Column Removal During Progressive Sampling

Gabor Melli, Siavash Amirrezvani, Felix Chen, Neil Russell
PredictionWorks, Research Centre
2628 Granville Street, Vancouver, B.C., CANADA, V6H 3H8
{gmelli, siavash, felix, ngr}@predictionworks.com

Abstract

Sampling and column (variable) selection are commonly used to gain insights that improve data mining performance on very large databases. This paper proposes an algorithm named CRPS (for Column Removal during Progressive Sampling) that integrates sampling and column reduction. This algorithm delivers a predictive model in less time than current approaches in tasks where not all columns are relevant. CRPS works in conjunction with predictive modeling algorithms that decide which columns to include in their models such as decision tree and logistic regression. The CRPS algorithm iteratively models on progressively larger samples and also removes some of the columns that were not used in the previous sample in order to reduce the amount of overall work. Other advantages of CRPS include the requirement of less memory to process the dataset, a ranking of relevance for each column, and the generation of simpler models based on fewer columns. The algorithm's time complexity is dependent on the number of columns in the final model and not the number of columns in the dataset. In the worst case when all columns are required by the model, CRPS does not increase the time complexity of progressive sampling, but could double the time of directly modeling all of the data. In more practical cases where some of columns are not relevant to the task, CRPS offers an advantage over traditional predictive modeling. Empirical results validate CRPS's efficiency with no degradation in accuracy.

1 Introduction

Predictive data mining algorithms are required to efficiently produce accurate and simple models for datasets with large numbers of rows (instances) and columns (features). Research into efficiency has focused on building faster algorithms, but some work has also gone into reducing the number of rows and columns to be evaluated. While there has been progress to efficiently reduce the number of rows from the search, such as through "progressive sampling" in [14], less progress has been made to efficiently reduce the number of columns. Much of the research in column reduction focuses on accuracy and model simplification at the expense of time. Given m columns, the successful "wrapper" approach for example adds a factor of m^2 to the underlying algorithm [8]. Reducing the number of columns, however, has a greater impact on computational complexity than reducing rows. The "curse of dimensionality" implies that most state-of-the-art induction algorithms have greater than linear time complexity on the number of columns while only linear time complexity on the number of rows.

In this paper we propose an efficient column reduction algorithm named CRPS that adds no computational factor to the overall processing time. The algorithm in fact changes the computational complexity of an underlying inductive algorithm from $O(n,m)$, where n is the number or rows and m is the number of columns, to $O(n,j)$ where j is the number of relevant columns to the underlying algorithm. As datasets become wider, such as the datasets presented during the KDD conference competition, greater attention will be paid to algorithms whose time complexity is measured not by the total number of columns in the dataset, but by the much smaller number of relevant columns.

The essence of CRPS is to use some of the column relevance information available from modeling smaller samples to eliminate irrelevant columns for larger samples of the dataset. Specifically, the proposed algorithm builds models on increasingly larger data samples while constraining the number of columns considered until no more reduction occurs. The name CRPS is derived from the phrase Column Reduction during Progressive Sampling. To efficiently detect which columns appear to be relevant in any one sample, the CRPS algorithm works by using state-of-the-art predictive modeling algorithms that explicitly decide which columns to include in their model. Two such algorithms are decision tree and logistic regression. In a sense, CRPS uses the underlying algorithm's ability to avoid overfitting to determine which columns are not relevant to the task at hand.

There are other valuable consequences from the use of CRPS. One useful output is the ranking of each column's relevance from sample to sample. This information allows an analyst to focus their attention on fewer columns. Fewer columns also mean that the final process against all the rows in the dataset will require less memory. This would allow larger datasets to be processed in memory. Fewer columns may also be referenced by CRPS generated models due to the removal of irrelevant columns during earlier samples. This exclusion of irrelevant columns may also increase accuracy. However, if relevant columns are accidentally removed then the final model may turn out to be less accurate. Empirical testing shows that CRPS requires considerably less time to deliver its model, produces somewhat simpler models, and does not impact accuracy.

The subsequent sections of the paper describe the CRPS algorithm in detail and analyze the algorithm's time complexity. Empirical results are then presented. The paper concludes with a summary of related research and a list of open research questions.

2 The CRPS Algorithm

In this section we present the details of the CRPS algorithm, which performs column reduction during progressive sampling. The algorithm is used in tandem with model-based predictive modeling algorithms that explicitly chose which columns to include in their output. After each sample is trained the information about which columns were used in the model progress to be included in the modeling of a larger sample. Randomly selected portions of the remaining columns are also included. As sampling proceeds, the number of randomly chosen columns is slowly decreased. As the sample size increases the number of columns used will begin to converge with the number of relevant columns required to produce the model. Table 1 presents the steps followed by CRPS.

Table 1 - Overview of CRPS algorithm

Start with all columns in subset c_i
For each sample n_i in sampling schedule N
  Construct model on rows n_i and columns c_i
  $\phi()$ = relevant columns from model
  $\delta()$ = a subset of unused columns
  New column subset $c_i = \phi() \cup \delta()$
Return last constructed model

The next sections describe CRPS' sampling schedule N, its column relevancy function $\phi()$, and its column removal schedule C that is created by the unused column selector $\delta()$. A more detailed algorithm is presented afterwards along with a computational complexity analysis.

2.1 Sampling Schedule - *N*

Several sampling schedules $N = \{n_0, n_1, n_2, \ldots, n_k\}$ are possible for given a dataset *N* with *n* rows. The value *k* will refer to the final sample to be tested. To achieve the goals of efficient information passing the schedule must continually increase until $n_k = n$. Of the approaches that were tested the geometric schedule proposed in [[14]] was finally selected. This choice became apparent after tests on simpler approaches such as the use of only one sample ($k=1$). The use of a single small sample to determine column relevancy always resulted in significant reduction in processing time was, but the final model's accuracy suffered. The less aggressive approach of using a single large sample to determined relevancy, say one-half of the rows, proved to be effective in terms of accuracy, but offered little computational advantage. The lack of advantage in speed-up is due to the fact that the modeling the large sample will consume half of the time required to build the entire dataset. Although using one single sample is not an effective approach to column reduction, the insight that some column relevance can be detected from samples motivated further research.

The fact that the use of one small sample occasionally achieved good results inspired the creation of a progressive sampling schedule that reduced the number of columns tested after each sample. To achieve significant gains in computational time the sample sizes were increased faster than the linear (arithmetic) rate proposed in [JOHN96]. The specific schedule used was $n_0=250$ and a doubling afterwards, or $N = \{n_0=250\times2^0, n_1=250\times2^1, n_2=250\times2^2, \ldots, n_k\}$. This allowed for more work to be done on smaller samples that were computationally inexpensive. By the time that the larger samples were tested a significant number of irrelevant columns would be eliminated. Another advantage to this schedule is that it offers a natural fit with the schedule proposed in [[14]]. The progressive sampling approach has been shown to be effective in the discovery of the minimum number of rows required to attain a plateau in accuracy. According to their research a starting number of $n_0=250$ is acceptable.

Although the proposed schedule can be produced before data processing, there are opportunities to skip past some samples. For example, in the case where no progress in column reduction will occur from a specific sample onward, testing can skip directly to the final sample. As demonstrated in the empirical tests, this opportunity occurred with the 'letter' dataset which has only relevant columns.

2.2 Column Relevancy Function - $\phi()$

To achieve its scale-up objective the CRPS algorithm requires an inexpensive method to detect column relevance information after each sample has been modeled. The column relevancy function used in CRPS is simply to review which columns where used in the model and which were not. The intuition behind this approach is that state-of-the-art induction algorithms have evolved significant abilities to avoid overfitting within their stopping criterions and pruning techniques. This approach has also been successfully used by [4] to pass column relevance information from a decision tree to a nearest-neighbor algorithm.

CRPS should be applicable to any induction algorithms that explicitly choose their columns. Examples include decision tree algorithms such as C4.5 [15], forward-selection logistic regression [1], and the naïve-bayes classifier [5]. Even algorithms that do not create a model such as instance-based approaches commonly perform a column relevancy step in advance [2]. One possible exception to this rule includes neural networks, which commonly operate as a black box.

Once a model is built for a sample the column relevancy function simply traverses the model to extract the set of relevant columns for that sample. The column relevance function

will be labeled as $\phi(M_i)$, where M_i is the model constructed on the i^{th} sample. For the logistic regression algorithm, traversal detects which columns were used as one of the coefficients of the equation. If, for example, the equation contains five coefficients then the column relevance function would return the five (5) columns associated to each coefficient. For decision tree algorithm, the relevancy function returns the set column used within the internal nodes of the tree as proposed in [4].

2.3 Column Removal Schedule - *C*

Given a sampling schedule *N* and a column relevancy function $\phi()$ the remaining question for iterative column reduction is the rate at which columns should be removed from sample to sample. CRPS column reduction schedule $C = \{c_0, c_1, c_2, \ldots, c_k\}$ is dynamically created as each sample is analyzed. Each c_i represents the columns that, along with rows in sample n_i, are analyzed by the underlying algorithm *A()*. To reduce the amount of effort expended on larger samples the proposed schedule reduces columns most quickly during the early stages of the sampling. By the time that the larger samples are modeled most of the irrelevant columns would have been identified and discarded. Another consideration for the removal schedule is that as the sample size n_i approaches the size of the entire dataset n_k then most, if not all, of the relevant columns should have been detected. Therefore, the last modeling step should not consider any random columns whatsoever and it should base column relevance solely on the second-to-last n_{k-1} modeling step.

Each column subset, c_i, in the column removal schedule *C* is based on relevant columns from the past sample $\phi(M_{i-1})$ and a subset from the remaining columns $\delta()$. The subset of remaining columns is based on the sample number *i* and how close the process is to the final sample n_i:

$$c_i = \phi(M_{i-1}) + \delta(n_i, i)$$

From sample to sample random column selector $\delta(n_i, i)$ returns fewer columns with the use of the following function:

$$\delta(n_i, i) = i^{-1} \times (|n| - |n_i|)/|n|.$$

Assume a dataset with 36 columns and 8,000 rows, and also a first sample size of two hundred fifty ($|n_0|$=250) rows. All columns are used to build a model on sample s_0. Assume that eight (8) columns were used in the model on sample n_0. When the next model is built on a sample with five hundred rows ($|n_1|$=500) only twenty-one (21) columns are used. The twenty-one columns include the eight columns deemed relevant in the s_0 model, and an additional thirteen ($\delta(n_1,1)$=13) columns that are randomly selected. The number thirteen is determined with the following calculation ((36-8) × 1/2 × (8000-500)/8000 = 8 + (28 × 1/2 × 15/16). If the number of relevant columns from sample to sample stays at 8 columns then the final schedule would be: $C = \{c_0=36, c_1=21, c_2=15, c_3=13, c_4=10, c_k=8\}$

2.4 Algorithm Analysis

The effect on time complexity from the combination of the three components above is dependent on the number of rows *n* and columns *m*. The complexity is also dependent on the number of relevant columns discovered at each sample, j_i. Table 2 presents a more detailed description of the algorithm under analysis.

Table 2: The CRPS algorithm.

Inputs: S is a set of examples
A(n_i, c_i) an induction algorithm that accepts a set of rows n_i and a set of columns c_i

Outputs: M - a predictive model
R - a list of ranked columns

Procedure CRPS (S, A)

n = row count in S
$i = 0$ *// set sample counter*
$n_i = 250$ *// starting sample size*
c_i = S.*cols* *// working set of columns*
WHILE $n_i \le n$ *// while more rows to process*
M_i = model induced by A(n_i, c_i)
$\delta(n_i,i) = i^{-1} \times (|n|-|n_i|)/|n|$
$c_i = \phi(M_{i-1}) \cup (S.cols - \phi(M_{i-1})) \times \delta(n_i,i))$
$i = i + 1$ *// increment sample counter*
$n_i = n_i \times 2$ *// double the sample size*
END WHILE

The worst-case time complexity for CRPS occurs when all, or almost all, of the columns are labeled as relevant at each sample. When this is the case CRPS will create a model based on all of the rows and columns at its last step, but will also have trained models on several intermediary samples. Assume that the computational complexity of the underlying modeling algorithm $O(f(m,n))$ has linear complexity on the number of rows n. At the last sample, the time required is $f(m,n)$. Because of the geometric sampling schedule the second-to-last sample processes only half of the rows, $n_{k-1} = n_k/2$, but all of the columns because they are all relevant. The time required at this step is $f(m,n/2)$. As we work backwards through the samples the computational sequence becomes $f(m,n) + f(m,n/2) + f(m,n/4) + \ldots + f(m,n_0)$. The sequence can be rewritten as $f(m,n) \times (1 + ½ + ¼ \ldots)$ which at the limit becomes $f(m,n) \times 2$. In the worst case, CRPS requires twice as much time as modeling the entire dataset. This complexity is identical to that of progressive sampling.

Although many datasets show worst-case characteristics, the intention is to apply CRPS to datasets with a substantial proportion of irrelevant attributes. The average-case complexity requires several assumptions to facilitate its calculation. The first assumption is that the underlying induction algorithm has linear complexity on both n and on m, $O(f(n,m)) = nm$. In general a state-of-the-art algorithm will have greater than linear complexity on the number of columns. Therefore this is a pessimistic assumption because CRPS will not recognize the benefit that it will likely recognize in practice from having reduced the number of columns. Halving the number of columns on the last and most time consuming step, for example, will usually save more than half of the processing time. The next assumption is that the number of relevant columns at each sample $\delta(M_i)$ remains at constant value j ($j<m$). Because of the linear assumption on $f(m,n)$ the processing can be divided into two components: the processing on the j relevant columns, and the processing on the randomly selected columns in schedule C.

At the limit the work on relevant columns approaches $2nj$ as it did for the worst-case on all column. The work that occurs above the darker line in Figure 1 is the work performed on non-

relevant columns. The work performed follows the sequence: $250\times(m\text{-}j)/1 + 250\times2\times(m\text{-}j)/2 + \ldots + 0$ which reduces to $(m\text{-}j) \times n_0 \log_2(n/n_0)$. Because this function is contained by jn, the average case complexity of CRPS, given the assumptions stated earlier, is jn.

The best case occurs when no columns are labeled relevant j_i=0. The sum of the area still holds $f(n,m,j) = jn + (m\text{-}j)log(n)$. With j=0 however the function reduces to $mlog(n)$. The best case complexity is $O(f())=mlog(n)$.

2.5 Example

This section illustrates the application of the CRPS algorithm to one of the datasets from the empirical tests: mushroom. The presentation will facilitate the discussions on empirical results and comparisons. The mushroom dataset is a two-class problem (edible/nonedible) with 8,196 rows and 22 columns of low-order nominal datatype. This is a small dataset, however, it is still a good candidate for CRPS because less than one-half of the 22 columns are used by the model when trained on the entire dataset. The underlying induction algorithm used in this example was Logistic Regression.

On the first iteration a model is built on 250 random rows and all of the columns. Assume that 9 columns were used in the model. On the second iteration a model is built on 500 random rows, the 9 'relevant' **columns** used in the previous model and (36-9)/2 random columns. The process iterates until all of the rows are modeled.

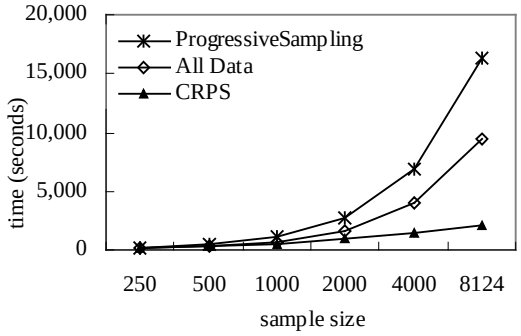

Figure 1 - The time required by Logistic Regression to process the mushroom dataset at different sample points. The "All Data" data is for non-progressive training of models for that given sample size.

The savings in time between CRPS and progressive sample are apparent in Figure 2 and Figure 3. As the sample size increases the savings in time between

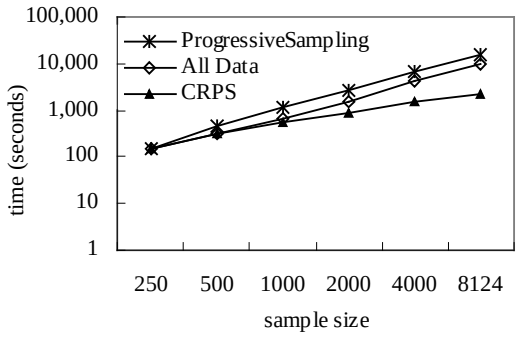

Figure 3 - The time required by Logistic Regression to process the mushroom dataset at different sample points. This figure is identical to Figure 2 except with the y-axis is in logarithmic scale.

CRPS and progressive sampling increase. Interestingly there also appears to be a substantial time savings between CRPS and modeling all of the data just once. The improvement appears to grow as the sample size increases. The effect of CRPS on model accuracy is illustrated in Figure 4. The effect on this dataset is negligible relative to modeling all of the data

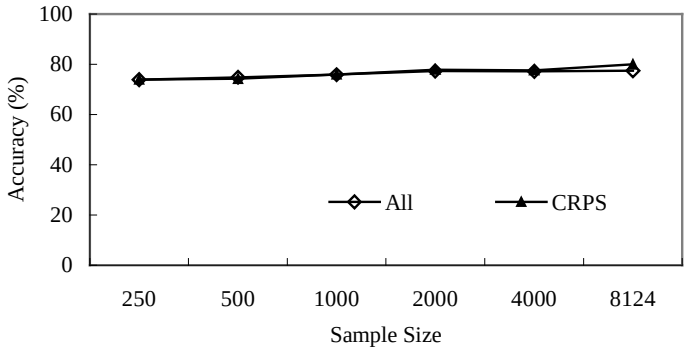

Figure 4 - The accuracy achieved by Logistic Regression on the mushroom dataset at different sample points.

Given the improvement in time relative to both progressive sampling and simple modeling, with no degradation in accuracy, the CRPS approach would be selected for this specific dataset.

3 Empirical Results

The use of column removal during progressive sampling is meant to scale-up the underlying induction algorithm and will ideally approximate the speed of modeling with just the relevant columns. This section presents the results of an empirical study that tests the validity of the CRPS algorithm with the use of both a decision tree and a logistic regression algorithm. The CRPS algorithm is incorporated into the on-line service at www.predictionworks.com/analyze/.

Tests were performed against synthetic and well-known datasets. The tests on well-known datasets were performed to determine the applicability of CRPS to real challenges. Testing was also performed on a synthetic dataset to better understand the algorithm's behavior as the number of irrelevant columns increased. In summary, the tests validate the contribution of the CRPS algorithm against plain progressive sampling and also against simple modeling on larger datasets.

3.1 Results on Benchmark Datasets

The CRPS algorithm was tested against several well-known datasets to validate that the algorithm is fast and accurate. Most datasets were obtained from the UCI repository [11]. The selection criteria were for datasets also used in [8] or in [14], and with more than 3,000 rows and 15 dimensions. In [10], Langley notes that most UCI datasets do not have many irrelevant columns because experts often created them with a specific purpose in mind. Datasets in data mining are often 'wide' because they serve multiple purposes. The KDD98 Cup dataset was also included to present a more realist data mining task.

Table 3: Summary of datasets. The last column is an approximation of the number of relevant columns.

Dataset	Rows:n	Cols:m	Rel.Cols:j
chess	3,196	36	19
waveform	5,000	40	20
mushroom	8,129	22	7
letter	20,000	16	15
KDDCup98	95,431	481	51

CRPS performance is compared against both "Progressive Sampling", as defined in [14], and to the simple modeling of "All" of the data. Where possible results for both decision tree (DT) and logistic regression (LR) are presented.

Table 4 shows that CRPS performed faster than Progressive Sampling on all datasets (greater than 1.0 ratio). Except for the Chess dataset and decision tree (DT) algorithm, CRPS was also generally faster than modeling "All" of the data. The speed-up was most noticeable for the KDD98 dataset, with a 4× and 8× multiplier in speed. CRPS also improves logistic regression's performance better than decision tree's because logistic regression "binarizes" nominal columns. This process multiplies the number of columns, many of which then become irrelevant.

Table 4 - Total train time comparisons.

Alg	Dataset	CRPS time (sec.)	Ratio to All Data	Ratio to Prog. Samp.
DT	Chess	3.0	0.64	1.38
DT	Mshrm	4.1	1.27	1.74
DT	Wave	7.3	1.26	2.14
DT	Letter	9.3	1.00	*NA*
DT	Letter	15.6	*NA*	1.84
DT	KDD98	927.0	3.98	7.39
LR	Chess	959.6	1.12	2.16
LR	Mshrm	2151.9	4.38	8.56

A 10-fold cross validation study was performed to determine the impact of CRPS on accuracy. Table 5 demonstrates that CRPS appears to have negligible effect on accuracy. Ideally the CRPS algorithm's removal of irrelevant columns would have improved accuracy as observed in other feature selection mechanisms. One possible interpretation of this effect is that irrelevant columns that tend to be included in the final model also tend to be included in models on smaller data samples.

Table 5 - Accuracy comparisons in percent. (ratio to CRPS results)

Alg	Dataset	CRPS	All Data	Ratio
DT	chess	94.45	94.14	1.00
DT	wave.	67.08	67.04	1.00
DT	mushr.	98.56	98.56	1.00
DT	letter	56.06	56.05	1.00
DT	kdd98	95.06	95.33	1.00
LR	chess	90.08	90.08	1.00
LR	mushr.	78.40	78.40	1.00

The final empirical test on benchmark datasets was an analysis of model simplicity. Table 6 below demonstrates that the models create by CRPS generally refer to fewer columns than a model built on all of the data. Fewer columns in the outcome model simplifies discussion about the factors that predict future behavior.

Table 6 - Number of relevant columns in final model

Alg.	Dataset	CRPS	All Data	Ratio
DT	chess	19	20	0.95
DT	mushr.	7	8	0.88
DT	letter	14	16	0.88
DT	wave.	14	25	0.56
DT	kdd98	49	52	0.94
LR	chess	5	5	1.00
LR	mushr.	4	4	1.00

3.2 Synthetic Datasets with Irrelevant Columns

Tests on synthetic datasets were performed to better understand the relationship between the proportion of irrelevant columns and CRPS performance. To control for this proportion a dataset with only relevant columns was updated by adding an increasing number of random columns. The foundational dataset with only relevant columns was composed of the nine (9) columns in the mushroom dataset that were considered relevant by the decision tree algorithm. Columns of random values of alternating nominal, ordinal, and continuous data types were then added. Figure 5 shows a growing improvement in time performance by CRPS as the number of irrelevant columns increase. Accuracy remains unaffected.

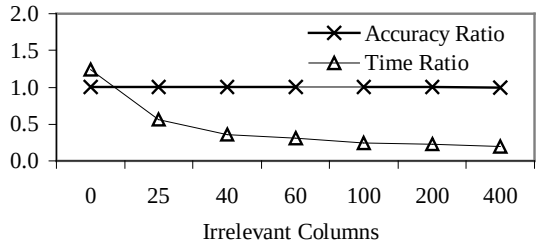

Figure 5 - The relative accuracy and time expended, between CRPS and modeling the entire dataset, as more irrelevant columns are added.

As mentioned in the discussion of the column reduction schedule the synthetic datasets tests illustrate further opportunity for column reduction. The first function to be tested was based on the inverse of the number of samples already considered $1/i$. This function resulted in no improvement in performance as more irrelevant columns were added. The reason for the plateau is due to always processing 1/k columns in the last sample. Because the synthetic dataset's 8,000 rows are covered after k=5 samples, the removal function would always process at least 1/5 of all of the columns. The removal function was modified to ensure that by the time that the last sample was considered only the columns found to be relevant in the second-to-last sample (k-1) would be considered. As Figure 5 illustrates this update allowed for continual improvement as more irrelevant columns were added without a negative impact on accuracy.

4 Related Work

CRPS is closely related to past work on feature selection and sampling. Even without access to automated approaches a data miner will commonly first work with smaller samples to understand the properties of the task and reduce the number of columns based on past experience before proceeding with the entire dataset. The automation of these two techniques for scaling induction algorithms is formally treated by Provost and Kolluri in [13]. Under the work's section on "data partitioning" the two methods are covered under "select feature subsets" and "process sequential subsets".

The two well-known feature selection approaches in the literature are filters [9] and wrappers [8]. The filter approach is unsupervised and uses techniques such as factor analysis and principal components analysis to identify redundant columns or to compress columns into a smaller set of dimensions. The wrapper approach is supervised and is used to isolate an optimal feature subset by considering how the modeling algorithm and the training set interact. Because of the large number of possible column combinations (2^m), both approaches use heuristics to contain the search space, usually to polynomial time (m^2).

Empirical testing has validated the increased accuracy and decreased model complexity associated with feature selection [2,4,8,9,10,12]. As Provost *et al* observe in [13], however, far less research into feature selection has focused on scaling. The use of feature selection as a scale-up technique offers significant opportunity since most state-of-the-art algorithms have worse than linear performance on the number of features. Model-building algorithms often achieve a computational complexity of $O(nm^2)$ [13]. The association between the number of features and performance is commonly referred to as the 'curse of dimensionality' [3] due to the fact that the search space grows exponentially with each added column.

Another feature selection approach closely related to CRPS is the use of the columns used in a decision tree to improve the accuracy of a nearest-neighbor algorithm proposed by Cardie in [4]. Kohavi and John note in [8] however that this approach of feature filtering likely improves performance due to the removal of irrelevant columns but will likely discard some relevant features due to fragmentation, and also keep some redundant features. Because CRPS is not acting as a feature filter for a different algorithm there is less concern of discarding relevant features. The presence of some redundant features remains but the improved scale-up offered by CRPS is a worthy compromise.

While feature selection addresses the impact on performance of smaller feature subsets, sampling investigates the impact from smaller case subsets. As more cases are analyzed, accuracy tends to increase but then slowly plateaus. Friedman also makes the case in [6] that a slow and powerful algorithm working on samples may provide superior accuracy over a fast and naive approach on the entire dataset. In [14] the use of a geometric schedule was proposed to locate the minimal number of cases required to attain the task's accuracy plateau based on its learning curve. If fewer rows are needed to reach the plateau then a substantial time improvement is achieved. The research concludes that "in a wide variety of realistic circumstances, progressive sampling is preferable to analyzing all instances from a database". The CRPS algorithm strengthens the case for progressive sampling.

5 Conclusions

This work presented the integration of feature selection and sampling to scale up model-based induction algorithms such as decision tree and logistic regression. The proposed CRPS algorithm reduces the underlying modeling algorithm's time complexity so that it is based on the number of columns in the final model instead of the full set of columns in the dataset. The work demonstrates that all future progressive sampling schedules should now consider passing column relevance information from sample to sample. Empirical results demonstrate that CRPS reduces time complexity, memory usage, and model complexity, without affecting accuracy. Even when being compared to the simple modeling of the entire dataset, CRPS is shown to be effective when datasets contain even a small proportion of nonrelevant columns. On large datasets, such as the KDD98 dataset, CRPS is shown to be an order of magnitude faster than progressive sampling and four times faster than simple modeling. The algorithm also produces a valuable ranking of each column's relevance.

Several future research challenges are apparent. The CRPS algorithm should be extensible to other popular model-building algorithms such as Naive Bayes [5] but this has yet to be proven. The CRPS column removal function may also be improved by noting the amount of chance in feature subset membership from sample to sample. Finally, the application of information passing from sample to sample shows promise as a general scale-up technique. Once the subset membership has stabilized, for example, other more robust feature selection methods, such as wrappers, may be attempted. The removal of uninformative rows during progressive sampling is another topic that should be investigated.

7 References

[1] Agresti, A. (1990), *Categorical Data Analysis*. New York: John Wiley.

[2] Aha, D. (1998), *Feature weighting for lazy learning algorithms*. In: H. Liu and H. Motoda (Eds.) Feature Extraction, Construction and Selection: A Data Mining Perspective. Norwell MA: Kluwer.

[3] Bellman, R. E. (1961), *Adaptive Control Processes*. Princeton University Press, Princeton, NJ.

[4] Cardie, C. (1993), *Using decision trees to improve case-based learning*, in Proceedings of the Tenth International Conference on Machine Learning, Morgani Kaufmann Publishers, Inc., pp. 25-32.

[5] Domingos, P. & Pazzani, M. (1996), *Beyond independence: Conditions for the optimality of simple Bayesian classifier*. In Machine Learning: Proceedings of the Thirteenth International Conference on Machine Learning. Morgan Kaufmann.

[6] Friedman, J. H. (1997), *Data mining and statistics: What 's the connection?* In Proceedings of the 29th Symposium on the Interface Between Computer Science and Statistics.

[7] John, G., & Langley, P. (1996), *Static versus dynamic sampling for data mining*. Proc. Second Intrl. Conf. on Knowledge Discovery and Data Mining pp. 367-370. Portland, OR: AAAI Press.

[8] Kohavi, R. and John, G. (1997), *Wrappers for feature subset selection*. Artificial Intelligence, 97(1-2):273-324.

[9] Kononenko, I. (1994) *Estimation attributes: Analysis and extensions of Relief*. In Bergadano, F. and Raedt, L. D., editors, Proceedings of the European Conference on Machine Learning.

[10] Langley, P. (1994), *Selection of relevant features in machine learning*, in AAAI Fall Symposium on Relevance, pp. 140--144.

[11] Murphy, P. M., and Aha, D.W. (1999), *UCI Repository of Machine Learning Databases*, Department of Information and Computer Science, University of California, Irvine, CA, 1999.

[12] Oates, T., and Jensen, D. (1998), *Large datasets lead to overly complex models: an explanation and a solution*. In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98), pp. 294-298, R. Agrawal and Pl Stolorz, Eds., Menlo Park, CA: AAAI Press.

[13] Provost, F., and Kolluri, V. (1999), *A survey of methods for scaling up inductive algorithms*. Data Mining and Knowledge Discovery 2

[14] Provost, F., Jensen, D., and Oates, T. (1999), *Efficient Progressive Sampling*. In Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining (KDD-99), pp. 23-32, San Diego, CA. ACM Press.

[15] Quinlan, J. R. (1993). C4.5: Programs for Machine Leanring. San Mateo, CA: Morgan Kaufmann.