Algorithms for Feature Selection in Rank-Order Spaces

Douglas J. Slotta[*] John Paul C. Vergara [†]
Naren Ramakrishnan[*] Lenwood S. Heath[*]

April 11, 2005

Abstract

The problem of feature selection in supervised learning situations is considered, where all features are drawn from a common domain and are best interpreted via ordinal comparisons with other features, rather than as numerical values. In particular, each instance is a member of a space of ranked features. This problem is pertinent in electoral, financial, and bioinformatics contexts, where features denote assessments in terms of counts, ratings, or rankings. Four algorithms for feature selection in such rank-order spaces are presented; two are information-theoretic, and two are order-theoretic. These algorithms are empirically evaluated against both synthetic and real world datasets. The main results of this paper are (i) characterization of relationships and equivalences between different feature selection strategies with respect to the spaces in which they operate, and the distributions they seek to approximate; (ii) identification of computationally simple and efficient strategies that perform surprisingly well; and (iii) a feasibility study of order-theoretic feature selection for large scale datasets.

1 Introduction

A number of important economic, financial, and scientific domains present situations where features are best interpreted via ordinal comparisons with other features, rather than as absolute values. This is especially the case when the features represent counts, ratings, rankings, or otherwise dimensionless quantities.

For instance, consider a financial analyst interested in assessing the outlook of the stock market (bear-vs-bull) by analyzing the proceedings of a trading day. Instances are hence trading days and the (binary) outlook forecasts constitute the classes. Each instance is described by, say, the average percentage change

[*]Department of Computer Science, Virginia Polytechnic Institute and State University
[†]Department of Information Systems and Computer Science, Ateneo de Manila University

experienced by stocks in various market indices, e.g., HANG SENG, NIKKEI, S&P, and DOW. For instance, a given trading day would be qualified using the features HANG SENG $\uparrow$ 2.5%, NIKKEI $\uparrow$ 4.8%, S&P $\downarrow$ 2.8%, DOW $\uparrow$ 1.3%. However, it is not meaningful to work directly with the percentage changes in an attribute-value sense; instead we rank the features and henceforth think of a trading day in terms of an order (total or partial) over the features. The above example would then be represented as: NIKKEI > HANG SENG > DOW > S&P. The goal of supervised learning in this setting is to infer a mapping from such orders to given classifications. For instance, we might learn that 'if DOW is ranked higher than S&P but ranked lower than NIKKEI, then the outlook is bullish.' Feature selection in such rank-order datasets is important for the same reasons it is in regular feature spaces, namely, reducing the complexity of induction, removing irrelevant information from a dataset, and improving prediction performance. For instance, the relative ordering of the HANG SENG index among the other indices might not be informative toward the goal of outlook prediction, and hence the feature can be safely eliminated.

There are many applications that highlight the importance of feature selection in rank-order datasets. In biomedical instrumentation, the desire is to select a subset of electrodes from an EEG dataset and use profiles of relative signal strength as indicators of patient health [29]. Here the instances are the patients, the classes are the diagnoses, and the features denote signal strength as measured using different electrodes. In large-scale gene expression assays [1, 27], one possible aim is to classify an experimental condition using expression changes only across a 'salient' subset of genes. For instance, by observing a handful of genes (features) and ranking them by their expression levels, it is possible to qualitatively characterize the cellular transcriptional state (class) for a given condition (instance). In decision-making referendums, one goal is to identify key voting indicators to infer political biases of constituencies. Here, the instances are the constituencies, the classes denote political party strongholds, and the features could be socio-economic indicators. On a lighter vein, it appears possible to classify a movie as an art film or a mass market flick by ranking critics! Given such a widespread prevalence of applications where rank order is pertinent [21], it is surprising that this feature selection problem has received little attention.

The formulation is made precise in Section 2, but, informally, we posit a setting where the features can be ordered within an instance (and thus, must come from the same domain). This is starkly different from the more traditional settings where the instances or classes are ordered, or where the feature values are ranked *across instances*. These other formulations are pertinent in applications such as recommender systems or information retrieval [4, 11], where the goal is to learn and mine a ranking or to infer total orders from given preference information. In this paper, we assume the existence of a supervised learning algorithm that learns predictive mappings from orders to classes and the focus is on feature selection as a preprocessing step to such an algorithm.

The specific focus is on identifying a subset of ordinal features (i.e., projections of the given set of features) that exhibit sufficient relationships to model feature-conditional class distributions for use in supervised learning. The result is a 'best subset' of ordinal features, not a 'best ordering' of the features; aggregating orderings by consolidating given ranking information is the purview of subjects such as social choice theory [24] (an area rich in impossibility results [2]).

In the remaining sections, some definitions are provided, four algorithms for feature selection are introduced, and theoretical intuitions as well as experimental results on synthetic and real datasets are presented.

2 Definitions

Definition 1. *Let $\mathbf{F} = \{F_1, F_2, \ldots, F_n\}$ be a set of* features, *and let D_i be the domain of feature F_i. Let $\mathbf{D} = D_1 \times D_2 \times \cdots \times D_n$ be the cartesian product of the feature domains. A* feature instance *is a tuple $\mathbf{f} = (f_1, f_2, \ldots, f_n) \in \mathbf{D}$. Let $C = \{C_1, C_2, \ldots, C_n\}$ be a set of* classes. *A* dataset *T is a nonempty multiset of pairs $(\mathbf{f}, c)$, where $\mathbf{f} \in \mathbf{D}$ and $c \in C$. Let $|T|$ be the multiset cardinality of T. Each dataset T implies a probability distribution P on $\mathbf{D} \times C$ as follows:*

$$P(\mathbf{f}, c) \quad = \quad \frac{|\{(\mathbf{f}, c) \in T\}|}{|T|}.$$

Furthermore, if feature instance $\mathbf{f}$ occurs in at least one pair of T, then the conditional probability

$$P(c \mid \mathbf{f}) \quad = \quad \frac{P(\mathbf{f}, c)}{P(\mathbf{f})}$$

is defined, for all $c \in C$. When we want to refer to the conditional probability distribution over C, we shall use the term $P(C \mid \mathbf{f})$.

Numerous criteria can be applied in a reduction of $\mathbf{F}$ to a subset $\mathbf{F}' \subset \mathbf{F}$, including improved accuracy of predictive modeling, smaller description length for learned mappings, or preservation of as much of the relationship between class distributions and features as possible. This latter criterion is the motivation for the classical work of Koller and Sahami [15]. For $\mathbf{f} \in \mathbf{F}$, let $\mathbf{f}_{\mathbf{F}'}$ be the projection of $\mathbf{f}$ onto the features in $\mathbf{F}'$. For a dataset T, let $T_{\mathbf{F}'} = \{\mathbf{f}_{\mathbf{F}'} \mid \mathbf{f} \in \mathbf{F}\}$ be the projection of T using the $\mathbf{F}'$ feature set. The goal advanced by Koller and Sahami is to approximate $P(C \mid \mathbf{f})$ with $P(C \mid \mathbf{f}_{\mathbf{F}'})$. A popular approach to characterizing the difference between two distributions is the (non-symmetric) KL-divergence [5].

Definition 2. *Let P and Q be probability distributions on the sample space X. The KL-divergence between P and Q is $KL(P, Q) = \sum_{x \in X} P(x) \log \frac{P(x)}{Q(x)}$, where $0 \log \frac{0}{Q(x)} = 0$ and $P(x) \log \frac{P(x)}{0} = \infty$ whenever $P(x) > 0$.*

Typically, P is the true distribution, and Q the approximation; $KL(P, Q)$ captures the number of extra bits required if we encoded data arising from the distribution P using a code designed using distribution Q. Koller and Sahami use KL-divergence to define two feature subset divergence quantities, $\delta_{\mathbf{F}'}$ and $\Delta_{\mathbf{F}'}$.

Definition 3. *(Koller and Sahami) Let $\mathbf{F}' \subset \mathbf{F}$. Let $T \subset \mathbf{D} \times C$ be a dataset. For each feature instance $\mathbf{f}$ in T, define its divergence to be*

$$\delta_{\mathbf{F}'}(\mathbf{f}) \quad = \quad KL(P(C \mid \mathbf{f}), P(C \mid \mathbf{f}_{\mathbf{F}'})).$$

Define the feature subset divergence *of $\mathbf{F}'$ to be*

$$\Delta_{\mathbf{F}'} \quad = \quad \sum_{\mathbf{f}} P(\mathbf{f}) \delta_{\mathbf{F}'}(\mathbf{f}),$$

where $P(\mathbf{f})$ is taken from the distribution of feature instances in T.

Two ways to utilize the feature subset divergence are (1) to define a divergence threshold and seek a smallest subset $\mathbf{F}'$ such that $\Delta_{\mathbf{F}'}$ is at most that threshold; and (2) to seek among all $\mathbf{F}' \subset \mathbf{F}$ of a fixed size one that minimizes $\Delta_{\mathbf{F}'}$.

For simplicity in incorporating orders into feature space, we assume that all feature domains are identical and that a total order is defined on that single domain. Any feature instance $\mathbf{f} = (f_1, f_2, \ldots, f_n)$ defines an order on the features in F by the rule $F_i < F_j$ if $f_i < f_j$. In this case, we can recast the given dataset into one whose features are boolean values that capture the relative order between pairs of feature values of an instance.

Definition 4. *Let $\mathbf{F} = \{F_1, F_2, \ldots, F_n\}$ be a feature set, where D is the common domain of F_i. Define the* boolean order feature set *for $\mathbf{F}$ to be*

$$\mathbf{B} \quad = \quad \{B_{i,j} \mid 1 \leq i < j \leq n\},$$

where the domain of each $B_{i,j}$ is $\{true, false\}$. Let $\mathbf{f} = (f_1, f_2, \ldots, f_n)$ be a feature instance. Then the boolean order features for $\mathbf{f}$ have values given by

$$b_{i,j} \quad = \quad \begin{cases} \textit{true} & \textit{if } f_i < f_j; \\ \textit{false} & \textit{otherwise.} \end{cases}$$

There are $\binom{n}{2}$ boolean order features. The vector comprised of the boolean order feature values $b_{i,j}$ of $\mathbf{f}$ is referred to as $\mathbf{b}$. If $T \subset \mathbf{D} \times C$ is a dataset, then the corresponding boolean order dataset *is the multiset $T^B = \{(\mathbf{b}, c) \mid (\mathbf{f}, c) \in T\}$.*

Boolean order datasets suffer greater space complexity than traditional datasets due to the $\binom{n}{2}$ boolean order features. This drawback can be ameliorated with the use of ranks.

Definition 5. *Let $\mathbf{F} = \{F_1, F_2, \ldots, F_n\}$ be a feature set, where D is the common domain of F_i. Define the* rank-order feature set *for $\mathbf{F}$ to be*

$$\mathbf{R} \quad = \quad \{R_1, R_2, \ldots, R_n\},$$

where the domain of each R_i is $\{1, 2, \ldots, n\}$. Let $\mathbf{f} = (f_1, f_2, \ldots, f_n)$ be a feature instance, and let $\pi_\mathbf{f} : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$ be a permutation that sorts $\mathbf{f}$ into non-decreasing order. (If two features have the same value, then choose $\pi_\mathbf{f}$ arbitrarily from the permutations that satisfy the above condition.) Then the rank-order features for $\mathbf{f}$ have values given by

$$r_i \quad = \quad \pi_\mathbf{f}(i).$$

The vector comprised of the rank-order feature values r_i of $\mathbf{f}$ is referred to as $\mathbf{r}$ (note that $\mathbf{r}$ is also a permutation of $\{1, 2, \ldots, n\}$). *If $T \subset \mathbf{D} \times C$ is a dataset, then the corresponding* rank dataset *is the multiset $T^R = \{(\mathbf{r}, c) \mid (\mathbf{f}, c) \in T\}$.*

Notice that the transformation to boolean order features provided in Definition 4 applies even if we begin with a rank-order dataset. Figure 1 illustrates Definitions 1, 4, and 5. Difficulties arise in Definitions 4 and 5 if any feature instance contains two identical feature values, resulting in rank ambiguity. However, rank ambiguity can be addressed in boolean order datasets by setting the domain to $\{\text{true}, \text{false}, =\}$, instead of just $\{\text{true}, \text{false}\}$. In fact, setting the domain to $\{\text{true}, \text{false}, =, \text{unknown}\}$, allows us to address partial orders in general. For rank-ordered datasets, there is a corresponding method to address rank ambiguity; addressing partial orders, however, is not straightforward in this context.

For simplicity, we assume total orders and that rank ambiguity does not occur. However, with additional labor, it is possible to relax the rank-ambiguity restriction and apply most of the statements and conclusions in this paper in the relaxed case; exceptions will be noted when appropriate. Since there is no rank ambiguity in $\mathbf{f}$, the permutation $\pi_\mathbf{f}$ is uniquely defined; call it the *feature order* of $\mathbf{f}$.

The update required if a feature is removed in the context of boolean order or rank-order datasets is more complicated than in the unranked context, as the projection onto a smaller feature space requires additional effort. If feature F_i is removed in a boolean order context where there are n original features and $\binom{n}{2}$ boolean order features, then there are $n-1$ boolean order features of the form $B_{i,j}$ or $B_{j,i}$ that must be eliminated. If feature F_i is removed in a rank-order context where there are n original features, then rank-order feature R_i must be eliminated, and the remaining $n-1$ rank-order features must be re-indexed and their rank values updated, For example, projecting the rank-order feature instance $\mathbf{r} = (3, 1, 4, 2)$ onto the first three (rank) features results in the rank-order feature instance $(2, 1, 3)$, not $(3, 1, 4)$.

Let $T \subset \mathbf{D} \times C$ be a dataset, and let T^R be the corresponding rank-order dataset. Let $(\mathbf{f}, c) \in T$ be a feature instance, and let $(\mathbf{r}, c) \in T^R$ be its corresponding rank-order feature instance. It is possible that $P(C \mid \mathbf{f}) \neq P(C \mid \mathbf{r})$. This is because information (specifically, information that is not order-specific) is lost in the transformation from T to T^R. On the other hand, order-specific information is now available in T^R, presumably not considered in the original dataset. We provide some results illustrating this point in Section 3.

F_1	F_2	F_3	F_4	class
20	40	65	33	a
20	40	65	33	a
50	25	55	99	b
88	76	10	60	a

$B_{1,2}$	$B_{1,3}$	$B_{1,4}$	$B_{2,3}$	$B_{2,4}$	$B_{3,4}$	class
true	true	true	true	false	false	a
true	true	true	true	false	false	a
false	true	true	true	true	true	b
false	false	false	false	false	true	a

R_1	R_2	R_3	R_4	class
1	3	4	2	a
1	3	4	2	a
2	1	3	4	b
4	3	1	2	a

Figure 1: Dataset T (top table) with its corresponding boolean order dataset T^B and rank-order dataset T^R (bottom tables). Note that T is a *multiset,* due to the first and second feature instances.

Since, in the context of a rank-order dataset, an instance (as well as its implied feature order) is a permutation, we require methods to measure differences between permutations. We present two established approaches for defining a distance function between two permutations, one that works with (rank) instances, and another that works with feature orders. The first distance function is Spearman's distance [26].

Definition 6.

$$sd(\pi_i, \pi_j) = \sum_{k=1}^{n} (\pi_i(k) - \pi_j(k))^2$$

Let P be a set of permutations. The center $\mathrm{ctr}(P)$ *of P is a permutation π_c (not necessarily in P) that minimizes*

$$\sum_{\pi \in P} (sd(\pi, \pi_c)).$$

Algorithmically, we can compute the permutation $\mathrm{ctr}(P)$ by summing the ranks in each position, across all permutations, and deriving $\mathrm{ctr}(P)$ from the order of the resulting sums (if there are duplicate sums, ties are broken arbitrarily).

$$\begin{array}{lll}
\pi_i = & F_3\ F_2\ F_4\ F_5\ F_1\ F_6 & \\
 & F_3\ F_2\ F_5\ F_4\ F_1\ F_6 & \text{swap}(F_4,F_5) \\
 & F_3\ F_5\ F_2\ F_4\ F_1\ F_6 & \text{swap}(F_2,F_5) \\
 & F_3\ F_5\ F_2\ F_1\ F_4\ F_6 & \text{swap}(F_4,F_1) \\
\pi_j = & F_3\ F_5\ F_2\ F_1\ F_4\ F_6 &
\end{array}$$

Figure 2: Three interchanges are required to transform π_i to π_j.

In the context of feature selection, every class C_j for which $P_j = \{\mathbf{r} \mid (\mathbf{r}, C_j) \in T^R\}$ is nonempty, $\text{ctr}(P_j)$ is a permutation at the center of a smallest hypersphere containing all the permutations of P_j. If C_i and C_j are distinct classes, then $\text{ctr}(P_i)$ and $\text{ctr}(P_j)$ are representatives of the two classes that can be used to define a distance between the two classes. A feature removal algorithm might choose to remove a feature that yields distances that are closest to the original distances.

A second distance function between permutations is based on swaps.

Definition 7. *A permutation* $\pi : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$ *is an* adjacent swap *if there exists an* i*, where* $1 \le i \le n-1$*, such that*

$$\pi(j) = \begin{cases} j+1 & \text{if } j = i; \\ j-1 & \text{if } j = i+1; \\ j & \text{otherwise.} \end{cases}$$

The function $kd(\pi, \pi')$ *is the minimum number of adjacent swaps required to transform* π *to* π'*.*

Figure 2 provides an example transformation of $\pi_i = F_3\ F_2\ F_4\ F_5\ F_1\ F_6$ to $\pi_j = F_3\ F_5\ F_2\ F_1\ F_4\ F_6$ in three adjacent swaps, showing the three intermediate permutations. The minimum number of swaps of adjacent pairs is exactly the same as Knuth's inversion count [14]. If $\pi_i(x) < \pi_i(y)$ and $\pi_j(x) > \pi_j(y)$ then the pair (x, y) is an inversion, where $\pi_i(x)$ is the x'th element of permutation π_i.

Instead of using centers based on Kendall's distance (the decision problem for determining such centers is in fact NP-complete [8]), we propose a different notion for feature removal that uses inversions. Consider the feature orders of two instances in rank-order space. In a transformation from one permutation to the other using inversions, each inversion involves two features. Let $\text{inv}(R_i, \pi_a, \pi_b)$ represent the number of inversions that a feature R_i is involved in, when going from π_a to π_b (for the example in Figure 2, $\text{inv}(F_2, \pi_i, \pi_j) = 1$ while $\text{inv}(F_5, \pi_i, \pi_j) = 2$).

Definition 8. *(Spoiler Count of a feature for a set of orders) Let* I *be a multiset of feature orders. The* spoiler count $\text{sp}(R_i, I)$ *of feature* R_i *with respect to* I *is*

$$\text{sp}(R_i, I) = \sum_{\pi_a, \pi_b \in I} \text{inv}(R_i, \pi_a, \pi_b).$$

Suppose I is the multiset of feature orders of instances associated with a single class in T^R. Since the instances belong to the same class and order is presumably indicative of class membership, these permutations ought to be similar in order. The feature that has the highest spoiler count contributes most to the differences in order and is therefore a good candidate for removal.

Definition 9. *(Spoiler Count of a feature for a dataset) For each class $c \in C$, let $T^R(c)$ represent the set of feature orders of instances associated to c in T^R. The* total spoiler count $\mathrm{tsp}(R_i)$ *of rank-order feature R_i is*

$$\mathrm{tsp}(R_i) = \sum_{c \in C} \mathrm{sp}(R_i, T^R(c)).$$

A feature selection algorithm might choose to remove the feature that has the highest total spoiler count.

3 Comparing Feature Spaces

In this section, we investigate relationships among feature spaces, boolean order spaces, and rank-order spaces. We first identify examples where order matters more than value and vice-versa.

Let F_k be a feature in $\mathbf{F}$, R_k the corresponding feature in $\mathbf{R}$, $\mathbf{F}' = \mathbf{F} - \{F_k\}$, and $\mathbf{R}' = \mathbf{R} - \{R_k\}$. We say that $\mathbf{F}_k$ is a *removable* feature if $\Delta_{\mathbf{F}'} = 0$. We now examine the relationship between $\Delta_{\mathbf{F}'}$ and $\Delta_{\mathbf{R}'}$. The next two conjectures and their counterexamples illustrate that the relationship is not simple or easily exploitable.

Conjecture 1. *Let T be a dataset with feature set $\mathbf{F}$ and suppose F_k is a feature such that $\Delta_{\mathbf{F}-\{F_k\}} = 0$. Then, $\Delta_{\mathbf{R}-\{R_k\}} = 0$ in T^R,*

Counterexample: Figure 3 provides an example of a dataset T and corresponding rank-order dataset T^R. Here, F_1 is a removable feature in dataset T (values assigned to F_1 are the same for all instances). This means $\Delta_{\mathbf{F}-\{F_k\}} = 0$. However, for $\Delta_{\mathbf{R}-\{R_1\}} \neq 0$ in T^R, because there is at least one instance $\mathbf{r}$ (such as $\mathbf{r} = (3, 1, 2)$) such that $\delta_{\mathbf{R}'}(\mathbf{r}) \neq 0$.

Conjecture 2. *Let T be a dataset with feature set $\mathbf{F}$, and let T^R with feature set $\mathbf{R}$ be its corresponding rank-order dataset. Suppose R_k is a feature such that $\Delta_{\mathbf{R}'} = 0$. Then, $\Delta_{\mathbf{F}'} = 0$ in T.*

Counterexample: Figure 4 provides an example of a dataset T and corresponding rank-order dataset T^R. Here, R_1 can be removed (as can any single feature) while retaining the same capacity to classify so that $\Delta_{\mathbf{R}'} = 0$ in T^R, where $\mathbf{R}' = \mathbf{R} - \{R_1\}$. However, $\Delta_{\mathbf{F}'} \neq 0$ for $\mathbf{F}' = \mathbf{F} - \{F_1\}$ since F_1 is in fact the feature that distinguishes the two instances in the dataset ($\delta_{\mathbf{F}'}(\mathbf{f}) \neq 0$ for both instances).

F_1	F_2	F_3	class
3	4	5	a
3	4	1	b
3	1	2	c
3	2	5	d

R_1	R_2	R_3	class
1	2	3	a
2	3	1	b
3	1	2	c
2	1	3	d

Figure 3: Datasets T (top) and T^R (bottom) where a removable feature in T does not apply in T^R.

F_1	F_2	F_3	class
1	4	5	a
2	4	5	b

R_1	R_2	R_3	class
1	2	3	a
1	2	3	b

Figure 4: Datasets T (top) and T^R (bottom) where a removable feature in T^R does not apply in T.

On the other hand, the following result demonstrates that rank-order datasets and boolean order datasets contain the same order-theoretic information with respect to feature selection.

Lemma 1. *Let T^R be a rank-order dataset with feature set $\mathbf{R}$, and let T^B with feature set $\mathbf{B}$ be its corresponding boolean order dataset. Furthermore, let $\mathbf{R}' \subset \mathbf{R}$. Define $\mathbf{B}' \subset \mathbf{B}$ to be the set of all features $B_{i,j}$ such that $R_i, R_j \in \mathbf{R}'$. Then, $\Delta_{\mathbf{R}'} = \Delta_{\mathbf{B}'}$.*

Proof. From Definition 4, it suffices to show that $P(\mathbf{r}) = P(\mathbf{b})$ and $\delta_{\mathbf{R}'}(\mathbf{r}) = \delta_{\mathbf{B}'}(\mathbf{b})$, for all instances $\mathbf{r}$. The equality $P(\mathbf{r}) = P(\mathbf{b})$ follows directly from how the boolean order set was constructed since there is a one-to-one correspondence between rank-order instances ($\mathbf{r}$) and boolean order instances ($\mathbf{b}$). We obtain that

$$\delta_{\mathbf{R}'}(\mathbf{r}) = KL(P(C \mid \mathbf{r}), P(C \mid \mathbf{r}_{\mathbf{R}'})),$$

and

$$\delta_{\mathbf{B}'}(\mathbf{b}) = KL(P(C \mid \mathbf{b}), P(C \mid \mathbf{b}_{\mathbf{B}'}))$$

are equal by the observation that the projections performed on each of the datasets are essentially equivalent. For a given $\mathbf{r}$, $P(C \mid \mathbf{r})$ and $P(C \mid \mathbf{b})$ obviously yield the same distributions, again because of the one-to-one transformation. For the distributions $P(C \mid \mathbf{r}_{\mathbf{R}'})$ and $P(C \mid \mathbf{b}_{\mathbf{B}'})$, on the other hand, we note that a projection in rank-order space preserves the relative order of the features even with the (possible) update in rank values. This in turn corresponds to the boolean order features that are projected in boolean order space. Thus, $P(C \mid \mathbf{r}_{\mathbf{R}'}) = P(C \mid \mathbf{b}_{\mathbf{B}'})$, and the result follows. $\square$

Lemma 1 suggests that it is sufficient to consider selection strategies on rank-order datasets and that analogous strategies using boolean order datasets will yield the same results.

4 Feature Selection Strategies

We present four feature selection strategies (two taking an information-theoretic approach and two motivated by the discrete mathematics concepts introduced in Section 2), all of which follow the standard backward stepwise selection framework [10].

```
$\mathbf{F}_0 \leftarrow \mathbf{F}$; $i \leftarrow 0$
**while** $cond(\mathbf{F}_i)$
    $F_k \leftarrow h(\mathbf{F}_i)$
    $\mathbf{F}_{i+1} \leftarrow \mathbf{F}_i - \{F_k\}$
    $i \leftarrow i + 1$
**end while**
return $\mathbf{F}_i$
```

In this meta-algorithm, the boolean function $cond(\mathbf{F}_i)$ either monitors subset size or subset divergence. The function $h(\mathbf{F}_i)$ is the feature selection function for this selection strategy. This function returns a feature from $\mathbf{F}_i$ in regular feature space but uses rank order space in its selection process. Recall that $\mathbf{F}$, F_k, and $\mathbf{F}_i$ correspond to $\mathbf{R}$, R_k, and $\mathbf{R}_i$ in rank-order space, as these terms may be used in the definition of $h(\mathbf{F}_i)$. The four selection strategies are KL, KS, CDV, and Spoilers.

Greedy KL Use rank-order space and greedily choose the feature that yields the minimum feature subset divergence when compared against $\mathbf{R}$. That is, choose $h(\mathbf{F}_i) = F_k$ that minimizes $\Delta_{\mathbf{R}_i - \{R_k\}}$ with respect to $\mathbf{R}$.

KS Adapt the Koller-Sahami algorithm [15] for use in rank-order space. First, find (approximate) Markov blankets for all features, in the Bayesian network of rank-order features (and class) implied by T^R. A Markov blanket for a set of features $\mathbf{F}'$ is another set of features $\mathbf{G}$ whose values, if known, render $\mathbf{F}'$ independent of all others (i.e., $\mathbf{F} - \mathbf{F}' - \mathbf{G}$). This term arises from the graphical models literature where a network encodes conditional independencies, and random variables satisfying the above definition form a 'blanket' around the given set of features. In the Koller-Sahami approach, we remove the feature F_k whose Markov blanket M_k (in rank-order space) yields the minimum feature subset divergence when compared against $M_k \cup R_k$. That is, $h(\mathbf{F}_i) = F_k$ that minimizes Δ_{M_k} with respect to $M_k \cup R_k$.

To approximate the computation of a Markov blanket M_k for a feature R_k, of a given size κ, the approach suggested in [15] is to pick the top κ features that, singly, yield minimum feature subset divergence when pair-wise unioned with R_k.

CDV (Center Distance Vector) Use rank-order space and, for each class, compute centers (ctr) of all permutations for that class given in the dataset. For each pair of classes, compute the Spearman's distances between the centers, obtaining a $\binom{|C|}{2}$-vector. For each feature, remove it, and recompute this vector of distances. $h(\mathbf{F}_i) = F_k$ where R_k is the feature in $\mathbf{R}_i$ that yields the minimum Euclidean distance between the recomputed vector and the original vector (computed from $\mathbf{R}$).

Spoilers Use rank-order space (feature orders) and remove the feature with the highest spoiler count. That is, $h(\mathbf{F}_i) = F_k$ that maximizes $\text{tsp}(R_k)$.

Table 1 identifies the time complexities per step for each of the four algorithms, in terms of $\mathbf{n}$, $\mathbf{c}$, and $\mathbf{m}$, which represent the number of features, classes, and instances, respectively.

The KL algorithm is dominated by the computation of feature subset divergence (Δ). Computing this quantity entails projecting each instance in a dataset and grouping identical instances for δ and Pr computation. The projection operation takes $\mathbf{O(n)}$ time and is carried out for all instances in each class. The computation of Δ thus takes $\mathbf{O(nm)}$ time, since the total number of instances is $\mathbf{m}$. Finally, since Δ is computed for each feature, the KL algorithm takes $\mathbf{O(n^2m)}$ time per iteration.

The KS algorithm takes $\mathbf{O(n^4cm}\kappa\mathbf{)}$ time per iteration, where κ is the predetermined blanket size. This is different from the standard Koller and Sahami

Strategy	Metric Utilized	Time Complexity Per Iteration
Greedy KL	Low feature subset divergence	$\mathbf{O(n^2m)}$
KS	Availability of Markov blanket	$\mathbf{O(n^4cm}\kappa\mathbf{)}$
CDV	Low Spearman's distances	$\mathbf{O(n^2(m+c^2))}$
Spoilers	High spoiler count	$\mathbf{O((n\log n)m^2)}$

Table 1: Details of the four feature selection strategies considered in this paper.

algorithm since the table for the cross-entropy of the class distribution for the given pairs of features (γ_{ij}) needs to be recomputed at each step and not just once at the beginning.

Computing ctr entails scanning each rank in each instance ($\mathbf{O(nm)}$), and computing the distances between each of the $\binom{\mathbf{c}}{2}$ pairs of centers ($\mathbf{O(c^2n)}$). Each of these computations are done $\mathbf{O(n)}$ times. Therefore the time complexity is $\mathbf{O(n^2(m+c^2))}$ per iteration for the CDV algorithm. Kendall [13] shows how the ctr is computed in $\mathbf{O(nm)}$. Essentially, the center rank is the rank of the features as ordered by the sum of their ranks for each feature.

Finally, using the spoiler count requires processing all pairs of instances in a class. There can be no more than $\binom{\mathbf{m}}{2} = \mathbf{O(m^2)}$ such pairs across all classes, since the sum of all class sizes equals $\mathbf{m}$. Determining spoiler counts for each of these pairs takes $\mathbf{O(n \log n)}$ time. The Spoilers algorithm therefore runs in $\mathbf{O((n \log n)m^2)}$ time per iteration. The spoiler count between two instances can be computed in $\mathbf{O(n \log n)}$ time through a modification of merge sort. Essentially a merge sort is performed to sort one permutation to be in the same order as the other. During the merge phase, as an item is merged from the right hand list, its spoiler count is incremented by the number of items remaining in the left hand list. As an item is merged from the left hand list, its spoiler count is incremented by the number of items from the right hand list that have preceded it.

5 Experimental Results

We now present experimental results with the above feature selection strategies, including descriptions of datasets, the experimental results, and discussion.

5.1 Datasets

The four heuristics described in Section 4 are tested against both synthetic and real world datasets. There are numerous ways of generating synthetic datasets, two of which were chosen as representative of the range of possibilities. The first method uses a minimal, known exact answer for each class as a seed for generating the rest of the data. The second method generates random orders for each class that are concordant without knowing the exact minimal answer *a priori*.

The first set of synthetic datasets is generated as follows. Select $\mathbf{c}$, the number of classes in the dataset, and build a dataset starting with $\mathbf{n_o}$ features where $\mathbf{n_o}$ is the smallest integer such that $\mathbf{n_o}! \geq \mathbf{c}$. Generate $\mathbf{c}$ different feature orders, one order for each class. Each of these orders determine an $(\mathbf{r}, c)$ instance-class pair. Select i, and generate i identical instances per class so that $\mathbf{m} = i\mathbf{c}$ is the total number of pairs in the dataset. Select e and r, the number of extraneous and redundant features to be added to the dataset, respectively. An extraneous feature is incorporated into the dataset by considering each instance separately and by uniformly randomly inserting the new feature into the exist-

							Ranks					
Class	Orders						a	b	c	d	e	f
C_1	a	f	e	c	b	d	1	5	4	6	3	2
C_1	a	f	b	c	e	d	1	3	4	6	5	2
C_1	a	f	e	b	d	c	1	4	6	5	3	2
C_2	f	d	c	b	e	a	6	4	3	5	2	1
C_2	b	f	c	e	d	a	6	1	3	5	4	2
C_2	f	d	c	e	a	b	5	6	3	2	4	1
C_3	c	b	e	f	d	a	6	2	1	5	3	4
C_3	e	c	f	d	b	a	6	5	2	4	1	3
C_3	e	c	f	d	b	a	6	5	2	4	1	3

Figure 5: Generated dataset, with $\mathbf{n_0}$=(a,f,e), e=(d,b), and r=(c) where c is redundant with e.

ing order implied in the instance. A redundant feature is incorporated into the dataset by uniformly randomly selecting an existing feature and then for each instance, inserting the new feature either to the left or the right of the selected feature. This way, the new feature and the selected feature have exactly the same relative-order relationships with the other features.

In effect, a dataset with $\mathbf{n} = \mathbf{n_0} + e + r$ features and $\mathbf{m} = i\mathbf{c}$ instances are generated given $\mathbf{c}$, i, e and r. Figure 5 provides an example of a generated dataset for $\mathbf{c} = 3$, $i = 3$, $e = 2$, and $r = 1$. The example starts with $\mathbf{n_0} = 3$ features, since $3! \geq 4$ and 3 is the smallest such value that satisfies the expression.

The second method of generating datasets uses the idea of concordance. For example, Kendall's distance measures concordance between a pair of rank orders. There is a corresponding method for computing the concordance of a set of rank orders, as shown in Figure 6. The τ concordance provides a measure of how close the order of each permutation is to every other permutation within the set. The range of the τ concordance is from 0 to 1. If the concordance is 1, then every permutation has the exact same order; if 0, they are as different from each other as they can possibly be. The complexity of the algorithm is $\mathbf{O(n^2m)}$.

This measure of concordance is used to construct synthetic datasets to test the feature selection strategies. To begin, a random set of ranks is generated containing c instances. The concordance of this set must be below some threshold $C_{\max}$. Each instance of this set forms the basis of a class. New randomly generated instances are added to each class as long as the concordance of that class remains above some threshold $C_{\min}$.

The real world datasets we use come from large scale gene expression data derived from DNA microarray experiments. Here, the instances are the microarray experiments, each feature corresponds to a given gene, and the feature values denote expression levels. Classes denote some classification of the exper-

```
TauConcordance($\Pi$).
INPUT: A set of permutations $\Pi = \{\pi_1 \ldots \pi_m\}$,
       each of length $n$
OUTPUT: The $\tau$ concordance.

1   $S \leftarrow 0$
2   for $i \leftarrow (1 : n-1)$
3       for $j \leftarrow (i+1 : n)$
4           sub $\leftarrow 0$
5           for $k \leftarrow (1 : m)$
6               $\pi \leftarrow \Pi(k)$
7               if $\pi(i) < \pi(j)$ then
8                   sub $\leftarrow$ sub + 1
9               else
10                  sub $\leftarrow$ sub - 1
11          $S \leftarrow S +$ sub$^2$
12  return $(2 \times S/(n^2 - n)(m^2 - m)) - 1/(m-1)$
```

Figure 6: Algorithm for computing the τ concordance for a set of rank orders.

imental conditions. We explore the possibility of considering the relative order between gene expression levels instead of their absolute values. This is because different experiments have different basal levels of expression and hence identifying which genes are more expressed than which other genes reveals greater insight into variabilities and similarities across experiments.

The life cycle of *Drosophila melanogaster* provides the first dataset from Arbeitman, et al. [1]. They examine the life cycle using 74 individual microarray experiments. The expression levels of 3107 genes were examined during the embryotic, larval, pupal, adult male and adult female stages. For our purposes, the classes are hence the life cycle stages. The intent is to identify those genes that are most important to the life cycle.

The second dataset is available from the Whitehead Institute Center for Genome Research at `http://www.broad.mit.edu/cancer`. This dataset is used to examine the leukemia type of a cell. This data was also used by Zhang, Yu, and Singer [28] to test their prediction method which uses a deterministic procedure to form forests of classification trees. There are 72 instances of 7,129 genes, 25 of which have acute myeloid leukemia (AML), 38 instances with B cell acute lymphoblastic leukemia (B-ALL), and 9 instances with T cell acute lymphoblastic leukemia (T-ALL). The dataset was tested in two ways, once with the 2 classes (ALL and AML), and again with all three classes (AML, B-ALL, T-ALL).

5.2 Results

The synthetic datasets we use in our experiments have $\mathbf{c} = 7$ ($\mathbf{n_0} = 4$) and $i = 20$. We used different combinations of values for e and r, where $e \in \{0, 2, 4, 6, 8\}$ and $r \in \{0, 2, 4, 6, 8\}$. Five samples for each combination were generated. Each of the four algorithms were carried out against the different datasets and feature subset divergence was measured at each iteration. Figure 7 displays plots of the extremes of the combinations. The number of features removed versus subset divergence (average across the 5 samples) for each heuristic are shown. The results suggest that the KS algorithm performs poorly with these datasets, whereas there is little distinction between the performance of the other three algorithms. Anecdotal evidence from our early experiments on the KS algorithm have shown that, for rank-order datasets, a small value of i performed marginally better than a larger value. Therefore, for these experiments, i was arbitrarily chosen to be 5.

We also measured Naive Bayes accuracy [22] at each iteration with 5-fold cross validation and a subset of the results are presented in Figure 8. The Naive Bayes classification method and 5-fold cross validation were performed using the Orange data mining system [6] using the equivalent boolean order dataset. The results also show the KS algorithm performing poorly. In addition, the charts suggest that the heuristics based on centers and spoilers perform better than the Greedy KL heuristic, particularly as the number of redundant features in the dataset increases. On the other hand, the Greedy KL heuristic performs slightly better than the other two heuristics when only extraneous features are added to the dataset.

For synthetic datasets generated using concordance methods, different combinations of $C_{\min}$ and $C_{\max}$ were tried. Figures 9 and 10 show two extremes. Both charts show the average of five samples, where $\mathbf{n} = 25$, $i = 100$, and $\mathbf{c} = 5$. Figure 9 has $C_{\min} = 0.1$ and $C_{\max} = 0.9$ so the classes are coherent and distinct. With these conditions, every feature selection strategy does well since almost any feature can be removed with ill effects. Figure 10 has $C_{\min} = 0.9$ and $C_{\max} = 0.1$ so the classes are ill defined and indistinct. In this case, the features need to be selected in a more intelligent manner. Spoilers do better than CDV with in turn does better than Greedy KL.

Figure 11 shows the results for the CDV algorithm and the Spoilers algorithm on the *Drosophilia melanogaster* dataset. The first 3097 features removed by the Spoilers algorithm and the first 3098 features removed by the CDV algorithm maintain a KL divergence of zero. The Naive Bayes accuracies using five-fold cross validation for the last 80 features for both algorithms are approximately the same, and both do better than the average results from uniformly chosen random sets of features. Random sets of size n, where $n = (2 \ldots 30)$ were selected 100 times for each n and their average Naive Bayes accuracy was calculated to provide the baseline comparison. The CDV algorithm found a set of 13 genes that get a 93.3% accuracy as shown in Table 2 while the Spoilers algorithm found a set of 6 genes that get a 93.3% accuracy as shown in Table 3.

For the cancer datasets using both two and three classes, shown in Fig-

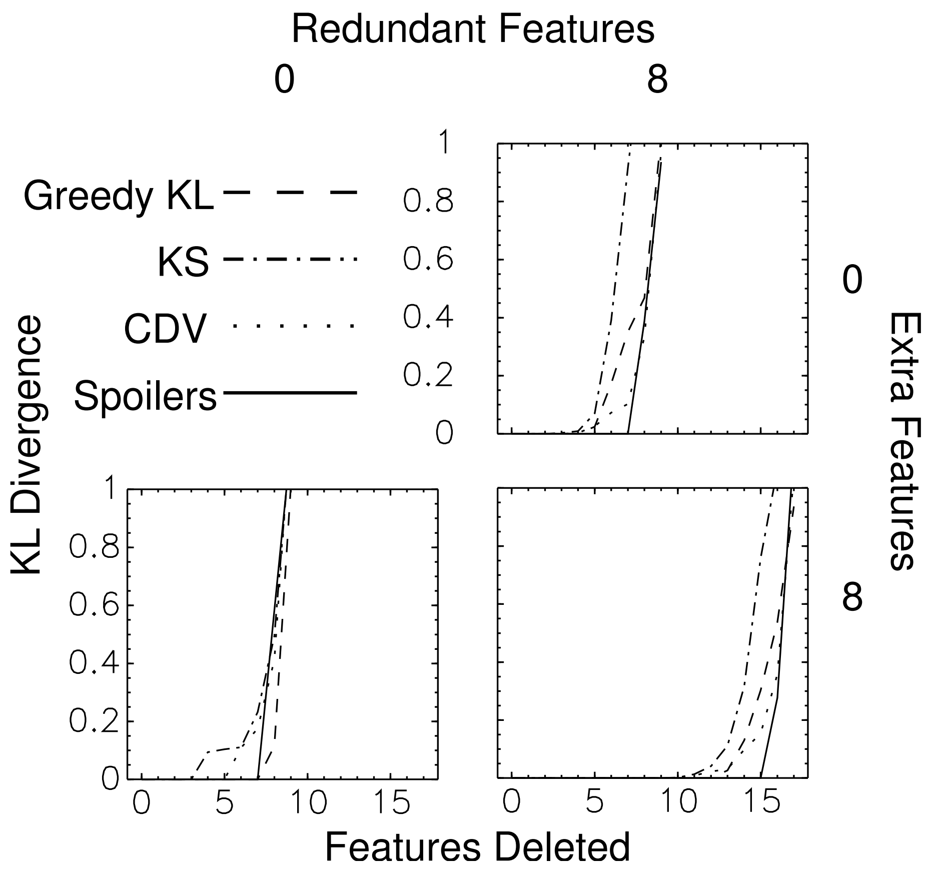

Figure 7: KL divergence for synthetic datasets showing the effects of redundant and/or extra features for different feature selection methods.

ures 12 and 13 respectively, the CDV algorithm performed markedly better than the Spoilers algorithm. The Spoilers algorithm did no better than the average random sample. The CDV algorithm found a set of 25 genes, shown in Table 4, for two classes that had a Naive Bayes five-fold cross validation accuracy of 93% and a set of 16 genes, shown in Table 5, that had an accuracy of 97% for three classes. This is reflected in the KL divergences which the CDV algorithm maintains to be zero longer than the Spoilers algorithm manages to do.

The results for the cancer datasets differ from Zhang et al. in the gene subsets that are considered important for classification. There are several differences in the goals of the algorithms. Our method performs feature selection only, while the Zhang et al. algorithm does both feature selection and classification.

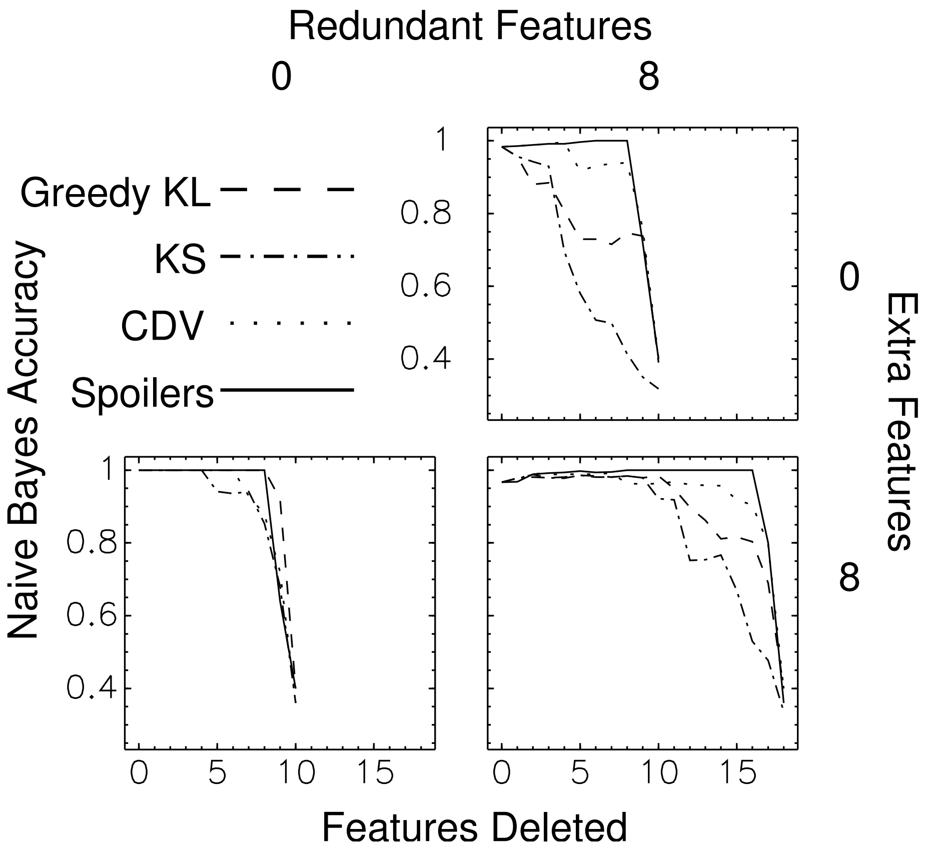

Figure 8: Naive Bayes five-fold cross validation accuracies for synthetic datasets showing the effects of redundant and/or extra features for different feature selection methods.

The Zhang et al. method is based upon the absolute values which are more variable under different experimental conditions than comparing relative values. As Tables 4 and 5 reveal, not withstanding that they have only two genes in common, many pertinent features have been picked by the CDV algorithm (e.g., a serine kinase, a cell cycle controller, a proto-oncogene, an oncostatin, all of which undoubtedly influence the progression of cancer).

5.3 Discussion

The experiments that were carried out suggest that the KS algorithm does not work well with rank-order spaces. The result is not surprising since the Koller-Sahami method treats ranks as independent values and do not particu-

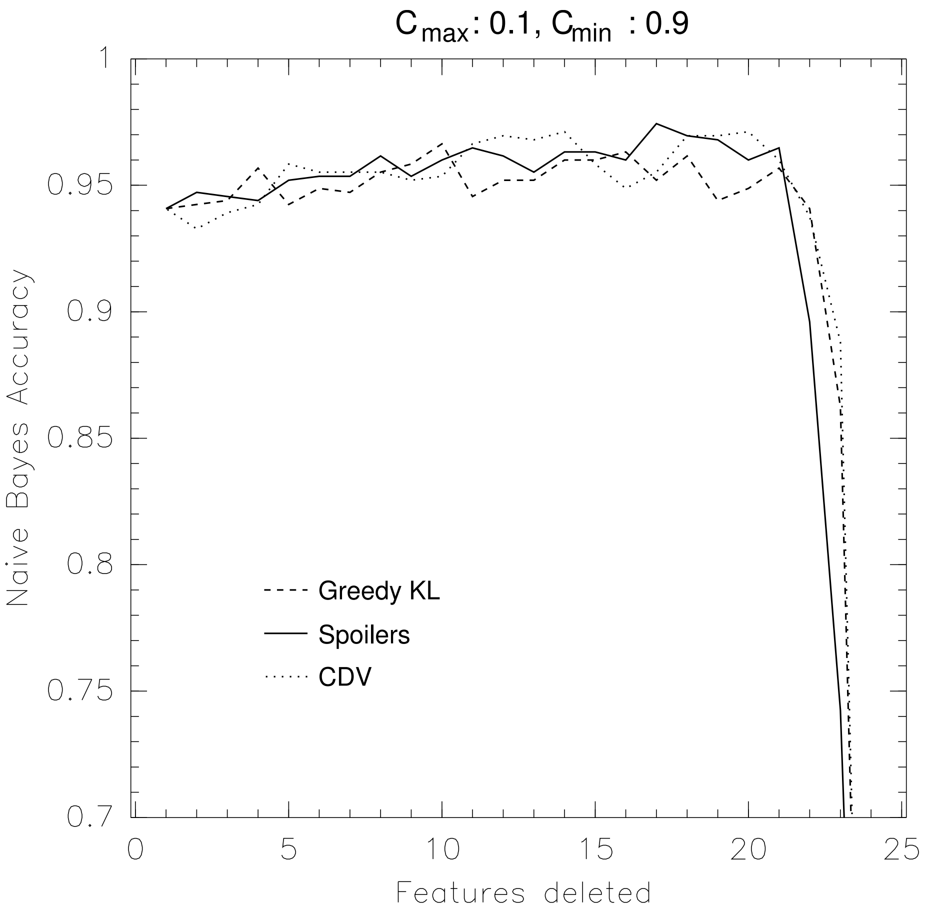

Figure 9: Naive Bayes five-fold cross validation accuracies for an easy synthetic dataset generated using concordance methods.

larly distinguish between its relationship with ranks of other features.

The experiments also suggest that the CDV algorithm and the Spoilers algorithm generally perform better than methods that directly aim to maintain low KL-divergence (the Greedy KL algorithm and the KS algorithm). To understand the success of these methods over the KL-divergence based methods, examine the dataset shown in Figure 14. For this dataset, features x and y are consistent across the classes. However, any feature except x may be removed and still maintain a KL divergence of zero. In addition, if y is removed, then any further feature selection process will not yield the optimal result. For this dataset, both the CDV algorithm and the Spoilers algorithm will retain y and correctly choose either w or z because they examine the underlying ordinal

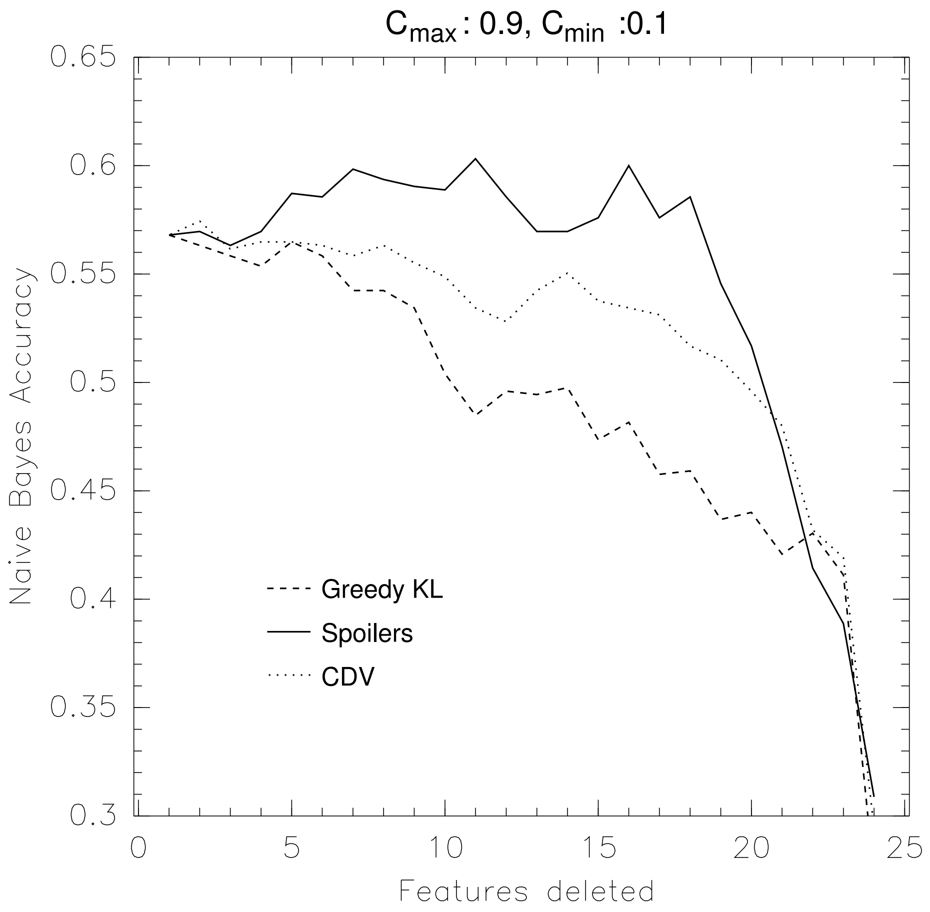

Figure 10: Naive Bayes five-fold cross validation accuracies for a hard synthetic dataset generated using concordance methods.

structure of the instances. In particular, the CDV algorithm deliberately addresses ordinal differences across instances in different classes while the Spoilers algorithm addresses similarities between instances in a class. On the other hand, the conditions imposed by the KL algorithm to arrive at a subset divergence of zero is easily satisfied resulting in a significant number of removable features, at the onset.

The phenomenon of multiple removable features is illustrated in Figure 15. The figure shows the KL divergence in greyscale for all of the feature selection subsets of length 1 through 4 from the original dataset given in Figure 5. The subset with the higest KL divergence is (a,b,c,e) with a KL divergence of 1.343 and is denoted with black. Those choices that have a zero KL divergence are

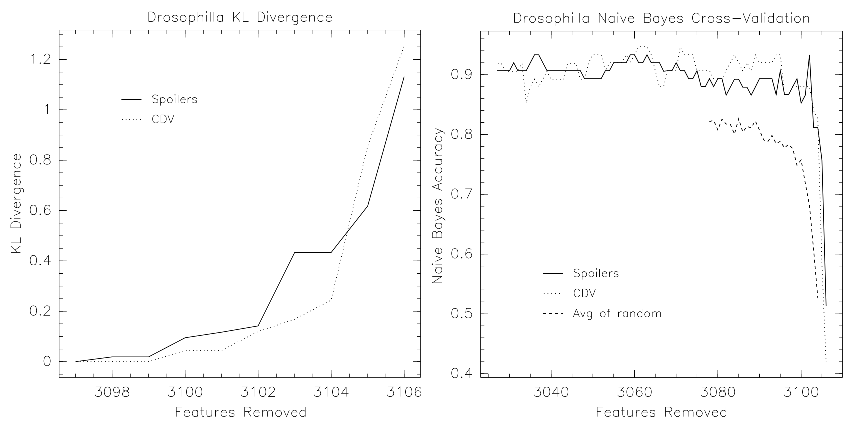

Figure 11: Results of two feature selection methods for the *Drosophilia melanogaster* life-cycle experiments.

Gene	**Annotation**
CG10033	foraging, cGMP-dependent protein kinase activity
CG10602	leukotriene-A4 hydrolase
CG12120	unknown function
CG12321	unknown function
CG12699	unknown function
CG14722	apoptotic protease activator activity
CG2019	peroxidase activity
CG2985	structural molecule activity involved in vitellogenesis
CG3991	tripeptidyl-peptidase II activity putatively
CG5940	cyclin-dependent protein kinase regulator activity
CG6483	serine-type endopeptidase activity
CG6933	structural constituent of peritrophic membrane (sensu Insecta)
CG7157	hormone activity involved in sperm storage

Table 2: Subset of 13 genes, chosen by the CDV algorithm, that yield a 93.3% accuracy using Naive Bayes five-fold cross validation when used to predict the *Drosophilia melanogaster* life-cycle.

shown with white and the rest have various shades inbetween based upon their magnitude. Note that any of the features can be selected at first and still have a KL divergence of zero. However, of the 17 possible paths leading to 2 features removed (still with a zero KL divergence), only 12 paths lead to the 2 choices of 3 features removed. A feature selection strategy based upon a greedy selection minimizing KL divergence at each step would only find one of these 2 best sets

Gene	Annotation
CG10096	unknown function
CG1140	3-oxoacid CoA-transferase activity
CG13841	unknown function
CG3121	putative microtubule binding
CG3333	putative pseudouridylate synthase activity
CG7157	hormone activity involved in sperm storage

Table 3: Subset of 6 genes, chosen by the Spoilers algorithm, that yield a 93.3% accuracy using Naive Bayes five-fold cross validation when used to predict the *Drosophilia melanogaster* life-cycle.

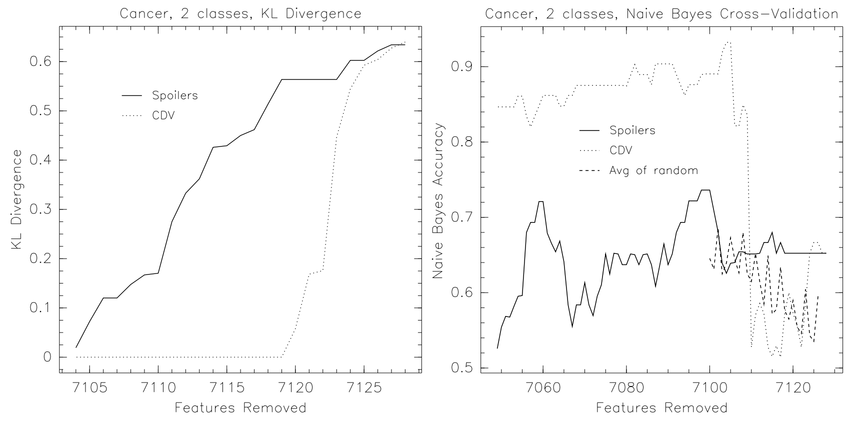

Figure 12: Results for two feature selection methods using the cancer dataset with two classes.

70% of the time. This ratio decreases as the problem size increases.

Of course, the other feature selection strategies do not guarantee optimal results either. The KS algorithm chooses (a,b,c) resulting in a KL divergence of 0.249, the CDV algorithm chooses (b,c,f) which has a KL divergence of 0.313 and the Spoilers algorithm chooses (b,c,e) with a KL divergence of 0.616.

Almost all experiments showed the Spoilers algorithm performing as well as the CDV algorithm, with the exception of the cancer dataset. For the experiments on the cancer datasets, Spoilers performed no better then the average random sample. The cancer data contained a large number features (7000+) and a small set of classes (2 or 3). In this case, the CDV algorithm might be better at global optimization than the Spoilers algorithm, which may have been confused by several different local optimal choices early in the feature selection process. However, except for this extreme case, the Spoilers algorithm

Gene	**Annotation**
D49824	HLA-B null allele mRNA
D70830	mRNA for Doc2 beta
HT2696	cholinesterase-related cell division controller
HT2798	Serine/Threonine Kinase Z25424
L13977	prolylcarboxypeptidase mRNA
M17733	thymosin beta-4 mRNA
M24899	triiodothyronine (ear7) mRNA
M27288	oncostatin M gene, exon 3
M27891	cystatin C (CST3) gene, exon 3
S94421	TCR eta=T cell receptor eta-exon
U06155	chromosome 1q subtelomeric sequence D1S553
U09953	ribosomal protein L9 mRNA
U14969	ribosomal protein L28 mRNA
U15422	protamine 1, protamine 2, and transition protein 2 genes
U28727	pregnancy-associated plasma protein-A preproform (PAPPA)
U33447	putative G-protein-coupled receptor (GPR17) gene
U63717	osteoclast stimulating factor mRNA
U79267	clone 23840 mRNA
U89717	9-cis-retinol specific dehydrogenase mRNA
U89922	lymphotoxin beta isoform variant, alternatively spliced mRNA
X02160	mRNA for insulin receptor precursor
X73460	mRNA for ribosomal protein L3
X79234	mRNA for ribosomal protein L11
X90846	mRNA for mixed lineage kinase 2
Z48579	mRNA for disintegrin-metalloprotease (partial)

Table 4: Subset of 25 genes that yield a 93% accuracy using Naive Bayes five-fold cross validation when used to predict the leukemia type (2 classes) of a cell.

performed well, and is much faster than the other algorithms. It is also possible that the methods of generating artificial rank-order data contained a bias toward feature selection using the Spoilers algorithm, since the generation method depends upon order.

6 Related Work

The feature selection problem considered in this paper is unique in its formulation although it has similar motivations in existing literature on feature selection. The importance of removing redundant as well as irrelevant features was recognized early in machine learning research and several theoretical frameworks have been put forth [16, 19, 3]. The work by Koller and Sahami, as is our re-

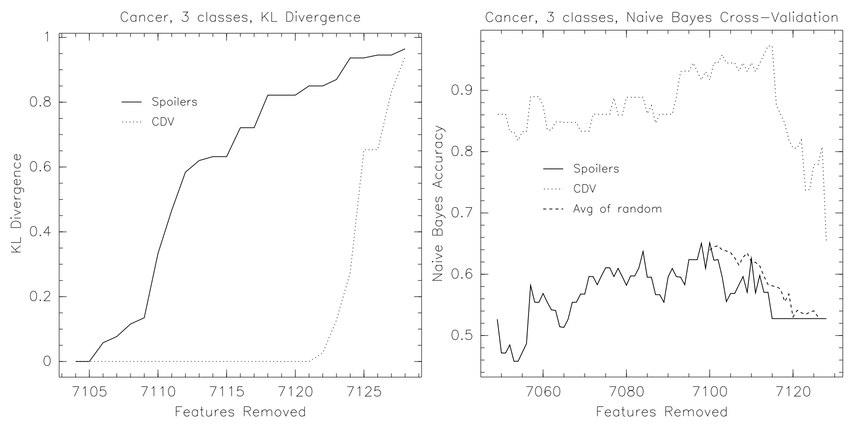

Figure 13: Results for two feature selection methods using the cancer dataset with three classes.

Gene	**Annotation**
D13789	mRNA for N-acetylglucosaminyltransferase III
HT26388	Mucin 1, Epithelial
L38941	ribosomal protein L34 (RPL34) mRNA
M17886	acidic ribosomal phosphoprotein P1 mRNA
M24194	MHC protein homologous to chicken B complex protein mRNA
M31606	phosphorylase kinase (PSK-C3) mRNA
M35093	secreted epithelial tumor mucin antigen (MUC1) gene
M84526	adipsin/complement factor D mRNA
M89957	cell surface glycoprotein (IGB) mRNA
U14969	ribosomal protein L28 mRNA
U86358	chemokine (TECK) mRNA
U89922	lymphotoxin beta isoform variant, alternatively spliced mRNA
X00437	mRNA for T-cell specific protein
X52056	mRNA for spi-1 proto-oncogene
X54741	CYPXIB2 gene for aldosterone synthase
X90846	mRNA for mixed lineage kinase 2

Table 5: Subset of 16 genes that yield a 97% accuracy using Naive Bayes five-fold cross validation when used to predict the leukemia type (3 classes) of a cell.

search, follows the preprocessing paradigm where feature selection is considered as preliminary to induction, and hence learning-algorithm-agnostic. The contrasting idea is the wrapper-based approach where feature selection is studied

					Ranks			
Class	Orders				w	x	y	z
C_1	w	x	y	z	1	2	3	4
C_1	x	w	z	y	2	1	4	3
C_2	y	x	z	w	4	2	1	3
C_2	w	z	y	x	1	4	3	2

Figure 14: Small rank-order dataset.

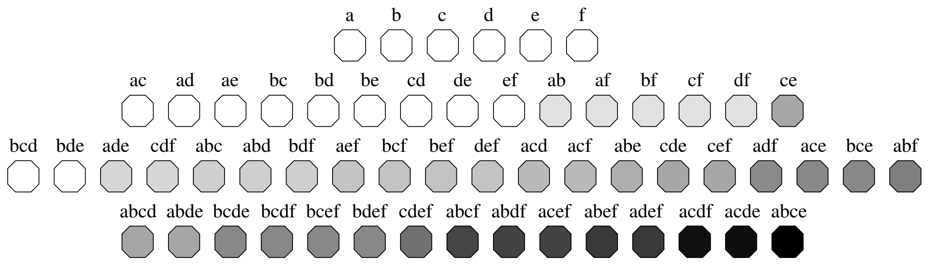

Figure 15: Chart showing the spectrum of KL divergences for all feature selection subsets for the data shown in Figure 5. Black denotes the highest KL divergence and white has a KL divergence of zero.

in the context of a specific learning algorithm (e.g., kernel machines [9]).

Order theoretic considerations have been introduced in different guises in machine learning research. Cohen et al. [4] describe how to induce an (approximate) global order from given (partial) rankings or preference information. A related problem is considered by Mannila and Meek [20] who view partial orders as generative models of sequential trace data. Kamishima and Akaho [12] extend the Cohen *et al.,* work in 'Learning from Order Examples' by accommodating problems where inputs are themselves orders. A compelling application context for these ideas is described in Kamishima's Nantonac collaborative filtering [11]. More recently, Lebanon and Lafferty [17] present a boosting-like algorithm to combine multiple rankings by reasoning about probability distributions over perturbations. This work is extended in [18] into a unifying framework for classification and ranking. These papers do not directly either address the problem of feature selection or even a supervised learning scenario from orders to discrete classes. Gionis et al. [7] study 'fragments of order' although their setting posits unlabeled data and the learned orders are actually sets of association rules summarized as dependencies. Sai et al. [23] describe a scenario closer to our work where dataset instances are viewed only through ordinal comparisons across features but their goal is to learn association rules between such ordinal comparisons, not to predict a class or feature selection. The work presented in this paper is different from all the above works either the viewpoint of the dataset (labeled), the input instance (a total order over features), or the desired output (a reduced set of features suitable for defining relationships from orders to classes). This context was first motivated in Slotta et al. [25] using a proteomics application domain.

7 Conclusions

Through considerations of preserving class-conditional distributions, we have presented four feature selection algorithms for reasoning in rank-order spaces. Via experiments on both synthetic and real-world datasets, we have identified a number of conclusions relating the way the heuristics operate, the characteristics of the datasets, and the performance results.

It appears that KL divergence is not the best measure to use for a feature selection heuristic, since it is likely that a number of random features may be removed from rank-order space and still have a KL divergence of zero. Only in the later stages does the KL divergence become meaningful. Our experiments show that the Naive Bayes classifier showed adverse effects much sooner for the KL-divergence based methods, even though the KL divergence was unchanged. This indicates that these strategies were making incorrect choices, leading them down a suboptimal path in the feature selection tree. On the other hand, the CDV algorithm and the Spoilers algorithm tend to make the correct choices as these algorithms deliberately examine the underlying ordinal structure of the instances.

This research opens several questions for future exploration. Is there a method for generating artificial rank-order data that better models real world datasets? Is there a faster algorithm for CDV so that its speed is favorably comparable with the Spoilers algorithm? Is there a better measure of divergence for rank-orders than KL divergence, one that considers relationships between features instead of just their values? Finally, are there better classification algorithms for rank-ordered data?

8 acknowledgements*

This work is supported through the Multidisciplinary University Research Initiative (MURI) program of the Department of Defense (Biomimetic Cell and Tissue Stasis; N00014-01-1-0852), the Metabolic Engineering for Cellular Stasis program of DARPA (N00173-98-1-G005-P00004 and N00173-02-1-G016), and the National Science Foundation (EIA-0103660, IBN-0219332, and ITR-0219322).

References

[1] Arbeitman, M., E. Furlong, F. Imam, E. Johnson, B. Null, B. Baker, M. Krasnow, M. Scott, R. Davis, and K. White: 2002, ‘Gene Expression During

the Life Cycle of *Drosophilia melanogaster*'. *Science* **297**, 2270–2275.

[2] Arrow, K.: 1970, *Social Choice and Individual Values.* Yale University Press, second edition.

[3] Blum, A. L. and P. Langley: 1997, 'Selection of relevant features and examples in machine learning'. *Artificial Intelligence* **97**(1-2), 245–271.

[4] Cohen, W., R. Schapire, and Y. Singer: 1999, 'Learning to Order Things'. *Journal of Artificial Intelligence Research* **Vol. 10**, pages 243–270.

[5] Cover, T. and J. Thomas: 1991, *Elements of Information Theory.* John Wiley and Sons.

[6] Demsar, J. and B. Zupan: 2004, 'Orange: From Experimental Machine Learning to Interactive Data Mining'. White paper, Faculty of Computer and Information Science, University of Ljubljana. http://www.ailab.si/orange.

[7] Gionis, A., T. Kujala, and H. Mannila: 2003, 'Fragments of Order'. In: *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* pp. 129–136, ACM Press.

[8] Grötschel, M., M. Jünger, and G. Reinelt: 1984, 'A Cutting Plane Algorithm for the Linear Ordering Problem'. *Operations Research* **32**(6), 1195–1220.

[9] Guyon, I. and A. Elisseeff: 2003, 'An Introduction to Variable and Feature Selection'. *Journal of Machine Learning Research* **3**, 1157–1182.

[10] Hastie, T., R. Tibshirani, and J. Friedman: 2001, *The Elements of Satistical Learning.* Springer-Verlag.

[11] Kamishima, T.: 2003, 'Nantonac Collaborative Filtering: Recommendation based on Order Responses'. In: *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* pp. 583–588, ACM Press.

[12] Kamishima, T. and S. Akaho: 2002, 'Learning From Order Examples'. In: *IEEE International Conference on Data Mining.* pp. 645–648, IEEE Computer Society.

[13] Kendall, M. and J. Gibbons: 1990, *Rank Correlation Methods.* Oxford University Press.

[14] Knuth, D.: 1998, *The Art of Computer Programming*, Vol. 3 - Sorting and Searching, pp. 11–22. Addison Wesley, second edition.

[15] Koller, D. and M. Sahami: 1996, 'Toward Optimal Feature Selection'. In: *Proceedings of the Thirteenth International Conference on Machine Learning (ICML'96).* pp. 284–292.

[16] Last, M., A. Kandel, and O. Maimon: 2001, 'Information-Theoretic Algorithm for Feature Selection'. *Pattern Recognition Letters* **22**(6/7), 799–811.

[17] Lebanon, G. and J. Lafferty: 2002, 'Cranking: Combining Rankings Using Conditional Probability Models on Permutations'. In: *ICML*. pp. 363–370.

[18] Lebanon, G. and J. Lafferty: 2003, 'Conditional Models on the Ranking Poset'. In: *Advances in Neural Information Processing Systems 15*. pp. 431–438.

[19] Littlestone, N.: 1988, 'Learning Quickly When Irrelevant Attributes Abound: A New Linear-Threshold Algorithm'. *Machine Learning* **2**(4), 285–318.

[20] Mannila, H. and C. Meek: 2000, 'Global partial orders from sequential data'. In: *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge discovery and Data Mining*. pp. 161–168, ACM Press.

[21] Marden, J.: 1996, *Analyzing and Modeling Rank Data*. CRC Press.

[22] Mitchell, T.: 1997, *Machine Learning*. McGraw Hill.

[23] Sai, Y., Y. Yao, and N. Zhong: 2001, 'Data Analysis and Mining in Ordered Information Tables'. In: *Proceedings of the IEEE International Conference on Data Mining*. pp. 497–504.

[24] Sen, A.: 1981, 'Social Choice Theory'. In: K. Arrow and M. Intriligator (eds.): *Handbook of Mathematical Economics*, Vol. 3. Elsevier, Chapt. 22.

[25] Slotta, D. J., L. S. Heath, N. Ramakrishnan, R. Helm, and M. Potts: 2003, 'Clustering Mass Spectrometry Data Using Order Statistics'. *Proteomics* **3**(9), 1687–1691.

[26] Spearman, C.: 1904, 'The Proof and Measurement of Association Between Two Things'. *American Journal of Psychology* **15**, 72–101.

[27] Spellman, P., G. Sherlock, M. Zhang, V. Iyer, K. Anders, M. Eisen, P. Brown, D. Botstein, and B. Futcher: 1998, 'Comprehensive Identification of Cell Cycle–regulated Genes of the Yeast *Saccharomyces cerevisiae* by Microarray Hybridization'. *Molecular Biology of the Cell* **9**, 3273–3297.

[28] Zhang, H., C. Yu, and B. Singer: 2003, 'Cell and tumor classification using gene expression data: Construction of forests'. *PNAS* **100**(7), 4168–4172.

[29] Zhang, X., H. Begleiter, B. Porjesz, W. Wang, and A. Litke: 1995, 'Event related potentials during object recognition tasks'. *Brain Research Bulletin* **38**(6), 531–538.