# Scalable Optimization-Based Feature Selection Using Random Sampling

**Jaekyung Yang and Sigurdur Olafsson**
**Department of Industrial and Manufacturing Systems Engineering**
**Iowa State University**
**Ames, IA 50011**

## Abstract

We analyze an optimization-based approach called the NP-Filter for feature selection and show how the scalability of this method can be improved using random sampling of instances from the training data. The NP-Filter has attractive theoretical properties as the final solution quality can be quantified and it is flexible in terms of incorporating various feature evaluation methods. We show how the NP-Filter can automatically adjust to the randomness that occurs when a sample of training instances is used, and present numerical results that illustrate both this key result and the scalability improvement that are obtained.

**Keywords**
Feature Selection, Scalability, Data Mining, Optimization, Nested Partition

## 1. Introduction

Feature selection is an important problem in data mining [8]. It can be used to eliminate redundant and irrelevant features from a data set, which shortens the learning time needed for induction algorithms that are applied to the data set, and in many cases also results in more accurate predictive models. Careful feature selection can improve the scalability of a data mining system as the induction is usually much faster with fewer features, and finally, feature selection also has an inherent value in that structural insights may be obtained by learning the features that are important. Thus, considerable computation effort is often justifiable for feature selection [11].

The feature selection problem involves selecting a best subset of features from a finite subset and may be formulated as a discrete optimization problem. As such, any number of well known optimization approaches can be applied to this problem and previous work has for example used mathematical programming [2], branch-and-bound [9], genetic algorithms [15], and evolutionary search [6]. In this paper we analyze a new optimization-based approach for feature selection called the nested partitions method. In particular, we focus on using random sampling to improve the scalability of the approach with regards to increasing number of instances. This issue of scalability has received considerable attention as the size of modern databases has increased, and several researchers have used a random sampling approach in this context [12]. This includes Domingo, Gavalda, and Watanabe that take a sequential sampling approach [3], John and Langley that use a dynamic sampling methods [4], Kiven and Mannila that provide bounds for sample sizes depending on the error measure used [7], and Kolluri and Toivonen that presented sampling methods for association rules [14].

The remainder of the paper is organized as follows. In Section 2 we introduce the new methodology. The main part of the paper is Section 3, which analyzes the scalability of the method with respect to the instance dimension by using synthetic data and several well-known data sets. Finally, Section 4 contains some concluding remarks.

## 2. Methodology

The following notation will be used throughout the paper. We let $T$ denote the training data, and $m = |T|$ be the number of instances. The set of all features is denoted $A^{(ALL)}$ and the number of features is $n = |A^{(ALL)}|$. A specific feature will be denoted $a \in A^{(ALL)}$, and finally, the performance of each feature subset is defined by a function $f$, and $f^*$ :denotes the optimal performance.

The main 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. Wrapper methods use the accuracy of the resulting predictive model. Thus, to evaluate a subset of features, a predictive model is induced based on these features, and the accuracy of this model estimated, usually using a statistical such as cross-validation or bootstrapping. This is an expensive evaluation and only applies for supervised learning. Filtering methods, on the other hand, select features before any other learning algorithm is applied. Thus, a different performance measure must 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-framework can be implemented as either a wrapper or filter, resulting in the NP-Wrapper and NP-Filter algorithm, respectively [10]. In this paper, we focus an a filter employing the following correlation based measure [5]:

$$f_{correlation}(A) = \frac{k\overline{r}_{ca}}{\sqrt{k + k(k-1)\overline{r}_{aa}}}, \tag{1}$$

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

The nested partitions (NP) method is a general optimization methodology that can be applied to any combinatorial optimization problems [13]. The main idea of the method is to use iterative partitioning of the feasible region, that then creates a partitioning tree or nested partitions. Thus, in each iteration a subset of the feasible region is determined to be the most promising or most likely to contain the global optimum. This subset is then partitioned into further subsets and what remains of the feasible region aggregated into one subset called the surrounding regions. Each of these subsets is randomly sampled and based on those samples a new subset is selected. If the surrounding region is selected, the algorithm backtracks, that is, simply moves to what was previously considered the most promising region. This approach can be effectively applied to feature selection as originally described by [10], either as a filter or a wrapper, depending on how feature subsets are evaluated.

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, and we can use a two-stage sampling procedure that determines how much random sampling effort, $N(y, d)$, is needed from each region to guarantee correct selection with probability $y$ within an indifference zone $d > 0$. The two-stage sampling also allows us to further analyze the convergence of the algorithm and develop statements concerning the quality of the solution once maximum depth is reached. In particular, an expression can be derived for the probability of having found sufficiently good solution the first time maximum depth is reached:

$$\Pr\{|f(\mathsf{A}(k)) - f^*| \le d\} \ge \Psi, \tag{2}$$

Where $d > 0$ is an indifference zone, that is a performance value difference that is considered insignificant, and

$$\Psi = \frac{y^n}{(1-y)^n + y^n}, \tag{3}$$

where $y$ 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. Sometimes it may be beneficial to stop the algorithm early, that is, we can specify a stopping depth $d_{stop}(n) \le n$, define the objective function on sets of feature subsets as

$$f(\mathsf{A}(k)) = \max_{a \in \mathsf{A}(k)} f(a), \tag{4}$$

and equation (4) holds with $\Psi$ replaced with

$$\Psi' = \frac{y^{d_{stop}(n)}}{(1-y)^{d_{stop}(n)} + y^{d_{stop}(n)}}. \tag{5}$$

Partitioning for the feature selection problem reduces to determining an order for the features and then the subregions correspond to either including a feature or not including a feature. Thus, assuming that the current most promising region is some subset $\mathsf{A}(k) \subset \mathsf{A}$ of the entire feasible region, then this subset is partitioned by fixing the next feature $a$ in the order, that is, the subsets are

$$\mathsf{A}_1(k) = \{A \in \mathsf{A}(k) : a \in A\} \tag{6}$$
$$\mathsf{A}_2(k) = \{A \in \mathsf{A}(k) : a \notin A\} \tag{7}$$

The surrounding region is simply $\mathsf{A}_3(k) = \mathsf{A} \setminus \mathsf{A}(k)$. 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 (1) performs significantly better, and this partitioning is used in all of the numerical experiments below.

**NP-Filter:**

```
Given $d_{stop}(n)$, $d$, $\Psi$ and an order $a_{[1]}, a_{[2]}, \ldots, a_{[n]}$ of features
Initialize $\mathsf{A}(0) \longleftarrow A$, $k \longleftarrow 0$, $A^* = \{\}$ and $f^* = \infty$
loop
    $A_1(k) \longleftarrow \{A \in \mathsf{A}(k) : a_{d(k)} \in A\}$, $A_2(k) \longleftarrow \{A \in \mathsf{A}(k) : a_{d(k)} \notin A\}$, $A_3(k) \longleftarrow A \setminus A(k)$,
    for every set $A_j(k)$
        $A^j_{best}(k) \longleftarrow \{\}$, $f^j_{best}(k) \longleftarrow \infty$, $i \leftarrow 1$
        loop
            $A_{ji}(k) \longleftarrow$ Randomly select a feature subset
              if $f_{ji}(k) < f^j_{best}(k)$ then $f^j_{best}(k) \longleftarrow f_{ji}(k)$, $A^j_{best}(k) \longleftarrow A_{ji}(k)$
              $i \leftarrow i+1$
        until enough feature subset samples given $d$ and $\Psi$
        $j^* \leftarrow \arg\min_j f^j_{best}(k)$
        if $j^* = 3$ then $\mathsf{A}(k+1) \longleftarrow \mathsf{A}(k-1)$
        else $\mathsf{A}(k+1) \longleftarrow \mathsf{A}_{j^*}(k)$
        $k \leftarrow k+1$
    end
until $d(\mathsf{A}(k)) = d_{stop}(n)$
```

Figure 1. Pseudocode for a NP Algorithm for Feature Selection

A complete description of the NP-Filter is shown in Figure 1. Note that it uses a fixed number of $n_0$ samples to evaluate each region, starts with the set $A$ of all possible feature subsets as the most promising region, and terminates when the depth of the most promising region has reached maximum, that is, it is a singleton. We also let $A^*$ be the best feature subset found and $f^*$ be the corresponding performance value, which is calculated according to equation (1) above.

## 3. Scalability of NP-Filter

In this section we consider the scalability of the new methodology. In particular, we evaluate the accuracy and computational time as functions of both the number of features and number of instances. Ideally, a highly scalable algorithm would achieve linear growth in the computational time while maintaining the accuracy level.

For these tests we use synthetically generated test data where both the number of instances and number of features are control parameters. In particular, we generate test sets with {50, 100, 200, 400, 800} instances, and {50,100, 200, 400, 800} features using the following approach. To create a single instance $i$, a value for the class feature $Y_i$ is generated according to a uniform distribution over the interval [-3,3], The value for each of the other features $X_{ij}$ is then generated according to $X_{ij} = r_j Y_i + (|r_j| - 1) \cdot Z_j$ where $r_j$ is the amount of correlation between feature $j$ and the class feature, and $Z_j$ is drawn from a unit normal distribution, $j = 1,2,\ldots,n$, $i = 1,2,\ldots,m$. For each of the test problems, 10% of the features are highly correlated with $|r_j| \geq 0.9$, 40% have correlation $0.3 \leq |r_j| < 0.9$, and 50% of the features do not correlate highly with the class feature, that is $|r_j| < 0.3$. The NP-Filter, followed by Naive Bayes classification model induction, is run five times for each of those test sets.

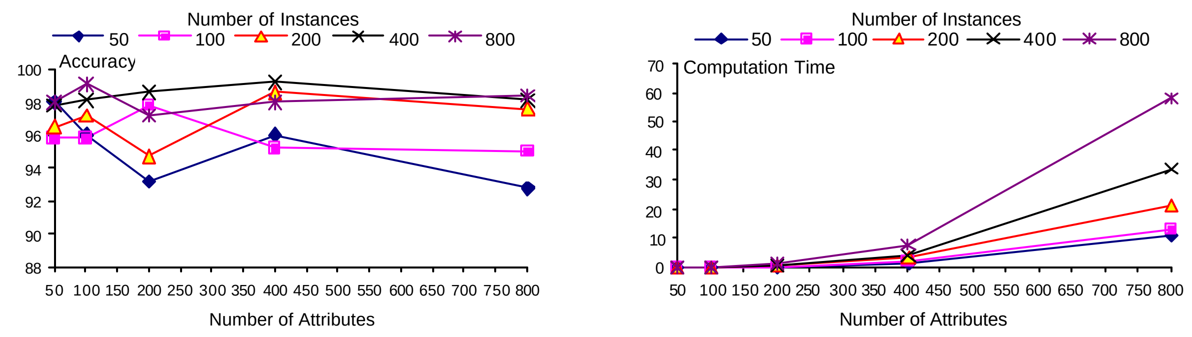

Figure 2. Accuracy and computation time as a function of features for the five instance settings (50 to 800).

Lets first consider the scalability with respect to the number of features. Figure 2 shows the accuracy and computation time as a function of number of features for the five instance settings (50 to 800 instances). From these results (left) we conclude that there is no significant change in the accuracy obtained as the number of features grows. The results for computational time (right) report that the time grows rapidly as the number of features increases as clearly shown in 400 and 800 instance settings, and indeed it appears to demonstrate exponential growth. Thus, although quality is not lost as the problem size increases, the time it takes to achieve this quality increases quickly and the NP-Filter is therefore somewhat lacking in terms of scalability with respect to the number of features. Addressing this issue is an important topic of future research.

Finally, looking at the scalability of the NP-Filter as a function of number of instances, Figure 3 reports the accuracy and computation time obtained as a function of the number of instances. An interesting observation from this figure is that the solution quality actually improves as the problem size increases, which is not entirely surprising as more instances imply more data is available to induce a model with high accuracy. Now turning to the computational time required to achieve this accuracy, As opposed to the rapid growth in computational time seen when the number of features increases, the time here grows only linearly, thus implying that the NP-Filter is scalable with respect to the number of instances **.**

In the NP method, a new set of instances is sampled in each iteration in such a way that this set is independent of the previous set. 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. As the proportion is decreased, more backtracking is required because 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 four well-known data sets described in Table 1 below [1].

We evaluate the estimated accuracy as well as the computation time when either 5%, 10%, 20%, 40%, 80%, or 100% of the instances is used by the NP-Filter. For example, when testing the 'vote' data with 20%, we set $v(435) = 0.2 \cdot 435 = 87$ instances. Other parameters are set as follows. The sampling effort is constant $N(y,d) = 5$ so $y$ and $d$ need not be specified, the stopping depth is maximum depth $d_{stop}(n) = n$, the order $a_{[1]}, a_{[2]}, \ldots, a_{[n]}$ is determined by the information gain. The results are reported as average and estimated standard deviation over five replications, and are shown in Table 2.

Table 1: Characteristics of the Tested Data Sets

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

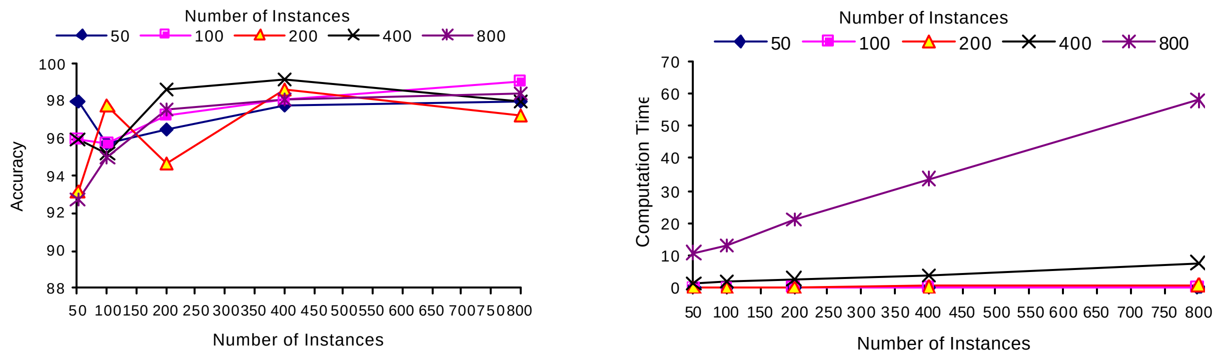

Figure 3. Accuracy and computation time as a function of instances for the five feature settings (50 to 800).

First note that the desired speedups in the algorithm are indeed achieved. By using 20% of the instances rather than 100% of the instance, the computing time is reduced by 57%, 27%, 25%, and 84% for the four data sets, respectively. Due to very high variance, however, we cannot say that this difference is significant for the 'audiology' data set. The cost of the speedup should be in terms of decreased accuracy (performance). However, the only data set that shows a significant difference is the 'audiology' data, where the estimated accuracy goes from 70.7% to 60.4%, a decrease of about 10% in performance. These are encouraging results; however, we should note that for all of the data sets the variability of the performance increases significantly. For example when looking at the 'cancer' data set, although the average performance increases slightly (73.9% versus 77.9%) when using only 20% of the instances, the estimated standard deviation goes up substantially (0.5 versus 5.2). For all data sets above, when the proportion of instances is very low, the variability of the performances gets bigger. This is to be expected as using fewer instances corresponds to the performance estimates used by the algorithm being more noisy.

Table 2: Effect of Using Fraction of Instance Space

| Data Set | Fraction | Accuracy | Speed (millisec) | Backtracking |
|---|---|---|---|---|
| vote | 100% | 93.3±1.3 | 3062±641 | 0±0 |
| | 80% | 94.3±0.9 | 2351±62 | 0±0 |
| | 40% | 93.4±2.6 | 1654±50 | 0±0 |
| | 20% | 92.9±3.0 | 1324±42 | 0±0 |
| | 10% | 89.8±3.5 | 1219±116 | 2.4±2.5 |
| | 5% | 86.7±5.2 | 1139±132 | 4.0±2.6 |
| audiology | 100% | 70.7±1.5 | 46976±12485 | 9.2±20.6 |
| | 80% | 75.1±4.6 | 85794±50447 | 142.8±127.4 |
| | 40% | 69.1±5.3 | 72690±30780 | 214.6±119.4 |
| | 20% | 60.4±2.4 | 34433±3849 | 189.0±32.0 |
| | 10% | 54.5±7.2 | 74208±57467 | 694.2±496.7 |
| | 5% | 40.0±17.7 | 65213±25911 | 866.24±362.2 |
| cancer | 100% | 73.9±0.5 | 1061±36 | 0±0 |
| | 80% | 73.4±3.3 | 985±67 | 0.6±0.9 |
| | 40% | 73.5±3.1 | 820±40 | 0.8±1.3 |
| | 20% | 77.9±5.2 | 791±76 | 2.4±2.6 |
| | 10% | 75.7±9.2 | 819±144 | 7.4±7.5 |
| | 5% | 75.7±6.4 | 2359±1227 | 45.6±33.8 |
| kr-vs-kp | 100% | 84.7±5.0 | 105994±11078 | 0±0 |
| | 80% | 85.4±6.8 | 89064±11079 | 0.2±0.4 |
| | 40% | 86.3±6.7 | 40708±3711 | 0±0 |
| | 20% | 90.9±2.2 | 16792±1459 | 0±0 |
| | 10% | 89.2±2.5 | 9956±1373 | 0±0 |
| | 5% | 81.8±6.1 | 18736±27060 | 101.8±220.9 |

The NP-Filter 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. This is indeed supported by the data in Table 2, as the average number of backtracking moves increases for each of the data sets. Even though it is expected that the speed would become slow as the proportion of instances decreases, the real time increases at the very lower proportion point after it decreases for a while. Excessive backtracking may slow down the NP-Filter. These results illustrate that sampling of instances is a reasonable way to improve the scalability of the NP-Filter with respect to large number of instances. But there would an optimal proportion point of instances, which should be researched more for finding the optimal point if it really exists. However, the effectiveness of this approach will in general depend on the particular data set being analyzed.

## 5. Conclusion

In this paper we consider a new optimization-based feature selection methodology called the NP-Filter that has been shown to be capable of finding very high quality feature subsets. However, the method is fairly computationally intensive and may not scale well to large number of instances. The main contribution of the paper is new methodology that uses random sampling of instances to improve the scalability of the NP-Filter. In particular, we have shown how the NP-Filter can use backtracking to correct any bias that may arise due to sampling variability. Furthermore, we show that by using the new sampling approach, significant speedups in computation time can be achieved. These conclusions are supported by numerical results on well known realistic data sets.

Our future work will focus on identifying relationships between the sampling levels and characteristics of the data set and developing a theoretical basis for determining optimal sampling percentage.

## References

1. Blake, C.L. and Merz, C.J., 1998, *UCI Repository of machine learning databases* <http://www.ics.uci.edu/mlearn/MLRepository.html>, University of California, Irvine, CA.
2. Bradley, P.S., Mangasarian, O.L., and Street, W.N., 1998, “Feature selection via mathematical programming”, *INFORMS Journal on Computing*, 10(2), 209-217.
3. Domingo, C. Gavalda R., and Watanabe, R., 2000, “Adaptive Sampling Methods for Scaling Up Knowledge Discovery Algorithms”, *Journal of Knowledge Discovery and Data Mining*.
4. John, G. and Langley, P., 1996,. “Static versus Dynamic Sampling for Data Mining”, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 367-370.
5. Hall, M.A., 1998, “Correlation-based feature selection for discrete and numeric class machine learning”, *Proceedings of the Seventeenth Int. Conf. on Machine Learning*, Stanford University, CA. Morgan Kaufmann.
6. Kim, Y.S., Street, W.N, and Menczer, F., 2000, “Feature selection in unsupervised learning via evolutionary search”, *Proceedings of the 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*
7. Kiven, J. and Mannila, H., 1994, “The Power of Sampling in Knowledge Discovery”, in *ACM Symposium on Principles of Database Theory*, 77-85.
8. Liu, H. and Motoda, H., 1998, *Feature Extraction, Construction and Selection: A Data Mining Perspective*, Kluwer Academic Publishers.
9. Narendra, P.M., and Fukunaga, K. (1977). “A branch and bound algorithm for feature subset selection”, *IEEE Transactions on Computers*, 26(9), 917-922.
10. Olafsson, S. and Yang, J., 2001, “Intelligent Partitioning for Feature Relevance Analysis”, *INFORMS Journal on Computing*, (submitted).
11. Olafsson, S. and Yang, J., 2001, “Scalable Optimization-Based Feature Selection”, *Proceedings of Workshop on Discrete Mathematics and Data Mining* (2nd SIAM Conference on Data Mining), Arlington, VA.
12. Provost, F. and Kolluri, V., 1999, “A Survey of Methods for Scaling up Inductive Algorithms”, *Data Mining and Knowledge Discovery 3*: 131-169.
13. Shi, L. and Olafsson, S., 2000, “Nested partitions method for global optimization”, *Operations Research*, 48, 390-407.
14. Toivonen, H., 1996, “Sampling Large Databases for Association Rules”, in *Proceedings of the 22$^{nd}$ International Conference on Very Large Databases*, 134-145.
15. Yang, J. and Honavar, V., 1998, “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.