Multivariate Discretization of Continuous Variables for Set Mining

Stephen D. Bay
Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92697
sbay@ics.uci.edu

ABSTRACT
Many algorithms in data mining can be formulated as a set mining problem where the goal is to find conjunctions (or disjunctions) of terms that meet user specified constraints. Set mining techniques have been largely designed for categorical or discrete data where variables can only take on a fixed number of values. However, many data sets also contain continuous variables and a common method of dealing with these is to discretize them by breaking them into ranges. Most discretization methods are univariate and consider only a single feature at a time (sometimes in conjunction with the class variable). We argue that this is a sub-optimal approach for knowledge discovery as univariate discretization can destroy hidden patterns in data. Discretization should consider the effects on all variables in the analysis and that two regions X and Y should only be in the same cell after discretization if the instances in those regions have similar multivariate distributions ($F_x \sim F_y$) across all variables and combinations of variables. We present a bottom up merging algorithm to discretize continuous variables based on this rule. Our experiments indicate that the approach is feasible, that it does not destroy hidden patterns and that it generates meaningful intervals.

Categories and Subject Descriptors
H.2.8 [**Database Management**]: Database Applications—*Data mining*

1. INTRODUCTION
In set mining the goal is find conjunctions (or disjunctions) of terms that meet all user specified constraints. For example, in Association Rule Mining [1] a common step is to find all itemsets that have support greater than a threshold. Set mining is a fundamental operation of data mining. In addition to association rule mining, many other large classes of algorithms can be formulated as set mining such as classification rules [14] where the goal is to find sets of attribute-value (A-V) pairs with high predictive power, or contrast set mining [4, 5] where the goal is to find sets that represent large differences in the probability distributions of two or more groups.

There has been much work devoted to speeding up search in set mining [6, 19] and there are many efficient algorithms when all of the data is discrete or categorical. The problem is that data is not always discrete and is typically a mix of discrete and continuous variables. A central problem for set mining and one that we address in this paper is "How should continuous values be handled?" The most common approach is to discretize them into disjoint regions and then apply the set mining algorithm.

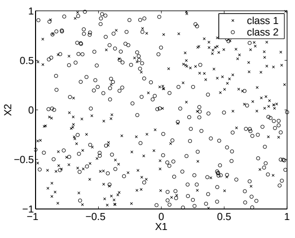
Past work on discretization has usually been done in a classification context where the goal is to maximize predictive accuracy. For example, discretizing continuous attributes for the naive Bayesian classifier can greatly improve accuracy over a normal approximation [8]. In Knowledge Discovery we often analyze the data in an exploratory fashion where the emphasis is not on predictive accuracy but rather on finding *previously unknown and insightful patterns* in the data. Thus we feel that the criteria for choosing intervals should be different from this predictive context as follows:

- *The discretized intervals should not hide patterns.* We must carefully choose our intervals or we may miss potential discoveries. For example, if the intervals are too big we may miss important discoveries that occur at a smaller resolution, but if the intervals are too small we may not have enough data to infer patterns. We refer to this as the *resolution problem.* Additionally, if the intervals are determined by examining features in isolation then with discretization we may destroy interactions that occur between several features.
- *The intervals should be semantically meaningful.* The intervals we choose must make sense to a human expert. For example, when we are analyzing census data we know that it is not appropriate to create intervals such as salary[\$26K,\$80K] because people who make \$26K/year are different qualitatively on many variables such as education, occupation, industry, etc. from people who make \$80K/year.

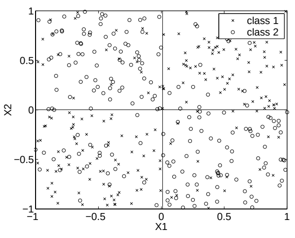
In addition, there is the obvious requirement that the method should be fast enough to handle large databases of interest.

We feel that one method of addressing these points is to

Figure 1: Noisy XOR



(a) XOR



(b) XOR with multivariate discretization

use *multivariate* as opposed to *univariate* discretization. In multivariate discretization one considers how all the variables interact before deciding on discretized intervals. In contrast, univariate approaches only consider a single variable at a time (sometimes in conjunction with the class).

We present a simple motivating example. Consider the problem of set mining on XOR data as in Figure 1a. Clearly one should discretize the data as in Figure 1b which is the result of the method we are proposing in this paper. However algorithms that do not consider more than one feature will fail. For example, Fayyad and Irani's recursive minimum entropy approach [9] will not realize that there is an interaction between X_1, X_2, and the class. It finds both X_1 and X_2 irrelevant and ignores them.

Our approach to this problem is to finely partition each continuous attribute into n basic regions and then to iteratively merge adjacent intervals only when the instances in those intervals have similar distributions. That is, given intervals X and Y we merge them if $F_x \sim F_y$. We use a multivariate test of differences to check this. Merging allows us to deal with the resolution problem and it automatically determines the number of intervals. Our multivariate test means that we will only merge cells with similar distributions so hidden patterns are not destroyed and the regions are coherent.

2. PAST DISCRETIZATION APPROACHES

The literature on discretization is vast but most algorithms are *univariate* in that they consider each feature independently (or only jointly with a class variable) and do not consider interactions with other features. For example, Fayyad and Irani [9] recursively split an attribute to minimize the class entropy. They use a minimum description length criterion to determine when to stop. Dougherty, Kohavi and Sahami [8] provide a good overview of many algorithms in this category. As we mentioned previously, these approaches can miss interactions of several variables.

Srikant and Agrawal [17] proposed an approach that would avoid this limitation. They finely divide each attribute into n basic intervals and then attempt to consider all possible combinations of consecutive basic intervals. However, this creates two problems they refer to as *ExecTime* and *ManyRules*. The ExecTime problem is that since there are $O(n^2)$ combinations of intervals for each feature the complexity will "blow up" especially when we consider the interactions with other features. The ManyRules problem is also related to the number of combinations. If an interval meets the minimum support requirement so does any range containing the interval; e.g., if age[20,30] meets the minimum support constraints then so will age[20,31], age[20,40] and so on. This can result in a huge number of rules.

Miller and Yang [13] note that Srikant and Agrawal's solution may combine ranges that are not meaningful and can result in unintuitive groupings. They present an alternative approach based on clustering the data and then building association rules treating the clusters as frequent itemsets.

3. MULTIVARIATE DIFFERENCE TESTS

A multivariate test of differences takes as input instances drawn from two probability distributions and determines if the distributions are equivalent. In statistical terms the null hypothesis H_0 is that $F_x = F_y$ and the alternate hypothesis is that the two distributions are different $F_x \neq F_y$. In this section, we review past approaches and discuss why they are inappropriate for our application. We argue for a new test based on recent work in contrast set mining [4, 5].

With a single dimension, we can use the Kolmogorov-Smirnov (K-S) two sample test or the Wald-Wolfowitz (W-W) runs test [7] to check for differences. These methods sort the examples and compute statistics based on the ranks of sorted members in the list. For example, the K-S test looks at the maximum absolute difference in the cumulative distribution functions. The Wald-Wolfowitz test uses the total number of runs R, where a run is a set of consecutive instances with identical labels. H_0 is rejected if R is small.

The problem with these methods is that the notion of a sorted list does not apply in multivariate data. Thus in their basic form the K-S and W-W tests are not useful for our problems. However, Friedman and Rafsky [10] generalized the notion of a sorted list by using a minimum spanning tree (MST). They use order information in the MST to calculate multivariate generalizations of K-S and the W-W tests. For the K-S variant, they use a height directed preorder traversal of the tree (visit subtrees in ascending order of their height) to define a total order on nodes in the tree. For the W-W test the multivariate generalization is to remove all edges in the tree that have different labels for the defining nodes and let R be the number of disjoint subtrees.

Unfortunately, the MST based test has a number of disadvantages: First, the generation of the MST requires pairwise distance measures between all instances. In data mining, variables can be both continuous and discrete thus developing a distance metric is not straightforward. Any MST developed will be sensitive to the metric used. Second, the MST is expensive to find. Using Prim's algorithm it is $O(N^2)$ [16] for N data instances. For our data sets N is usually very large thus making the complexity prohibitive. Finally, the above tests were designed to measure significance and have no meaningful interpretation as a measure of the size of the differences between the two distributions. For example, one cannot relate changes in the test statistic (i.e. difference in cumulative distribution function, distribution of runs) to meaningful differences in underlying analysis variables such as age or occupation. Additionally, significance by itself is not sufficient [2] because as $N \to \infty$ all differences, no matter how small, will show up as significant.

We propose using an alternate test of differences based on Contrast Set miners such as STUCCO [4, 5]. STUCCO attempts to find large differences between two probability distributions based on observational data. For example, given census data if we compare PhD and Bachelor's degree holders, STUCCO would return differences between their distributions such as: P(occupation = sales | PhD) = 2.7%, while P(occupation = sales | Bachelor) = 15.8%.

The mining objectives of STUCCO can be stated as follows: Given two groups of instances G_1 and G_2, find all conjunctions of attribute value pairs C (contrast sets) such that:

$$P(C \mid G_1) \neq P(C \mid G_2) \tag{1}$$

$$|\text{support}(C|G_1) - \text{support}(C|G_2)| \geq \delta \tag{2}$$

Support is a frequency measurement and is the percentage of examples where C is true for the given group. Equation 1 is a significance criterion and ensures that the differences we find could not be explained by fluctuations in random sampling. We test this by using a chi-square test which must reject independence of group membership and probability of C. Equation 2 is a size criterion and estimates how big the difference is between two distributions. We require the minimum difference in support to be greater than δ.

STUCCO finds these contrast sets using search. It uses a set enumeration tree [15] to organize the search and it uses many of the techniques in [6, 19] such as dynamic ordering of search operators, candidate groups and support bounds in conjunction with pruning rules geared for finding support differences. STUCCO also carefully controls error caused by multiple hypothesis testing (i.e., false positives).

We use STUCCO as a multivariate test of differences as follows. If STUCCO finds any C that meets both constraints then we say that F_x is substantially different from F_y, otherwise we say that F_x is similar to F_y (i.e. $F_x \sim F_y$).

4. MULTIVARIATE DISCRETIZATION

Given our test from the previous section, we now present our algorithm for MultiVariate Discretization (MVD):

1. Finely partition all continuous attributes into n basic intervals with equal width or frequency discretization.
2. Select two adjacent intervals X and Y that have the minimum combined support and do not have a known discretization boundary between them as candidates for merging.
3. If $F_x \sim F_y$ then merge X and Y into a single interval. Otherwise place a discretization boundary between them.
4. If there are no eligible intervals stop. Otherwise go to 2.

We test if $F_x \sim F_y$ by using STUCCO where the instances that fall in X and Y form the two groups whose distributions we compare. STUCCO requires that we specify δ which represents how big a difference we are willing to tolerate between two distributions. This allows us to control the merging process: small δ means more intervals and large δ means fewer intervals. We set δ adaptively according to the support of X and Y so that any difference between the two cells must be larger than a fixed percentage of the entire dataset. For example, if we tolerate differences of size up to 1% of the entire distribution then we set $\delta = 0.01/\min\{\sup(X), \sup(Y)\}$.

4.1 Efficiency

For each continuous feature, MVD may call STUCCO up to $n-1$ times where n is the number of basic intervals. Each invocation of STUCCO may require an evaluation of an exponential number of candidates (i.e. all combinations of attribute-value pairs) and up to $|A|$ passes through database. This begs the question of how we can implement MVD efficiently when it calls a potentially expensive mining routine.

We believe that it will run efficiently because of the following reasons: First, although the worst case running time is exponential, in practice STUCCO runs efficiently on many datasets [5]. Second, the problems passed off to STUCCO are often easier than that faced by the main mining program. STUCCO only needs to consider the examples that fall into the two ranges that are being tested for merging. This can be only a small fraction of the data set. Third, STUCCO only needs to find a single difference between the groups and then it can exit. It does not need to find all differences. Finally, calling STUCCO repeatedly will result in many passes over the database. However the limiting factor for mining algorithms is the exponential number of candidates that need to be considered, not passes through database (e.g. [6]).

5. SENSITIVITY TO HIDDEN PATTERNS

We test the ability of MVD to properly discretize data with hidden patterns in high dimensional data by defining a problem called Parity R+I. This is a continuous version of the parity problem where there are R continuous variables ranging from [-0.5,0.5] with a uniform distribution, one continuous irrelevant variable also ranging from [-0.5,0.5], and a boolean class variable. If an even number of the first R features are positive then the class variable is 1; 0 otherwise. We then added 25% class noise and generated 10000 examples from this distribution.

We used MVD with equal frequency partitioning (100 examples per division) on the Parity 5+I problem. This problem is difficult because there is a hidden 6 dimensional relationship and with our initial partitioning of features we have only 10000 instances to be divided into $100^6 \times 2$ possible cells. We ran five trials and Table 1 shows the cutpoints we found for each feature (MVD found at most 1 cutpoint per feature). The true solution is [0,0,0,0,0,ignore]. MVD did very well at identifying the relationship between F1,...,F5 and the class. Although it did not exactly reproduce the desired cutpoints, it came reasonably close, and a set miner should still be able to identify the parity relationship. MVD failed only once out of the five trials and it always managed to identify the irrelevant variable. In contrast, univariate discretizers will only be able to solve Parity 1+I problems.

Table 1: MVD Cutpoints for Parity 5+I

	Feature					
Trial	F1	F2	F3	F4	F5	F6
1	0.08	0.03	0.15	-0.08	0.01	ignore
2	-0.12	-0.10	-0.15	0.19	-0.02	ignore
3	-0.13	0.20	0.00	0.06	-0.10	ignore
4	-0.18	ignore	ignore	ignore	ignore	ignore
5	0.02	0.06	-0.15	0.16	0.08	ignore

6. EVALUATION

In this section, we show that our approach is efficient and can be used in set mining without a large overhead on the entire process. We also demonstrate by example that our approach generates meaningful results.

For our experiments we compared MVD with Fayyad and Irani's recursive minimum entropy approach with the MDL stopping criteria. We refer to this as ME-MDL and we used the MLC++ [12] implementation of this discretizer. Past work has shown that ME-MDL is one of the best methods for classification [8, 11]. We also compared our execution times with Apriori to give an indication of how much time discretization takes relative to the set-mining process. We used C. Borgelt's implementation of Apriori (in C).[1]

We ran experiments on five databases which are summarized in Table 2. The first four databases are publically available from the UCI KDD Archive [3]. The UCI Admissions dataset represents all undergraduate student applications to UCI for the years 1993-1999. The data contains variables such as ethnicity, school (e.g. Arts, Engineering, etc.), if an offer of admission was made, GPA, SAT, etc.

We ran all experiments on a Sun Ultra-5 with 128 MB of RAM. We used the following parameter settings: The basic intervals were set with equal frequency partitioning with 100 instances per interval for Adult, SatImage, and Shuttle, 2000 per interval for UCI Admissions, and 10000 per interval for Census-Income. We required differences between adjacent cells to be at least 1% of N. ME-MDL requires a class variable and for Adult, Census-Income, SatImage, and Shuttle we used the class variable that had been used in previous analyses. For UCI Admissions we used Admit = {yes,no} (i.e., was the student admitted to UCI).

Table 2: Description of Data Sets

Data Set	# Features	# Continuous	# Examples
Adult	14	5	48812
Census-Income	41	7	199523
SatImage	37	36	6435
Shuttle	10	9	48480
UCI Admissions	19	8	123028

6.1 Execution Time

Table 3 shows the discretization time for MVD, ME-MDL and the time taken by Apriori to perform frequent set mining on MVD's discretizations. MVD's time was generally comparable to ME-MDL. Both discretization processes usually took longer than than Apriori but they were not excessively slow. Census-Income was exceptionally difficult for Apriori which ran out of memory and could not mine frequent itemsets at 10% support. We tried mining with 30% support but even at this level Apriori could not complete mining in reasonable time and we stopped it after 10 CPU hours.

Table 3: Discretization Time in CPU seconds

Data Set	MVD	ME-MDL	Ap. (10%)	Ap. (5%)
Adult	65	541	44	104
Census-Income	11065	8142	-	-
SatImage	127	36	0	4
Shuttle	77	318	1	2
UCI Admissions	370	772	131	204

6.2 Qualitative Results

We believe that our approach of combining ranges only when they have similar distributions will lead to semantically meaningful intervals. We demonstrate this by comparing the intervals formed by MVD with ME-MDL.

For several variables MVD and ME-MDL obtained similar discretizations. For example, Figures 2a and 2b show the discretization boundaries for the age variable in the Adult dataset superimposed on top of it's histogram. The intervals for both MVD and ME-MDL are narrow at younger age ranges and wider at older ranges. This makes intuitive sense as the data represents many employment related variables and people's careers tend to change less as they age.

However, MVD and ME-MDL differed significantly on many other variables. Figures 3a and 3b show the discretizations boundaries found for parental income on UCI Admissions data. Note that we plotted the logarithm of the income for visualization purposes only; we did not transform the variables before discretization. The MVD cutpoints occur at meaningful locations: {\$17000, \$30000, \$51760, \$75000}. We can easily relate these to our notions of poverty and wealth. The ME-MDL discretization is not meaningful. Consider that it grouped everybody with parental income between \$36K and \$200K together. Additionally, ME-MDL had many cutpoints distinguishing applicants at the upper range of parental income (i.e. over \$400K).

Another problem with using ME-MDL in a discovery context is that it requires a class variable and *the discretization is sensitive to this.* While being sensitive to the class variable is probably good in a classification context, it is not for discovery. Stable results are essential for domain users to accept the discovered knowledge [18]. We discretized UCI Admissions data with ME-MDL using sex and year as alternate class variables and we found wildly different cutpoints. Using sex produced income cutpoints at {\$55K, \$162K, \$390K} and using year produced cutpoints at {\$13K, \$95K}.

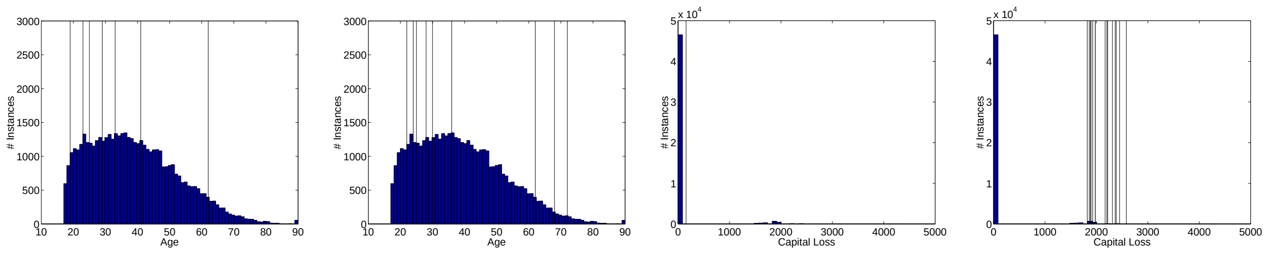
Figures 2c and 2d show the discretization cutpoints found for Capital-Loss on Adult. In this case, most data analysts would probably agree with MVD as opposed to ME-MDL. We ran Apriori using both MVD and ME-MDL's boundaries for capital-loss (using MVD's discretization for all other variables) and found that the poor cutpoints chosen by ME-MDL can hide important association rules. For example, with MVD's cutpoint we were able to find rules such as capital-loss $\geq$ \$155 $\rightarrow$ salary $>$ \$50K (support 2.3%, confidence 50.1%). This rule states that declaring a capital-loss is strongly predictive of having a high salary. The base rate for salary $>$ \$50K is 24% so the rule identifies a group with twice this probability. We did not find any similar rules with ME-MDL's discretization because it used too many partitions making it difficult for Apriori to infer relationships.

Income and Capital-Loss are very skewed variables, however, ME-MDL can also produce unintuitive boundary points for more normal data. Figures 3c and 3d show the discretization boundaries found for GPA on UCI Admissions.

These results suggest that perhaps ME-MDL is not appropriate for knowledge discovery when the goal is understanding the data. ME-MDL can generate too many intervals that are close together and are at odd locations. In MVD our similarity test prevents this from happening because it

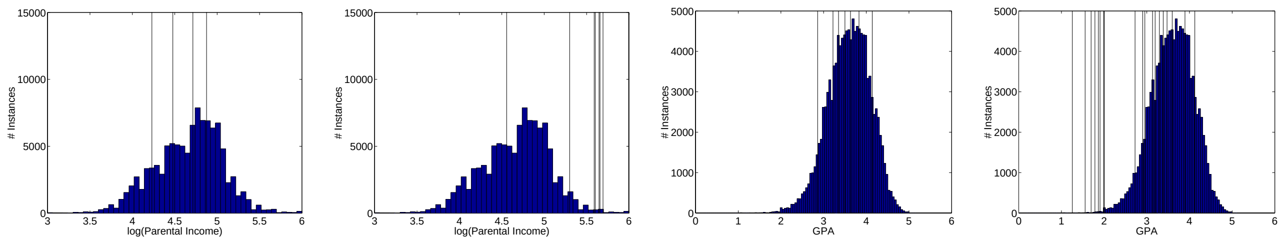
[1]available from http://fuzzy.cs.Uni-Magdeburg.de/~borgelt/

Figure 2: Discretization Cutpoints for Age and Capital-Loss on the Adult Census Data



(a) MVD (b) ME-MDL (c) MVD (d) ME-MDL

Figure 3: Discretization Cutpoints for Parental Income and GPA on UCI Admissions Data



(a) MVD (b) ME-MDL (c) MVD (d) ME-MDL

requires that adjacent intervals not only be different but also be different by a certain minimum size.

7. CONCLUSIONS

We presented a multivariate discretization algorithm that finely partitions continuous variables and then merges adjacent intervals only if their instances have similar multivariate distributions. Merging allows us to automatically determine an appropriate resolution to quantize the data. Our multivariate test ensures that only similar distributions are joined. Our experimental results on synthetic data indicate that our algorithm can detect high dimensional interactions between features and discretize the data appropriately. On real data our algorithm ran in time comparable to a popular univariate recursive approach and produced sensible discretization cutpoints.

8. ACKNOWLEDGMENTS

This research was funded in part by the National Science Foundation grant IRI-9713990.

9. REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining associations between sets of items in massive databases. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 207–216, 1993.

[2] D. Bakan. The test of significance in psychological research. *Psychological Bulletin*, 66(6), 1966.

[3] S. D. Bay. The UCI KDD archive. Irvine, CA: University of California, Information and Computer Science, 1999. [http://kdd.ics.uci.edu/].

[4] S. D. Bay and M. J. Pazzani. Detecting change in categorical data: Mining contrast sets. In *Proc. of the 5th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 302–306, 1999.

[5] S. D. Bay and M. J. Pazzani. Detecting group differences: Mining contrast sets, under review.

[6] R. J. Bayardo. Efficiently mining long patterns from databases. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, 1998.

[7] W. J. Conover. *Practical Nonparametric Statistics*. John Wiley & Sons, first edition, 1971.

[8] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *Proc. 12th Int. Conf. on Machine Learning*, 1995.

[9] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proc. of the 13th Int. Joint Conf. on Artificial Intelligence*, 1993.

[10] J. H. Friedman and L. C. Rafsky. Multivariate generalizations of the Wald-Wolfowitz and Smirnov two-sample tests. *Annals of Statistics*, 7(4), 1979.

[11] R. Kohavi and M. Sahami. Error-based and entropy-based discretization of continuous features. In *Proc. of the 2nd Int. Conf. on Knowledge Discovery and Data Mining*, pages 114–119, 1996.

[12] R. Kohavi, D. Sommerfield, and J. Dougherty. Data mining using MLC++, a machine learning library in C++. *Journal of Artificial Intelligence Tools*, 6(4):537–566, 1997.

[13] R. J. Miller and Y. Yang. Association rules over interval data. In *Proc. of the 1997 ACM SIGMOD Int. Conf. on Management of Data*, 1997.

[14] J. R. Quinlan. *C4.5 programs for machine learning.* Morgan Kaufmann, 1993.

[15] R. Rymon. Search through systematic set enumeration. In *Third Int. Conf. on Principles of Knowledge Representation and Reasoning*, 1992.

[16] R. Sedgewick. *Algorithms in C.* Addison-Wesley, 1990.

[17] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proc. ACM SIGMOD Conf. on Management of Data*, 1996.

[18] P. Turney. Technical note: Bias and the quantification of stability. *Machine Learning*, 20:23–33, 1995.

[19] G. I. Webb. OPUS: An efficient admissible algorithm for unordered search. *Journal of Artificial Intelligence Research*, 3:431–465, 1995.