

To sum-up this section, the backward elimination strategy, used by certain methods, is very inefficient for large-scale datasets, which may have hundreds and thousands of original attributes. Most forward selection wrapper methods satisfy the basic requirements of anytime algorithms, but they are highly expensive in terms of the computational effort. The existing filter algorithms are computationally cheaper, but they are evaluating features in a random order, making them more similar to *contract* algorithms, where intermediate results are hardly useful. In the next section, we are describing the information-theoretic method of feature selection, initially introduced by us in (Maimon, Kandel, and Last, 1999) and (Last and Maimon, 1999). As shown in this paper for the first time, the method is much faster than the wrapper techniques and it can be implemented as an anytime algorithm, when the computation time is limited.

3 Information-Theoretic Method of Feature Selection

Our method selects features by constructing an *information-theoretic connectionist network*, which represents interactions between the predicting (*input*) attributes and the classification (*target*) attributes. The minimum set of input attributes is chosen by the algorithm from a set of *candidate input* attributes. The network construction procedure is described in sub-section 3.1. The theoretical properties of the algorithm in the context of anytime computation are evaluated in sub-section 3.2.

3.1 Network Construction Algorithm

An information-theoretic network is constructed for each target attribute separately. It consists of the root node, a changeable number of hidden layers (one layer for each input attribute), and a target layer. Each hidden (target) layer consists of nodes representing different values of an input (target) attribute. The network differs from the structure of a standard decision tree (see Quinlan, 1986 and 1993) in two aspects: it is restricted to the same input attribute at all the nodes of each hidden layer and it has interconnections between the terminal (unsplitted) nodes and the final nodes, representing the values of the target attribute

Without loss of generality, we present here a search procedure for constructing a multi-layer network of a *single* target attribute A_i. In a general case, the network should be re-built (starting with Step 4) for every target attribute defined in a database.

Step1 - Given a relation schema and available domain knowledge, partition the schema into a subset of *candidate input* and a subset of *target* attributes.

Step 2 - Enter a minimum significance level α for splitting a network node (default: $\alpha = 0.001$).

Step 3 - Read records of a relation. Records with non-valid or missing target values are ignored by the algorithm. Missing values of candidate input attributes are encoded in a pre-determined form (e.g., by assigning them a special code).

Step 4 - Initialize the structure of the information-theoretic network to a single root node associated with all records and a target layer for values of the target attribute. Initialize to zero $MI\ (A_i\ ;\ I_i)$, the estimated mutual information (Cover, 1991) between the target attribute A_i and the set of input attributes I_i.

Step 5 - If the maximum number of layers (equal to the number of candidate input attributes) is exceeded, **stop** and return the list of selected (input) attributes. Otherwise, go to the next step.

Step 6- Repeat for every candidate input attribute $A_{i'}$, *which is not in the network*:

Step 6.1 – Initialize to zero the degrees of freedom and the estimated conditional mutual information ($MI\ (A_{i'}\ ;\ A_i\ /\ I_i)$) of the candidate input attribute and the target attribute, given the final layer of hidden nodes.

Step 6.2 – If $A_{i'}$ is a continuous attribute, then find the best partitioning of the attribute. The details are provided in (Last and Maimon, 1999).

Step 6.3 – Else (if the attribute $A_{i'}$ is discrete), Do

Step 6.3.1 - Repeat for every node z of the final hidden layer:

Step 6.3.1.1 - Calculate the estimated conditional mutual information of the candidate input attribute *i'* and the target attribute *i*, given the node *z* ($MI\ (A_{i'};\ A_i\ /\ z)$).

Step 6.3.1.2 - Calculate the statistical significance of the estimated conditional mutual information, by using the likelihood-ratio statistic (based on Rao and Toutenburg, 1995).

Step 6.3.1.3 - If the likelihood-ratio statistic is significant, mark the node as "splitted" and increment the estimated conditional mutual information of the candidate input attribute and the target attribute, given the final hidden layer of nodes ($MI\ (A_{i'};\ A_i\ /\ I_i)$); else mark the node as "terminal".

Step 6.3.1.4 - Go to next node.

Step 6.3.2 - Go to next candidate input attribute.

Step 6.3.3 - EndDo

Step 7 - Find a candidate input attribute maximizing the estimated conditional mutual information ("the best candidate attribute").

Step 8 - If the maximum estimated conditional mutual information is zero, **stop** and return the list of input attributes. Otherwise, go to the next step.

Step 9 - Add a new hidden layer to the network: make the best candidate attribute a new input attribute and define new nodes for a Cartesian product of splitted hidden nodes in the previous layer and the values of the best candidate attribute.

Increment the estimated mutual information $MI (A_i ; I_i)$ by the amount of estimated conditional mutual information $MI (A_{i'} ; A_i / I_i)$ at the current layer. According to the chain rule (see Cover, 1991), the overall mutual information between I_i and A_i is equal to the sum of conditional mutual informations at all the hidden layers.

Step 10 - Go to Step 5.

An example of an information-theoretic connectionist network, which has three hidden layers (related to three selected attributes), is shown in Fig. 1. The performance of the algorithm is evaluated in Section 4 below.

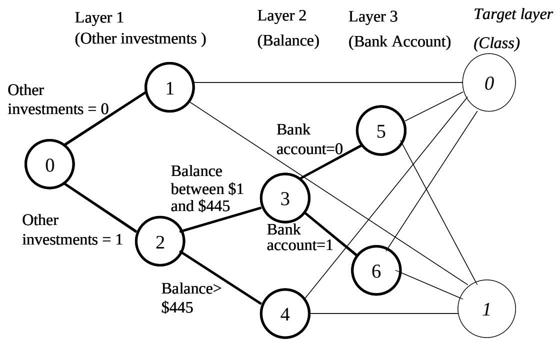

Fig. 1. Information-Theoretic Network: Credit Dataset

3.2 Anytime Properties of the Information-Theoretic Algorithm

According to Zilberstein (1996), the desired properties of anytime algorithms include the following: *measurable solution quality*, which can be easily determined at run time, *monotonicity* (quality is a non-decreasing function of time), *consistency* of the quality w.r.t. computation time and input quality, diminishing returns of the quality over time, interruptibility of the algorithm (from here comes the term *any time*), and preemptability with minimal overhead. Thus, measuring the quality of the intermediate results is the key concept of anytime algorithms. Below, we are discussing several alternatives for measuring the quality of an information-theoretic network.

The most straightforward quality measure is the *classification accuracy* of the model on the training cases. Training accuracy is a non-decreasing function of time, since at each iteration, the algorithm adds a new feature to make the model more accurate. However, this parameter cannot be used as a direct indicator of the solution quality due to the well-known problem of overfitting (see Mitchell, 1997): the model having the highest training accuracy does not necessarily generalize best on the new (validation) cases. This implies that we should use a separate *validation set* for evaluating the quality of the selected features. A similar approach of combining the training and the validation error rates is used by the CART™ method (Breiman et al., 1984). However, the calculation of the validation accuracy has two serious limitations: it reduces the size of the available training set and increases the computation time (which makes the quality less *recognizable*). Moreover, the accuracy measures may be problematic for datasets with skewed class distributions. This leads us to considering measures, which are computationally cheaper and more general than the classification rate.

Since the network construction is aimed at maximizing the mutual information between a set of input attributes and the target attribute (see Step 7 in the algorithm), we can use the mutual information itself as the quality measure of the network. Mutual information is defined by (Cover 1991) as the difference between unconditional and conditional entropies ($MI (A_i; I_i) = H (A_i) - H (A_i / I_i)$). For an empty set of selected features, the mutual information is zero. Adding input features decreases the conditional entropy and increases the mutual information. The optimal value of the mutual information is equal to the unconditional entropy of the target attribute, which is different for every dataset. This means that the mutual information is not a *normalized* quality function (see Zilberstein, 1993). The quality of the estimated mutual information depends on another parameter: significance level, defined in the step 2 of the algorithm. Thus, the *user perception* of the quality of the results may be a complex, non-linear function of both the mutual information and its statistical significance. A general fuzzy-theoretic approach to automating the human perception of data is described in (Last and Kandel, 1999).

To represent the automated perception of the network quality, we will use here a new measure, called *fuzzy information gain*, which is defined as follows:

$$FGAIN = \frac{2}{1+e^{out}}, \quad out = \frac{\beta\alpha H(A_i / I_i)}{MI(A_i ; I_i)} \qquad \textbf{(1)}$$

Where

H (A_i / I_i) - estimated conditional entropy of the target attribute A_i, given the set of input attributes I_i

MI $(A_i ; I_i)$ - estimated mutual information between the target attribute A_i and the set of input attributes I_i

α - significance level, used by the algorithm

β - scaling factor, representing the perceived utility ratio between the significance level and the estimated mutual information. The meaning of different values of β is demonstrated in Fig. 2. The shape of *FGAIN (MI)* varies from a step function for low values of β (about 1) to almost a linear function, when β becomes much higher (about 500). Thus, β can be used to represent the level of user-specific quality requirements.

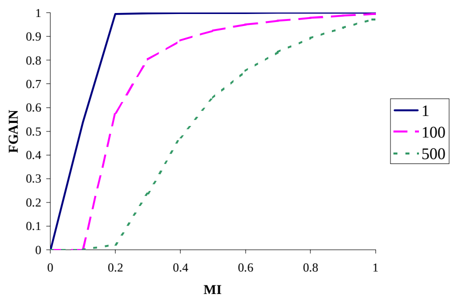

Fig. 2. Fuzzy Information Gain as a function of *MI*, for three different values of β.

Interpretation. FGAIN is defined above as a continuous monotonic function of three parameters: α, *H* (A_i / I_i), and *MI* $(A_i ; I_i)$. It is non-increasing in the significance level α, because lower α means higher confidence and, consequently, higher quality. In the ideal case $\alpha = 0$, which implies that *FGAIN* is equal to one. *FGAIN* is also non-increasing in the conditional entropy *H* (A_i / I_i), because lower conditional entropy represents lower uncertainty of the target attribute, given the values of the input attributes. If the target attribute is known perfectly (*H* $(A_i / I_i) = 0$), *FGAIN* obtains the highest value (one). On the other hand, *FGAIN* is non-decreasing in the mutual information *MI* $(A_i; I_i)$ that represents the decrease in the uncertainty of the target. When *MI* $(A_i ; I_i)$ becomes very close to zero, *FGAIN* becomes exponentially small.

Now we need to verify that our method of feature selection has the desired properties of anytime algorithms, as defined by Zilberstein (1996). The conformity with each property is checked below.

- *Measurable quality*. According to equation (1), the Fuzzy Information Gain can be calculated directly from the values of conditional entropy and mutual information after each iteration of the algorithm.
- *Recognizable quality*. In (Last and Maimon, 1999), we have shown that the mutual information can be calculated incrementally by adding the conditional mutual information of each step to the mutual information at the previous step. This makes the determination of *FGAIN* very fast.
- *Monotonicity*. According to Steps 7 - 9 of the algorithm, a new attribute is added to the set of input attributes only if it causes an increase in the mutual information. This means that the mutual information is a non-decreasing function of run time. Since one can easily verify, that the Fuzzy Information Gain is a monotonic non-decreasing function of *MI*, the monotonicity of the quality is guaranteed. In fact, *FGAIN* is a step function, since no new results can be available between the completions of two succeeding iterations. An example of such quality function is shown by Zilberstein (1993).
- *Consistency*. The theoretical run time of the algorithm has been shown by us in (Last and Maimon, 1999) to be quadratic-logarithmic in the number of records and quadratic polynomial in the number of initial candidate input attributes. In the next section, we are going to analyze experimentally the performance profile of the algorithm on datasets of varying size and quality.
- *Diminishing returns*. This property is very important for algorithm's practical usefulness: it means that after a small part of the running session, the results are expected to be sufficiently close to the results at completion time. We could prove this property mathematically, if we could show that the mutual information is a concave function of the number of input attributes. Though the last proposition is not true in a general case, it is possible to conclude from

Fano's inequality (see Cover, 1991) that the mutual information is *bounded* by a concave function. The actual concavity of *FGAIN* is evaluated empirically in the next section.

- *Interruptibility*. The algorithm can be stopped at any time and provide the current list of selected attributes. Before the completion of the first iteration, the algorithm will provide an empty list resulting in zero quality. Each iteration forms, what is called, a *contract anytime algorithm*, i.e. the corrections of *FGAIN* are available only after termination of an iteration.
- *Preemptability*. Since the algorithm maintains the training data, the list of input attributes, and the structure of the information-theoretic network, it can be easily resumed after an interrupt. If the suspension is expected to be long, all the relevant information may be stored in files on a hard disk.

4 Experimental Results

According to Zilberstein (1996), the performance profile (PP) of an anytime algorithm denotes the expected output quality as a function of the execution time *t*. Since there are many possible factors affecting the execution time, the performance profile, in many cases, has to be determined empirically and not analytically. To study the performance profile of the information-theoretic method for feature selection, we have applied it to several benchmark datasets, available from the UCI Machine Learning Repository (Blake and Merz, 1998). Rather than measuring the absolute execution time of the algorithm on every dataset, we have normalized it with respect to the *completion time*, which is the minimal time, when the expected quality is maximal (Zilberstein, 1993). Obviously, this relative time is almost independent of the hardware platform, used for running the algorithm.

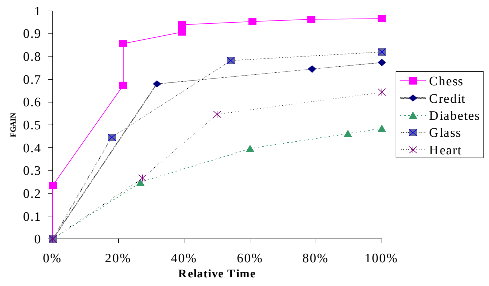

Fig. 3. Performance profile of the information-theoretic algorithm

We have used seven datasets for our analysis (see Table 1), but in two datasets (Breast and Iris), the run time was too short to be detectable by the computer system (Pentium II 400 MHZ). Thus, we are presenting in Fig. 3 performance profiles for five datasets only. Two important observations can be made from this chart. First, we can see that *FGAIN* is a non-decreasing function of execution time. The second observation is about the diminishing returns: except for the Chess dataset, the performance profiles are *concave* functions of time. We have explained the theoretical background of this result in sub-section 3.2 above.

The number of selected features in each dataset and the absolute execution times are shown in Table 1. The size of the datasets varies between 150 and 3,196 cases. The total number of candidate input attributes is up to 36, including nominal and continuous features. On average, less than 30% of the attributes have been selected by the algorithm, when it was run to its termination. The completion time starts with undetectable (less than 0.1 sec.) and goes up to 1.65 sec. for the Diabetes dataset, which has 768 records and 8 continuous attributes. These times are significantly lower than the execution times of a wrapper selector, which may vary between 16sec and several minutes for data sets of similar size (see Liu and Motoda, 1998).

Another question is how useful are the selected features for the classification task? The selected features can be considered useful, if a classifier's accuracy remains at approximately the same level. To verify this assumption, we have partitioned each dataset into training and validation records, keeping the standard 2/3 : 1/3 ratio (Liu and Motoda, 1998). The C4.5 algorithm (Quinlan, 1993) has been trained on each dataset two times: before and after feature selection. The error rate of both models has been measured on the same validation set. The minimum and the maximum error rates have been calculated for a 95% confidence interval. As one can see from Table 2, the error rate of C4.5 after feature selection is not significantly different from its error rate with all the available features. Moreover, it tends to be slightly lower after applying the feature selection algorithm. One exception is the Chess dataset, where the error rate has increased beyond the upper bound of the confidence interval. Due to the feature selection procedure, the stability of the error rate is accompanied, in most datasets, by a considerable reduction in the size of the decision tree model (measured by the number of tree nodes).

The novelty of our approach is that it allows capturing the trade-off between the solution quality and the time saved and/or complexity of classification represented by the number of input attributes. This can be crucial for classification algorithms working with a large number of input attributes, or with real time constraints. Alternative quality measures and costs of meta-reasoning can be studied in the process algebra framework provided by $-calculus (Eberbach, 2000) which formalizes anytime algorithms. For example, in terms of $-calculus expressing the tradeoff between the quality solution and the time, can be thought as a new measure $FGAIN_{tot}=(1-t)\,FGAIN$, where t is a normalized execution time (assuming that the execution time is bounded), or alternatively *out* in *FGAIN* can be modified.

Another benefit of $-calculus would be if we have a complete system consisting of several modules working in parallel, sequentially, or alternatively as we have in multi-agent systems. In other words, $-calculus tries to address such questions like: having two systems performing their own classifications, what will be the final quality combining them into one system. $-calculus should allow to integrate, for instance, data-mining perception and learning modules with action selection and monitoring. The key aspect is the separability of the quality/cost measures, allowing to express the quality of the whole system as the function of its component qualities.

Table 1. Feature selection: summary of results

Dataset	**Data Size**	**Classes**	**Continuous**	**Nominal**	**Total Attributes**	**Selected Attributes**	**Completion Time (sec.)**
Breast	699	2	9	0	9	3	-
Chess	3196	2	0	36	36	9	0.28
Credit	690	2	6	8	14	3	1.04
Diabetes	768	2	8	0	8	4	1.65
Glass	214	6	9	0	9	3	0.61
Heart	297	2	6	7	13	3	0.22
Iris	150	3	4	0	4	1	-
Mean	**859**	**3**	**6**	**7**	**13.3**	**3.6**	**0.76**

Table 2. Error rate and tree size of C4.5 before and after feature selection

	Validation	Before F.S.				After F.S.	
Dataset	Items	Tree Size	Error Rate	Min.	Max.	Tree Size	Error Rate
Breast	204	29	5.4%	2.3%	8.5%	19	4.9%
Chess	1025	45	1.3%	0.6%	2.0%	29	3.0%
Credit	242	26	14.5%	10.0%	18.9%	3	14.0%
Diabetes	236	63	28.4%	22.6%	34.1%	23	23.3%
Glass	71	39	36.6%	25.4%	47.8%	39	33.8%
Heart	93	33	19.4%	11.3%	27.4%	16	24.7%
Iris	49	9	0.0%	0.0%	9.5%	5	2.0%

5 Summary

In this paper, we have presented a novel algorithm for feature selection, which can be interrupted at any time and provide us with a partial set of selected features. The quality of the algorithm results is evaluated by a new measure, the fuzzy information gain, which represents the user perception of the model quality. The performance profile of the algorithm has been shown to be a non-decreasing and mostly concave function of execution time. The quality of the final output has been confirmed by applying a data mining algorithm (C4.5) to a set of selected features.

Topics for future research include consideration of alternative quality measures, predicting expected quality for a given run time (and vice versa), and integrating anytime feature selection with real-time learning systems.

Acknowledgment

This work was partially supported by the USF Center for Software Testing under grant no. 2108-004-00.

References

1. H. Almuallim and T. G. Dietterich, Efficient Algorithms for Identifying Relevant Features, Proc. 9th Canadian Conf. on AI, pp. 38-45, 1992.
2. C.L. Blake and C.J. Merz , UCI Repository of machine learning databases, http://www.ics.uci.edu/~mlearn/MLRepository.html, 1998.
3. L. Breiman, J.H. Friedman, R.A. Olshen, & P.J. Stone, Classification and Regression Trees, Wadsworth, Belmont, CA, 1984.
4. R. Caruana and D. Freitag, Greedy Attribute Selection, Proc. 11th Conf. on Machine Learning, pp. 28-36, 1994.
5. T. M. Cover, Elements of Information Theory, Wiley, New York, 1991.
6. T. Dean and M. Boddy, An Analysis of Time-Dependent Planning, Proc. AAAI-88, pp.49-54, AAAI, 1988.
7. P. Domingos and M. Pazzani, On the Optimality of the Simple Bayesian Classifier under Zero-One Loss, Machine Learning, vol. 29, pp. 103-130, 1997.
8. G.H. Dunteman, Principal Components Analysis, Sage Publications, Inc., Newbury Park, CA, 1989.
9. E. Eberbach, Expressiveness of $-Calculus: What Matters?, Proc. The Ninth Intern. Symp. on Intelligent Information Systems IIS'2000, Springer-Verlag, Bystra, Poland, June 2000.
10. E. Eberbach, Expressing Evolutionary Computation, Genetic Programming, Artificial Life, Autonomous Agents and DNA-Based Computing in $-Calculus – Revised Version, Proc. Congress on Evolutionary Computation CEC'2000, San Diego, CA, July 2000.
11. J.F. Elder IV and D. Pregibon, A Statistical Perspective on Knowledge Discovery in Databases. In Advances in Knowledge Discovery and Data Mining, U. Fayyad, et al., Eds. AAAI/MIT Press, Menlo Park, CA, pp. 83-113, 1996.
12. U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, From Data Mining to Knowledge Discovery: An Overview. In Advances in Knowledge Discovery and Data Mining, U. Fayyad, et al., Eds. AAAI/MIT Press, Menlo Park, CA, pp. 1 -30, 1996.
13. J.H. Friedman and J.W. Tukey, A Projection Pursuit Algorithm for Exploratory Data Analysis, IEEE Transactions on Computers, vol. 23, no. 9, pp. 881-889, 1974.
14. E.J. Horvitz, Reasoning about Beliefs and Actions under Computational Resource Constraints, Proc. of the 1987 Workshop on Uncertainty in AI, Seattle, Washington, 1987.
15. G. H. John, R. Kohavi, and K. Pfleger, Irrelevant Features and the Subset Selection Problem, Proc. Machine Learning: Proc. of the 11th Int'l Conf., San Mateo, CA, pp. 121-129, 1994.
16. J-O. Kim and C.W. Mueller, Factor Analysis: Statistical Methods and Practical Issues, Sage Publications, Inc., Beverly Hills, 1978.
17. G. J. Klir and B. Yuan, Fuzzy Sets and Fuzzy Logic: Theory and Applications, Prentice-Hall Inc., Upper Saddle River, CA, 1995.
18. H.F. Korth and A. Silberschatz, Database System Concepts, McGraw-Hill, Inc., New York, 1991.
19. M. Last and O. Maimon, An Information-Theoretic Approach to Data Mining, Submitted to Publication, 1999.
20. M. Last and A. Kandel, Automated Perceptions in Data Mining, Proc. 1999 IEEE International Fuzzy Systems Conference, Seoul, Korea, pp. 190-197, 1999.
21. N. Littlestone, Learning Quickly When Irrelevant Attributes Abound: A New Linear-threshold Algorithm, Machine Learning, no. 2, pp. 285-318, 1988.
22. H. Liu and R. Sutiono, Feature Selection via Discretization, IEEE Transactions on Knowledge and Data Engineering, vol. 9, no. 4, pp. 642-645, 1997.
23. H. Liu and H. Motoda, Feature Selection for Knowledge Discovery and Data Mining, Kluwer, Boston, 1998.
24. O. Maimon, A. Kandel, and M. Last, Information-Theoretic Fuzzy Approach to Knowledge Discovery in Databases. In Advances in Soft Computing - Engineering Design and Manufacturing, R. Roy, T. Furuhashi and P.K. Chawdhry, Eds. Springer-Verlag, London, 1999.
25. T.M. Mitchell, Machine Learning, McGraw-Hill, New York, 1997.
26. S. Russell and E. Wefald, Do the Right Thing: Studies in Limited Rationality, The MIT Press, 1991.
27. J.R. Quinlan, Induction of Decision Trees, Machine Learning, vol. 1, no. 1, pp. 81-106, 1986.
28. J. R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, CA, 1993.
29. C.R. Rao and H. Toutenburg, Linear Models: Least Squares and Alternatives, Springer-Verlag, Berlin, 1995.
30. S. Zilberstein, Operational Rationality through Compilation of Anytime Algorithms, Ph.D. Dissertation, University of California at Berkeley , 1993.
31. S. Zilberstein, Using Anytime Algorithms in Intelligent Systems, AI Magazine, vol. 17, no. 3, pp. 73-83, 1996.