Journal of the Society of Korea Industrial and Systems Engineering
Vol, 29, No. 2, pp.75－82, June 2006.

확장 가능한 요소선택방법을 위한 분석적 접근†

양재경 · 이태한

전북대학교 산업정보시스템공학과, 공업기술연구센터

Analytical Approach for Scalable Feature Selection

Jae-Kyung Yang · Tae-Han Lee

Department of Industrial and Information Systems Engineering,
Research Center of Industrial Technology, Chonbuk National University

본 연구에서 조합 최적화(Combinatorial Optimization) 이론에 바탕을 두고 있는 네스티드 분할(Nested Partition, 이하 NP) 방법을 이용한 최적화 기반 요소선택 방법(Feature Selection)을 제안한다. 이 새로운 방법은 좋은 요소 부분집합을 찾는 휴리스틱 탐색 절차를 채용하고 있으며 데이터의 인스턴스(Instances 또는 Records)의 무작위 추출(Random Sampling)을 이용하여 이 요소선택 방법의 처리시간 관점에서의 성능을 향상 시키고자 한다. 이 새로운 접근 방법은 처리시간 향상을 위해 2단계 샘플링 방법을 채용하여 근접 최적해로의 수렴(Convergence)을 보장하는 샘플 사이즈를 결정한다. 이는 알고리듬이 유한한 시간내에 끝이날 때 최종 요소 부분집합 해의 질(Qualtiy)에 관한 정확한 설명을 할 수 있는 이론적인 배경을 제시한다. 중요 결과를 예시하기 위해서 다양한 형태의 다섯 개의 데이터 셋을 이용하였으며 다섯 번의 반복 실험을 통한 실험 결과가 제시되며, 이 새로운 접근 방법이 기존의 단순 네스티드 분할 방법 기반의 요소선택 방법보다 처리시간 관점에서 더욱 효율적임을 보여준다.

Keywords : Data Mining, Feature Selection, Nested Partition, Optimization, Sampling

1. Introduction

Feature selection can be used to improve the simplicity of a data mining system, as well as maintain acceptable accuracy for the learning algorithm to be used. It is also known that feature selection can improve the scalability of a data mining system as the learning process is usually faster with fewer features. In this paper, we are interested in improving the scalability of the feature selection process itself with respect to large number of instances. Our approach is based on an optimization-based feature selection method that uses the nested partitions (NP) metaheuristic [8], which has been shown to perform well when compared with other feature selection methods [5]. The NP method uses random search to explore the entire space of possible feature subsets, and is thus similar to methods such as genetic algorithms [12] and evolutionary search [7]. However, the search strategies themselves are quite different.

We show that using random sampling of instances can considerably reduce the computational time of the NP based feature selection algorithm. Since the random sampling may add noise to the evaluation of each feature subset, we propose a two-stage variant of the algorithm that can be used to control this noise and is guaranteed to converge to a near-optimal feature subset in finite time. Using sampling of instances to improve scalability has been investigated intensely in the literature, and perhaps the most important, but yet difficult issue, is determining the appropriate sample size to maintain an ac-

† This work was supported by the Research Center of Industrial Technology at CBNU.

ceptable accuracy. Some of the related research includes determining the sufficient sample sizes for finding association rules [9], progressive sampling methods [6], finding best sample sizes using a tuple relational calculus [3], and investigating the effect of class distribution on scalable learning [10].

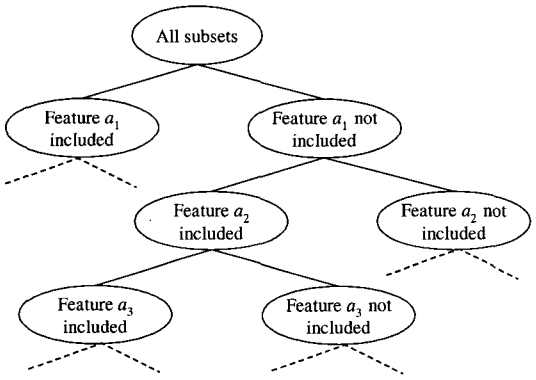
2. NP-Based Feature

2.1 Selection

The following notation will be used:

T : Training data (instances).
m : Number of instances ($m = |T|$).
$A^{(ALL)}$: Set of all features.
n : Number of features ($n = |A^{(ALL)}|$).
a : A specific feature ($a \in A^{(ALL)}$).
f : Performance measure.
f^* : Optimal Performance.

The feature selection problem involves identifying a subset A of the set $A^{(ALL)}$ of all n features that performs well given the training set T of m instances. The performance is measured according to some measure f, and the objective is to find the optimal subset $A^* \subseteq A^{(ALL)}$, where $f^* = f(A^*) = \lim_{A \subseteq A^{(ALL)}} f(A)$.



<Figure 1> Partitioning tree

The nested partitions (NP) method is a general optimization methodology that can be applied to any combinatorial optimization problems. The main idea of the method is to use iterative partitioning of the feasible region, that then creates a partitioning tree as shown in <Figure 1>.

It uses *partitioning* to divide the space of all possible feature subsets into regions that can be analyzed individually and then aggregates the results from each region to determine how to continue the search, that is, how to concentrate the computational effort. In other words, the NP method adaptively takes *random samples of feature subsets* from the entire space of possible feature subsets and concentrates the sampling effort by systematic partitioning of this space. A key component in formulating the feature selection problem is selecting a performance measure. Depending on how this is done, feature selection methods may be divided into two categories: wrappers and filters. Wrappers use the accuracy of the resulting classification. Thus, to evaluate a subset of features, a predictive model is induced based on these features. This is an expensive evaluation and only applies for supervised learning. Filters, on the other hand, select features before any other learning algorithm is applied. A different performance measure must therefore be specified. When choosing a wrapper or filter, the general consideration is that wrappers will give better performance when used with a supervised learning method, whereas filters are usually much faster. The NP optimization method can be implemented as either a wrapper or filter [5]. Here, we focus on a filter employing the following correlation based measure [2] :

$$f_{correlation}(A) = \frac{k\bar{\rho}_{ca}}{\sqrt{k + k(k-1)\bar{\rho}_{aa}}}, \quad \cdots\cdots (1)$$

where k is the number of features in the set A, $\bar{\rho}_{ca}$ is the average correlation between the features in this set and the classification feature, and $\bar{\rho}_{aa}$ is the average correlation between features in the set A.

The NP method searches through the space of feature subsets by evaluating the entire subsets. On the other hand, it also incorporates methods that evaluate individual features into the partitioning to impose a structure that speeds the search. When it is done in such a way that good solution as clustered together in the same subsets, then those subsets are selected by the algorithm with relatively little effort. We now discuss an intelligent partitioning strategy when solving feature selection problems. Thus given a current set A(k) of potential feature subsets, partition the set into two disjoint subsets

$$\boldsymbol{A}_1(k) = \{A \in \boldsymbol{A}(k) : a \in A\} \quad \cdots\cdots (2)$$
$$\boldsymbol{A}_2(k) = \{A \in \boldsymbol{A}(k) : a \notin A\} \quad \cdots\cdots (3)$$

The surrounding region is simply $A_3(k) = A \backslash A(k)$. Each

of these three regions is then sampled as discussed above and based on these samples the next most promising region is selected. In theory, the features can be selected in an arbitrary order, but an intelligent partitioning where features are ordered according to their information gain performs significantly better, and this partitioning is used in all of the numerical experiments below.

This partitioning creates a tree of subsets that we refer to as the partitioning tree. The distance of the current promising region from the top of the tree, which corresponds to the minimum number of iterations it takes to get to this region, we refer to as the depth of the region. Once a maximum depth region is reached, that is a region that will not be partitioned further, the algorithm terminates. In the context of feature selection problem, this maximum depth will be equal to the number of features that are considered for either inclusion or exclusion from the selected set.

The key to the convergence of the NP method is the probability by which a region is selected correctly in each iteration. A sufficient condition for asymptotic convergence is that this probability of correct selection is bigger than one half, and to guarantee that a minimum probability is obtained, Olafsson [4] proposed using a two-stage sampling procedure that determines how much random sampling effort $N(\Psi,\delta)$ is needed from each region to guarantee correct selection with probability Ψ within an indifference zone $\delta > 0$. If this sampling effort is used, the probability of having found sufficiently good solution the first time maximum depth is reached, that is, when the search space has been reduced to a single feature subset, is bounded as follows:

$$\Pr\{|f(\mathbf{A}(k)) - f^*| \le \delta\} \ge \Psi, \quad (4)$$

where

$$\Psi = \frac{\Psi^n}{(1-\Psi)^n + \Psi^n}, \quad (5)$$

Here Ψ is the user selected minimum probability by which a correct selection is made in each iteration, and n is as before the total number of features.

We call the NP method applied to feature selection using the filter evaluation the NP-Filter, and if it also uses the two-stage sampling approach the Two-Stage NP-Filter (TSNP-Filter). A pseudo-code for the TSNP-Filter is shown in Appendix A, and used the following notation. We let N_j denote the number of sample sets in $A_j(k)$, j = 1, 2, 3., j-th subregion in the k-th iteration and $X_{ij} = f(A_i^j)$, where A_i^j and $f(\cdot)$ are defined as the sample performance of i-th set in the j-th region. The two-stage ranking-and-selection procedure takes n_0 samples in the first stage, and then determines the total number N_j samples required from the j-th region using based on the sample variance of the performance estimates.

3. Instance Sampling in the NP-Filter

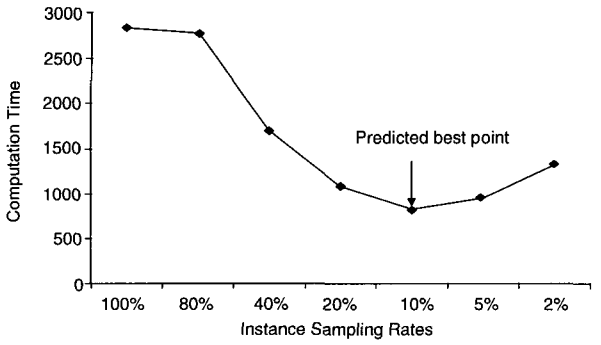
In this section, we consider improving the scalability of the feature selection method in terms of its ability to handle increasing number of instances. The NP method was originally conceived for simulation-based optimization and is therefore naturally consistent with using performance estimates that are noisy due to sampling. Indeed, in the NP-Filter, a new set $A(k)$ of instances is sampled in each iteration in such a way that this set is independent of the previous sets: $A(0)$, $A(1)$, ⋯, $A(k-1)$. Thus, if the new instances indicate an erroneous decision has been made,the backtracking feature of the NP method enables the algorithm to make corrections, thus correcting the potential bias. The question still remains as of how large of a portion of the database is needed by the NP method. In particular, as the proportion is decreased and more backtracking is required. Then at some point the computational inefficiencies of backtracking will outweigh the savings obtained by using fewer instances. To evaluate these questions empirically, we apply the NP-Filter. We used four small data sets from the UCI repository of machine learning databases[1]. The characteristics of these data sets are shown in <Table 1>. As the NP-Filter is randomized algorithms, we run five replications for each experiment and report the average.

<Table 1> Characteristics of the tested data

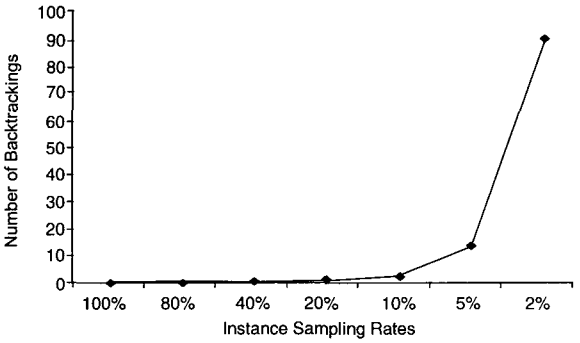
Data Set	Instances	Features
lymph	148	18
vote	435	16
audiology	226	69
cancer	286	9
kr-vs-kp	3196	36

<Figure 2> illustrates the computation time needed by the NP-Filter for different sampling rates used. We note from the graph of that figure that at first the computation time decreases, but if the sampling rate becomes less than approximately 10% of the instances, the computational time actually

increases.



<Figure 2> Computational time for instance sampling rates (data set 'vote')



<Figure 3> Number of backtrackings on instance sampling rates (data set 'vote').

The intuitive explanation of this is shown in the right hand graph. It is clear that number of backtrackings abruptly increases when less than 10% of the instances are used. This means that even though each iteration may take less computation time, the number of iterations until maximum depth is reached increases dramatically, hence increasing the overall computation time. There is therefore some optimal sample rate R^*, where the NP-Filter would perform best.

To find this optimal rate, we must consider the cause of backtracking. The NP-Filter backtracks when it discovers that the surrounding region is actually more promising than the current most promising region. It thus corrects mistakes made due to noisy performance estimates by backtracking when the error is discovered, so we would expect to see more backtracking when fewer instances are used. In particular, when too few instances are used, the noise is excessive and backtracking must hence increase dramatically to compensate for the noise.

In order to get a better feel for how the noise in the performance increases as a function of decreasing sample rate, we consider the data sets in <Table 1>. To calculate the true amount of noise, we must calculate the sample variance given all the feature subsets in a particular region, for all possible levels of the tree. Since the total number of non-empty feature subsets is 2^n-1, where n is the number of features, this quickly becomes infeasible, so instead of working with the datasets directly, we work with subsets of the dataset, each containing 7 randomly selected features. Even for such small datasets, 127 feature subsets must be evaluated for each experiment. The results are shown in <Table 2>.

<Table 2> Performance variances for sampling rates of instances

Sample Rates	100%	80%	60%	40%	20%	10%	5%	2%
vote1	0.0	1.4	4.0	5.0	8.1	17.3	27.5	N/A
vote2	0.0	6.6	9.1	16.4	28.0	38.3	41.9	N/A
audiology1	0.0	1.5	4.2	6.1	16.9	33.4	48.8	94.3
audiology2	0.0	1.2	1.9	5.3	14.9	25.6	58.7	91.1
audiology3	0.0	0.9	2.4	4.9	10.8	28.7	58.2	185.4
cancer1	0.0	0.7	2.1	4.3	19.2	49.1	109.7	N/A
cancer2	0.0	0.5	1.8	4.5	14.7	52.7	104.7	N/A
cancer3	0.0	0.6	1.4	3.0	13.9	53.4	150.8	N/A
kr-vs-kp1	0.0	0.2	0.4	0.9	2.8	5.9	8.9	14.9
kr-vs-kp2	0.0	0.1	0.1	0.2	0.5	1.2	2.7	6.3
kr-vs-kp3	0.0	0.1	0.1	0.2	0.5	1.2	2.7	9.5

From <Table 2> it is clear that the performance variance increases rapidly as the instance sampling rate decreases. Indeed, all of the test datasets exhibit exponential pattern. We also note that although all of the datasets illustrate exponential growth, the rate is different for each dataset. We infer that the sample variance increases exponentially as the sample rate decreases, but that the rate of increase the exponential increase is application dependent and must be estimated from the data.

4. Determining the Sampling Rate for the TSNP-Filter

As for other methods that employ instance sampling to improve performance, finding the optimal sampling rate R^* is the biggest challenge when using the NP-Filter. As seen in Section 3, this sampling rate is related to the backtracking,

which is in turn related to the variance of the performance. Intuitively, decreasing the sampling rate will always decrease the computation time for each iteration, but very small samples may cause a large variance of performances and cause excessive backtracking. This will in turn increase the number of iteration and the overall computation time.

The trade-off is therefore between the computation time within each iteration, and the number of iterations needed until convergence is achieved. In the TSNP-Filter, the expected computation time within an iteration is a function of the performance variance, and the expected number of iterations is a function of the probability of selecting the correct region. Thus, analytical expression can be obtained for both these quantities, and the optimal trade-off achieved.

4.1 Formulation

In this section, we formulate the trade-off between minimizing the computation time within an iteration and minimizing the number of iterations for the TSNP-Filter as an optimization problem and find the optimal instance sampling rate R^* by solving this problem. We use the following notation. The total computation time is $T = T_1 + T_2 + \cdots + T_K$, where T_j is the computation time in the jth iteration and K is the total number of iterations. As before $N = N_j(k)$ denotes the number of sample feature sets at each iteration k, and I is the number of sample instances. The sampling rate is therefore given by $R = I / m$.

We are interested in minimizing the expected time of iteration k that can be affected by the performance variability and sampling rate of instance. We may therefore find a solution using the trade-off between $E[N_k | K]$ and $E[T_k | N_k]$. Since from the previous section it is known that the variability can be represented as the number of instances and $E[N_k | K]$ would increase as the number of instances decreases while $E[T_k | N_k]$ would decrease. Therefore, we formulate the trade-off as the following optimization problem.

$$\text{Min } \lambda \cdot E[N_k | k] + (1-\lambda) \cdot E[T_k | N_k] \quad \cdots\cdots (6)$$

It is clear that as functions of the sampling rate R, $E[N_k | K]$ is decreasing and $E[T_k | N_k]$ is increasing. Furthermore, their scale can be different, so we weight them together using an weight λ that should be determined by the experimenter. We know that $E[N_k | K]$ can be written in terms of the expected performance variance. However, the expected performance variance depends both on the application and the manner in which partitioning is done, so an analytical form cannot be obtain. The same is true for $E[T_k | N_k]$, but our empirical results strongly indicate certain patterns for these expected values, and we state those as assumptions.

Assumption 1. The expected calculation time of each feature sample is directly proportional to the number of instances.
Assumption 2. The relationship between performance variance and instance sampling rate is exponentially distributed. $E[S^2(k)] = c_1 e^{-c_2 \cdot R}$ for $c_1 > 0$, $c_2 > 0$.

As noted before, these is no theoretical justification for Assumptions 1-2, but they are both intuitively appealing and supported by our empirical results. Now, given those assumption, the optimal instances sampling rate is found in the following key theorem.

Theorem 1. Let Assumptions 1-2 hold. By using uniform sampling rateof instances and selecting the initial number of sample feature subsets sufficiently small so that it is smaller than then required number of samples, the optimal instance sampling rate is given by

$$R^* = -\frac{1}{c_2} \cdot \ln\left[\frac{(1-\lambda) \cdot c_0 \cdot \delta^2 \cdot m}{\lambda \cdot h^2 \cdot c_1 \cdot c_2}\right] \quad \cdots\cdots (7)$$

Proof. By Assumption 1, the expected computation time in each iteration, given the number of samples is directly proportional to the number of instances, that is,

$$E[T_k | N_k] = c_0 \cdot E[I] = c_0 \cdot I \quad \cdots\cdots (8)$$

From the fact that if the number of instance samples decreases, the performance variability of feature sample sets increases exponentially as stated in the previous section. The expected number of feature sample sets in each iteration can be stated as follows:

$$E[N_k | K] = \frac{h^2}{\delta^2} E[S^2(k)] \quad \cdots\cdots (9)$$

Based on the (6) and the assumptions, the restated problem is as follows.

$$\lim_{R} \quad \lambda \cdot \frac{h^2}{\delta^2} \cdot c_1 e^{-c_2 \cdot R} + (1-\lambda) \cdot c_0 \cdot m \cdot R$$

$$\text{subject to } 0 < \lambda < 1$$

$$0 < R \leq 1.$$

This problem can be solved by taking the derivative of the objective function and identifying the minimum point that satisfies the constraints. In particular, since

$$\frac{d^2Cost}{dR^2} = \frac{\lambda \cdot h^2}{\delta^2} \cdot c_1 \cdot c_2^2 \cdot e^{-c_2 \cdot R} > 0$$

any solution to

$$0 = \frac{d}{dR}\left[\lambda \cdot \frac{h^2}{\delta^2} \cdot c_1 e^{-c_2 \cdot R} + (1-\lambda) \cdot c_0 \cdot m \cdot R\right]$$
$$= -Rc\lambda \cdot \frac{h^2}{\delta^2} \cdot c_1 e^{-c_2 \cdot R} + (1-\lambda) \cdot c_0 \cdot m$$

is a minimum. It follows that R^* is given by equation (7).

The value of λ can be chosen by an experimenter according to the preference. The indifference zone, δ and selection probability, P^* that determines the value of h should be also determined by the experimenter. If δ is small and P^* is large, the sampling rate would be large and vice versa.

4.2 Numerical Results

The NP method guarantees a correct selection with probability Ψ, within an indifference zone $\delta > 0$. However, since the TSNP-Filter incorporates a heuristic approach, the robustness of this must be evaluated empirically. The constants c_0, c_1, and c_2 are calculated empirically for each of the datasets in <Table 1>, and R^* calculated according to equation (7). To evaluate if the TSNP-Filter solution is within the indifference zone, the true optimum must be known. Since this is computationally intractable except for small datasets, we again use modified data sets that now contain 8 randomly selected features in addition to the class feature. First we find the optimal solution for each dataset using an enumerative approach and then calculate how many solutions of the TSNP -Filter out of 100 replications are within the indifference zone δ. For the other parameters, we set λ = 0.5, δ = 1 and 5 percentage points, and Ψ = 0.75, 0.85 and 0.95. The results are reported in <Table 3>.

From <Table 3>, we note that as expected the sample rate is smaller for δ = 5 than δ = 1, and smaller when Ψ is smaller. Thus, by adjusting the instance sampling rate appropriately, the quality of the solutions found by the TSNP-Filter remains constant. This is supported by the results in <Table 3>, which shows that for each problem there is no significant difference in the accuracy obtained. The percentage of time that this accuracy is within the indifference zone is reported in <Table 4>. We note that for indifference zone of δ = 5, the estimated probability of being within the indifference zone is actually significantly higher that the minimum probability Ψ of correct selection. The intuitive explanation for this is that when the indifference zone is selected this large then it is relatively easy to find feature subsets with accuracy within the indifference zone, and hence this will happen most of the time, even if Ψ = 0.75 is selected. When the indifference zone is smaller, δ = 5, then the estimated probabilities closely follow the prescribed minimum Ψ, but except for the 'vote' dataset, the minimum is not met exactly.

<Table 4> Probabilities that a solution is within the indifference zone

Dataset	δ	Ψ		
		0.75	0.85	0.95
vote	5	0.96	0.98	0.98
	1	0.78	0.88	0.96
audiology	5	0.98	0.98	1.00
	1	0.72	0.83	0.89
cancer	5	0.83	0.88	0.97
	1	0.65	0.72	0.81
kr-vs-kp	5	0.90	0.94	0.95
	1	0.63	0.74	0.87

<Table 3> Accuracy of the TSNP-Filter on the reduced datasets

Dataset	δ	Ψ					
		0.75		0.85		0.95	
		Sampling Rate (%)	Accuracy	Sampling Rate (%)	Accuracy	Sampling Rate (%)	Accuracy
vote	5	16	95.57±0.27	21	95.57±0.26	23	95.58±0.25
	1	32	95.59±0.25	37	95.58±0.26	39	95.61±0.11
audiology	5	27	42.62±0.03	30	43.61±0.02	35	43.63±0.01
	1	44	43.63±0.02	47	43.63±0.02	51	43.64±0.02
cancer	5	24	70.32±0.24	28	70.34±0.11	31	70.35±0.06
	1	40	70.35±0.17	45	70.34±0.09	47	70.36±0.03
kr-vs-kp	5	8	65.55±0.48	13	65.57±0.32	15	65.55±0.41
	1	25	65.51±0.21	29	65.59±0.15	32	65.61±0.07

<Table 5> Comparison of three different scalability methods

Dataset	Approach	Sample Rate	Accuracy	Speed	Backtracks
vote	TSNP-Filter	16	93.2±1.3	786±113	0.2±0.4
	NP-Filter w/sampling	10	92.4±1.0	816±167	1.6±2.2
	NP-Filter	100	93.5±0.4	2820±93	0.0±0.0
audiology	TSNP-Filter	27	70.2±1.6	27722±6804	128.8±24.8
	NP-Filter w/sampling	10	69.2±2.4	35839±14563	371.0±182.0
	NP-Filter	100	69.7±1.9	41105±3255	0.0±0.0
cancer	TSNP-Filter	24	73.5±0.5	418±10	2.4±2.8
	NP-Filter w/sampling	10	72.6±1.2	486±89	7.4±3.4
	NP-Filter	100	73.2±0.6	795±83	0.0±0.0
kr-vs-kp	TSNP-Filter	3	89.0±0.4	5189±492	0.0±0.0
	NP-Filter w/sampling	5	89.0±1.2	7246±809	1.8±3.0
	NP-Filter	100	87.9±5.7	107467±8287	1.8±3.0

The results reported above provide some insights into how the TSNP-Filter works. However, we are primarily interested in how the new two stage sampling approach improves the performance of the original NP-Filter. We thus make a three-fold comparison between the TSNP-Filter, the original NP-Filter, and the NP-Filter with a constant sampling rate found by experiments with sampling rates of R = {100,80,60,40,20,10,5,2} and selecting the best rate. The results are reported in <Table 5>. A more interesting result is that the TSNP-Filter even performs better than the NP-Filter with sampling where the sampling rate is determined experimentally as the best sample rate. The intuitive reason for this is that this approach uses the same sample rate in every iteration without consideration of the size of the regions begin compared in that iteration. This means that it tends to over sampling in certain situations when the decision is relatively easy. The TSNP-Filter, on the other hand, automatically determines the best sampling rate and does this very effectively.

5. Conclusion

We showed that by using random sampling of instances, the speed of the NP-based feature selection method can be improved significantly. The key issue in using sampling is to determine the sample size. For the NP-based approach, using very small portion of a sample rate causes too much noise in the performance evaluation that causes the algorithm to make incorrect moves that must be corrected through backtracking. Hence, the number of iterations increases and the overall computation time does as well. The optimal sampling rate will depend on both the size and structure of the particular dataset, so it cannot be easily determined a priori. However, we proposed a two-stage sampling approach that determines the necessary sampling effort based on the estimated variance. The numerical results reported show that sampling works well in general, and that the two-stage approach finds very good sample rates in an automated manner.

References

[1] Blake, C. L. and Merz, C. J., *UCI Repository of machine learning databases*, http://www.ics.uci.edu/mlearn/MLRepository.html, University of California, Irvine, CA (Date Accessed: October 31, 2003), 1998.

[2] Hall, M. A., "Correlation-based feature selection for discrete and numeric class machine learning," in *Proceedings of the Seventeenth International Conference on Machine Learning*, Stanford University, CA. Morgan Kaufmann, 1998.

[3] Kivinen, J. and Mannila, H., "The power of sampling in knowledge discovery," in *ACM Symposium on Principles of Database Theory*, pp. 77-85, 1994.

[4] Olafsson, S., "Two-stage nested partitions method for stochastic optimization," *Methodology and Computing in Applied Probability*, 6 : 5-27, 2004.

[5] Olafsson, S. and Yang, J., "Intelligent partitioning for feature selection," *INFORMS Journal on Computing*,

in print, 2005.

[6] Provost, F., Jensen, D. and Oates, T., "Efficient progressive sampling," in *Proceedings of the fifth International Conference on Knowledge Discovery and Data Mining*, pp. 23-32, 1999.

[7] Shi, L. and Olafsson, S., "Nested partitions method for global optimization," *Operations Research*, 48 : 390-407, 2000.

[8] Toivonen, H., "Sampling large databases for association rules," in *Proceedings of the 22nd International Conference on Very Large Databases*, pp. 134-145, 1996.

[9] Weiss, G. M. and Provost, F., "The effect of class distribution on classifier learning: an empirical study," Technical Report ML-TR-44, Department of Computer Science, Rutgers University August 2, 2001.

[10] Yang, J. and Honavar, V., "Feature subset selection using a genetic algorithm," In H. Motada and H. Liu (eds), *Feature Selection, Construction, and Subset Selection: A Data Mining Perspective*, Kluwer, New York, 1998.

Appendix. TSNP-Filter PSEUDO-Code

Given $K > 1$, n_0, $d_{\text{stop}}(n)$, δ, Ψ and an order $a_{[1]}, a_{[2]}, \ldots, a_{[n]}$ of features

Initialize $A(0) \leftarrow A$, $k \leftarrow 0$, $A^* = \{\}$ and $f^* = \infty$

Loop

$A_1(k) \leftarrow \{A \in A(k) : a_{d(k)} \notin A\}$,

$A_2(k) \leftarrow \{A \in A(k) : a_{d(k)} \notin A\}$,

$A_3(k) \leftarrow A \setminus A(k)$,

For every set $A_j(k)$

$A^j_{best}(k) \leftarrow, f^j_{best}(k) \leftarrow \infty, i \leftarrow 1$

Obtain n_0 sample sets

Calculate the first-stage sample means and variance

$$\overline{X}_j^{(1)}(k) \leftarrow \frac{1}{n_0}\sum_{i=1}^{n_0} X_{ij}(k)$$

$$S_j^2(k) \leftarrow \frac{\sum_{i=1}^{n_0}[x_{ij}(k) - X_j^{(1)}(k)]^2}{n_0 - 1}, \text{ for j=1,2,3}$$

Compute the total sample size

$$N_j(k) \leftarrow \max\left\{n_0 + 1\left\lceil \frac{h^2 S_j^2(k)}{\delta^2} \right\rceil\right\}$$

where δ is the indifference zone and h is a constant that is determined by n_0 and the minimum selection probability P^* of correct selection.

Loop

$A_{j_i}(k) \leftarrow$ Randomly select a feature subset

if $f_{j_i}(k) < f^j_{best}(k)$ then

$f^j_{best}(k) \leftarrow, f_{j_i}(k), A^j_{best}(k) \leftarrow A_{j_i}(k)$

$i \leftarrow i + 1$

Until enough feature subset samples given δ and Ψ

$j^* \leftarrow \arg\min_j f^j_{best}(k)$

if $j^* = 3$ then $A(k+1) \leftarrow A(k-1)$

else $A(k+1) \leftarrow A_{j^*}(k)$

$k \leftarrow k + 1$

End

Until $d(A(k)) = d_{stop}(n)$