Feature Selection in Web Applications By ROC Inflections and Powerset Pruning

Frans M. Coetzee[1], Eric Glover[1], Steve Lawrence[1], C. Lee Giles[1,2]

[1] NEC Research Institute, 4 Independence Way, Princeton, NJ 08540

[2] Information Sciences and Technology, Pennsylvania State University, University Park, PA 16801

E-mail: {coetzee,compuman,lawrence,giles}@research.nj.nec.com

Abstract

A basic problem of information processing is selecting enough features to ensure that events are accurately represented for classification problems, while simultaneously minimizing storage and processing of irrelevant or marginally important features. To address this problem, feature selection procedures perform a search through the feature power set to find the smallest subset meeting performance requirements. Major restrictions of existing procedures are that they typically explicitly or implicitly assume a fixed operating point, and make limited use of the statistical structure of the feature power set. We present a method that combines the Neyman-Pearson design procedure on finite data, with the directed set structure of the Receiver Operating Curves on the feature subsets, to determine the maximal size of the feature subsets that can be ranked in a given problem. The search can then be restricted to the smaller subsets, resulting in significant reductions in computational complexity. Optimizing the overall Receiver Operating Curve also allows for end users to select different operating points and cost functions to optimize. The algorithm also produces a natural method of Boolean representation of the minimal feature combinations that best describe the data near a given operating point. These representations are especially appropriate when describing data using common text-related features useful on the web, such as thresholded TFIDF data. We show how to use these results to perform automatic Boolean query modification generation for distributed databases, such as niche metasearch engines.

1 Introduction

A basic problem of information processing is selecting enough features to ensure that events are accurately represented for use in classification problems, while simultaneously minimizing the number of irrelevant or marginally important features. Selection of the smallest size feature set meeting performance requirements not only reduces storage requirements and computational complexity, but is required to ensure good generalization of the final classifiers [1, 2].

The feature selection problem becomes acute when the operating point (recall/precision setting) that will be used by the end user is unknown. In metasearch engines for example, load allocation, micro-payment schedules, and bandwidth requirements can result in different desired operating points for querying the secondary sources [3, 4, 5, 6]. Similarly, when storing and indexing the massive quantities of financial news available on the web for later use in risk assessment, the exact cost and risk functions that will be used by the end users cannot be known in advance.

In this paper we address this feature selection problem for two-class classification applications on discrete feature sets. Our objective is to select the most parsimonious subset of features that can meet a range of future classification performance requirements. In contrast to most feature selection procedures [7, 8, 9, 2, 10], where either explicitly or implicitly a specific operating point is assumed, we consider the optimization of the complete Receiver Operating Curve (ROC). The Receiver Operating Curve (ROC) is the minimal complete representation of classification performance on a subset of the features. This one-dimensional curve completely summarizes the optimal tradeoff between precision and recall, independent of the dimension of the feature space. The ROC is parameterized by the classifiers on the feature subset that yield the maximal recall for a given level of precision; however, the ROC depends only on the statistics of the data. Finally, the ROC curve captures all possible ratios of *a-priori* probabilities of the true and false class; hence, changes in these probabilities that occur in dynamic environments such as the web can flexibly be compensated for without redesign of the classifiers.

Optimal feature selection requires computing the ROC on each possible subset of features, and ranking the subsets according to performance and size. The optimal feature set to use will depend on the operating point; a minimal covering set can be selected for indexing. Unfortunately, there are a number of practical difficulties with this ideal approach. First, the size of the feature power set is exponential in the number of features, which prevents exhaustive characterization of feature subsets of even relatively modest size. Second, while the optimally efficient procedure for calculating the ROC on a single subset given the class-conditional densities has long been known, namely the Neyman-Pearson (NP) design procedure [11, 12], little is known about computing the ROC when data is finite. Third, computing the ROC on a given subset is expensive for large sets of features, even when the class densities are known. These elements combine to make feature selection a remarkably challenging problem.

Existing feature selection procedures have mostly ignored the problem of complete ROC characterization by implicitly assuming an operating point. These approaches use the classification rate of a single classifier, or a scalar

0-7695-0942-8/01 $10.00 © 2001 IEEE

metric such as Kullback-Leibler divergence, as a proxy for overall performance on a given subset [7, 8, 9]. (The excellent approach in [7] uses the more sophisticated notion of a Markov blanket, but in implementation a metric such as entropy still has to be used to estimate dependence). These assumptions result in significantly inferior performance when a different operating point or cost function is later used, and a lack of flexibility in adjusting to operating environment changes. Unfortunately, classification performance cannot be completely summarized by a scalar.

A less obvious problem results from the still incomplete understanding of the performance of existing induction procedures on small data sets. Most researchers focus on using different strategies for guiding the search through the feature power set, assuming that the performance on each subset can be reliably represented (typically, cross validation is used to control overfitting). The more subtle but crucial point is that in order to compare two subsets, such as when a feature is eliminated, the induction algorithm also has to guarantee that the classifier structures used to characterize two subsets are both near optimal. Unfortunately, except for exhaustive optimization, none of the induction approaches used in existing feature selection algorithms are known to satisfy this property on arbitrarily large feature subsets.

In addition to our considering the overall ROC, our method therefore differs from existing approaches due to its extensive exploitation of the statistical properties of the induction algorithm (Neyman-Pearson design) on finite data [13]. The key result from this analysis is that even when sufficient data exists such that the classification performance of a given classifier can be obtained with high accuracy, the probability of finding the optimal classifiers on a feature subset will become arbitrarily small as the feature subset size increases. Subset selection procedures can therefore only reliably estimate the optimal possible performance on a feature subset, and the ranking of a subset, when the number of features is less than some data-dependent limit. Our approach dynamically estimates this critical subset size to dynamically prune the feature power set, reducing the problem from an exponential, to a polynomial search problem.

The paper is arranged as follows. Section 2 formalizes the feature selection and classifier design procedure, and summarizes the necessary major statistical results from [13]. New work starts in Section 3 where we define a filter structure (a form of partial ordering) on the feature subsets using the Receiver Operating Curves. We show how the NP design analysis can be combined with the ROC filter structure to define a method for pruning the feature power set. We illustrate the performance of the algorithm on a synthetic example, where the statistics are known.

One advantage of the enumeration of the classifiers performed by the Neyman-Pearson algorithm is that the classifier structures can be manipulated to yield a Boolean representation of the minimal feature combinations that best describe the data near a given operating point. These representations are especially appropriate for describing data using common text-related features useful on the web, such as thresholded TFIDF data. In Section 4, we apply the approach to selecting features for representing conference calls for papers in an index, and also for constructing simple classifiers that can be used for query modifications in a search engine.

2 Subset and Classifier Characterization

This section summarizes the Neyman-Pearson design procedure, and recent results that describe the performance of this method when applied to finite data [13].

Assume that we are given N possible features, $Q^* = \{x_0^*, x_1^*, \ldots x_{N-1}^*\}$. For simplicity, we assume that the features are binary, that is $x_i^* \in \{0,1\}$. For example, these features can be thresholded TFIDF values obtained on documents. The analysis carries over in a straightforward manner when x_i^* assumes values in any finite alphabet. We assume two hypotheses H_0 (false class, or irrelevant) and H_1 (true class, relevant) on the input space $\chi(Q^*) = \prod_{j=0}^{N-1}\{0,1\}$ with class conditional probabilities $\mathcal{P}\{x^* \mid H_0\}$ and $\mathcal{P}\{x^* \mid H_1\}$ respectively.

For a given feature subset $Q \subseteq Q^*$ consisting of l features, we obtain a set of possible inputs $\chi(Q) = \{x = (x_0, x_1, \ldots x_{l-1}) | x_i \in Q^*\}$. Each sample $x \in \chi(Q)$ is denoted by a bit string of length l. We adopt the convention of denoting vectors using symbols without subscripts, with vector elements indicated by subscripts. We further frequently associate the bit string x with its integer mapping, e.g. $x = (x_0, x_1) = (1,0) = 2$.

Given a feature subset Q, a classifier function $\Gamma : \chi(Q) \rightarrow \{0,1\}$ assigns labels, either 0 or 1, to every element in the binary sequence space $\chi(Q)$, thereby forming decision regions $\mathcal{L}(\Gamma)_0$ and $\mathcal{L}(\Gamma)_1$ in $\chi(Q)$ for the two classes respectively. There are 2^{2^l} distinct different classification rules $\Gamma_j, j = 0, 1, \ldots 2^{2^l} - 1$ for separating the two classes. Each of these decision rules yields a probability of false alarm $P_f(\Gamma)$ and of detection $P_d(\Gamma)$, defined by

$$\begin{aligned} P_f(\Gamma) &= \sum_{x \in \mathcal{L}(\Gamma)_1} \mathcal{P}\{x \mid H_0\} \\ P_d(\Gamma) &= \sum_{x \in \mathcal{L}(\Gamma)_1} \mathcal{P}\{x \mid H_1\} \end{aligned} \tag{1}$$

The set $AOS = \{(P_f(\Gamma_j), P_d(\Gamma_j))\}$ of the operating points defined by the 2^{2^l} binary mappings on a feature set is referred to as the *Achievable Operating Set*. We note that by switching between the outputs of two classifiers with some probability, any operating point (P_f, P_d) on the line connecting the operating points of the two classifiers can be produced [11]. Hence, any operating point within the convex hull of the achievable operating set can be implemented.

The Receiver Operating Curve (ROC) is the set of operating points yielding the maximal detection rate for a given false alarm rate. The ROC efficiently summarizes the inherent difficulty of separating the two classes on a given subset. The subset of the 2^{2^l} classifiers Γ_j lying on the ROC will be referred to as the ROC support classifiers.

Exhaustive enumeration of the classifiers on a subset is and will be practically impossible except for trivially small

cases (even when $l = 5$, $2^{2^l} \simeq 4.3 \times 10^9$). An induction procedure is required for finding the ROC support classifiers. The Neyman-Pearson (NP) design procedure provides a solution to the problem of efficiently obtaining the ROC and its support classifiers when the class probabilities are known (e.g. when infinite data is available). Under these conditions, this method provably is the most efficient method for obtaining the ROC. According to this theory the 2^l possible strings x are ranked according to the likelihood ratio function $\zeta : \chi \rightarrow \Re^+$

$$\zeta(x) \quad = \quad \mathcal{P}\{x \mid H_1\}/\mathcal{P}\{x \mid H_0\} \qquad (2)$$

The ROC support classifiers are provably found in order of increasing false alarm performance by successively assigning strings in decreasing order of likelihood ratio to the true class decision region. Hence, there are 2^l ROC support classifiers.

We make the above discussion concrete by providing a simple example. Consider a simple two-feature test subset $x = (x_0, x_1)$, where the true density functions are as follows:

$x_1x_0(x)$	00(0)	01(1)	10(2)	11(3)
$p(x\|H_1)$	0.30	0.35	0.20	0.15
$p(x\|H_0)$	0.15	0.25	0.40	0.20

The ROC support classifiers $\Gamma^i(x), i = 0, 1, 2, 3$ developed via the NP design are as shown below:

$x_1, x_0(x)$	00(0)	01(1)	10(2)	11(3)	P_f	P_d
$\zeta(x)$	2.00	1.40	0.50	0.75		
$rank(\zeta(x))$	0	1	3	2		
$\Gamma^0(x)$	0	0	0	0	0.00	0.00
$\Gamma^1(x)$	1	0	0	0	0.15	0.30
$\Gamma^2(x)$	1	1	0	0	0.40	0.65
$\Gamma^3(x)$	1	1	0	1	0.60	0.80
$\Gamma^4(x)$	1	1	1	1	1.00	1.00

For example, to achieve a detection rate of 65% at a false alarm rate of 40%, we can use classifier $\Gamma^2(x)$, which classifies a document as relevant when feature one is 0 (i.e. both features absent, or only feature 0 present).

The Neyman-Pearson design approach is a search procedure, where the problem of finding the classifier function Γ that maximizes the P_d at a given value of P_f is reduced from searching a space of dimension 2^{2^l} to one of searching a space of dimension 2^l, an enormous reduction in complexity. Further, aside from exhaustive enumeration, no other general induction procedure is known to the authors that can always reliably provide the ROC curve, even given the class densities.

In practice, the class conditional distributions are unknown and statistics must be estimated from a finite labeled data set. Formally, we consider the set of all possible class conditional densities as a sample space Θ. Each classification problem is generated by sampling two elements from Θ, yielding the values $\theta_{j|Hi} = p(x = j|H_i), j = 0, 1, \ldots L-1, i = 0, 1$ where $L = 2^l$. We assume these two class distributions are independent. A finite data set is drawn independently from each of these class conditional distributions, with n_i samples yielding $k_{j|Hi}$ successes (occurrences) of symbol x_j for class $H_i, i = 0, 1$. We assume the class data is labeled correctly. The data is used to compute class conditional histograms (or density estimates $\hat{\theta}_{j|Hi} = k_{j|Hi}/n_i$), and likelihood ratio estimates

$$\hat{\zeta}_j \quad = \quad \hat{\theta}_{j|H1}/\hat{\theta}_{j|H0} \qquad (3)$$

On finite data, the ROC is calculated by applying the NP procedure for ranking the histogram bins (each bin corresponding to a particular bit string) according to the likelihood ratio estimates $\hat{\zeta}_j$. There is a one-to-one mapping between the ranking, and the set of classifiers that the procedure yields as its estimate of the ROC support classifiers.

A complete discussion of the propagation of errors that occur through the design procedure can be found in [13]. However, the following major trends are of importance for this paper:

1. Given a particular binary classifier, and an *independent* set of n_1 samples from the true class, the true detection performance can be bound in a region around the sample estimate of the detection rate. Assume that k of the n_1 samples are classified correctly, and $\hat{P}_d = k/n_1$. The posterior distribution $\mathcal{P}\left\{P_d \mid \hat{P}_d\right\} \simeq beta(n_1, n_1\hat{P}_d)$ where

$$p_{P_d|\hat{P}_d}(y|\hat{P}_d) \quad = \quad \frac{y^k(1-y)^{n_1-k}}{\beta(k+1, n_1-k+1)} \qquad (4)$$

The estimated false alarm rate similarly localizes the true underlying false alarm rate, with $\mathcal{P}\left\{P_f \mid \hat{P}_f\right\} \simeq beta(n_0, n_0\hat{P}_f)$.

The importance of this result is that for a given classifier Γ, bounding the location of the true operating point $(P_f(\Gamma), P_d(\Gamma))$ around the estimated operating point $(\hat{P}_f(\Gamma), \hat{P}_d(\Gamma))$ can be done with arbitrary confidence purely as a function of the number of data samples n_0 and n_1 used for evaluation. This bound is independent of feature subset size. For example, localization of over 90% probability of the posterior probability occurs in the interval $\hat{P}_d \pm 2.5\%$ when $n_1 = 1024$.

The set of operating points $(\hat{P}_f(\Gamma), \hat{P}_d(\Gamma))$ obtained by estimating detection and false alarm rates for each of the classifiers produced by the NP design procedure define a curve. This curve is the *Estimated Performance Curve* (EPC). The result above implies that the EPC is an accurate reflection of the performance of a set of classifiers when a sufficient number of data points is available, irrespective of the number of features. This result is consistent with more general theories of generalization such as PAC-learning theory.

Other methods such as PALO [14] and mean-level hill climbing [15] do ensure that the ranking of classifiers takes into account error estimate variation. However, these methods *do not* address the problem of ensuring that the *optimal* classifier on the set has been found. This issue is discussed below.

2. The set of classifiers produced by NP design depends on the order of the sort of the likelihood estimates $\zeta(x)$. When feature set size increases, the sort order is corrupted by sampling error, and sub-optimal classifiers will be found. While the statistics of the sort error depend on the exact form of the underlying distributions $\theta_{j|H1}$ and $\theta_{j|H0}$, some general trends exist that can be intuitively explained.

The density estimates are histograms, and the likelihood ratio estimates depend on the histogram bin counts. When most of the mass of the distributions remains concentrated in a few bins as the feature dimension increases, bin counts are accurate, the likelihood ratio sort errors are localized, and the classifiers produced are optimal. However, when features are added, either irrelevant or with less-than perfect correlation, a randomization effect occurs; the same amount of data is spread out over more bins. Bins with small bin counts have a high probability of being sorted incorrectly, and errors in true performance appear on the order of the true probability mass found in these low-confidence bins. Ultimately, most bins contain only a few samples, and the estimated densities (bin counts) cannot be distinguished from samples from uniform class densities. All sorts will become equally likely. In this case the sort will effectively perform random assignment of labels, and the true performance of the classifiers returned by NP design approaches the $P_f = P_d$ curve.

Therefore, as the feature subset size increases, at some point the possible improved discrimination offered by additional features is defeated by errors in the sort procedure, and the true performance of the classifiers approaches the $P_f = P_d$ curve.

3. As the number of features increases, using the training data set to evaluate performance (a procedure yielding an ROC estimate we call Naive Estimated Performance Curve (NEPC)) will result in increasingly optimistic performance estimates. This is intuitively obvious; as the number of bins increase, ultimately every sample obtained from both the true and the false class will occupy an individual bin. The NP design procedure will label the bin with the corresponding class label, yielding perfect discrimination on the training set.

The above statistical effects combine to yield the effects illustrated graphically in Figure 1. As the number of features increase for a fixed data size, the true performance curve approaches the $P_d = P_f$ line, not because better performance is not possible, or the performance of a classifier cannot be accurately estimated, but simply because the probability of finding the optimal classifiers via a non-exhaustive search becomes negligible. Further, the Naive Estimated Performance Curve diverges from the Estimated Performance Curve, becoming increasingly optimistic as the feature subset size increases.

At this point we emphasize that the existence of a critical size has only been rigorously studied for Neyman-Pearson design. Different induction procedures (i.e. mechanisms for searching the set of classifiers) could exist for estimating the ROC curve, possibly with better error scaling properties at the cost of increased computation. None are presently known to the authors.

We should expect a threshold effect for any induction algorithm that cannot guarantee the ranking of classifiers on a subset, even when only a single operating point is being evaluated. This is especially true in backward selection procedures. In this case, the common assumption that irrelevant features can easily be detected [10] depends not only on the commonly recognized requirement of the induction algorithm returning a reliable estimate of the performance of a single classifier [14, 15]. The more subtle but crucial point is that the induction algorithm also has to provide some guarantee that each of the two classifier structures used to rank the two different subsets (one before removal of a feature, the other after removal of a feature) are optimal. At present, only exhaustive enumeration of the classifier space on each subset, which is completely impractical, satisfies this requirement for arbitrarily large feature subsets. None of the induction methods used by any of the other existing feature selection methods appears to meet this requirement. Most do not even guarantee the lesser requirement of consistency in ranking as the amount of data increases, as is guaranteed by the NP procedure.

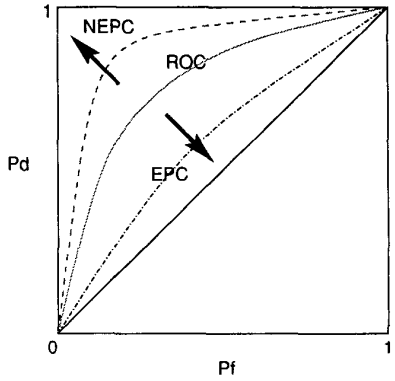

Figure 1. The classifiers yielding the best detection P_d for a given false alarm rate P_f determine the ROC curve. The Estimated Performance Curve (EPC) summarizes the estimated performance of the set of classifiers produced by the NP procedure on finite data. While EPC location is highly accurate given enough data, the set of classifiers produced by the NP procedure is generally suboptimal. As the number of features l increases, bin labels are increasingly assigned randomly and the EPC approaches the $P_f = P_d$ curve. The Naive Estimated Operating Performance Curve (NEPC) reuses the training data to estimate the performance, and over-estimates true performance. As l increases the classifiers memorize the training data and the NEPC approaches perfect performance.

3 Filter Structure of the ROC curves

In this section we show how partial orders can be defined on the feature subset, and how to use these orderings in conjunction with the statistical analysis above to prune

the feature subset space.

Note that in general, ROC curves for two different subsets will cross; which subset is optimal will depend on the desired operating point. Therefore, when performing feature subset selection based on ROC curves, one has to keep a number of subsets, each optimal over some range of P_f.

However, the power set does allow for some ordering. Consider two feature subsets Q_1 and Q_2. For some sets, an ordering called *uniform preferability* can be established on the ROC curves denoted $\succeq$, where $Q_1 \succeq Q_2$ if $\forall(Pd, Pf) \in \text{ROC}(Q_2) \exists(Pd', Pf') \in \text{ROC}(Q_1)$ where $Pd' \geq Pd$ and $Pf' \leq Pf$. Graphically, the ROC curve for feature subset Q_1 lies above the ROC curve for feature subset Q_2. We use two properties of $\succeq$. First, $\succeq$ always defines a filter structure [16] (directed set ordering) on the power set via subsumption, since $Q_1 \supseteq Q_2 \rightarrow Q_1 \succeq Q_2$, as any features not present in the subset will at worst be ignored by the optimal Bayesian fusion procedure. Second, given two subsets Q_1 and Q_2, the ROC of $Q_1 \cup Q_2$ has to be uniformly preferable to the convex hull of the two ROCs, which corresponds to sampling the classifier structures on Q_1 and Q_2.

Because of the two properties above, and the trends shown in Figure 1, it is possible to define a forward feature selection procedure that automatically estimates the size of the maximal feature subset that can be ranked. The procedure is shown graphically in Figure 2. By comparing the sets Q_1 and Q_2 of sizes l_1 and l_2 respectively, the convex hull of $\text{ROC}(Q_1) \cup \text{ROC}(Q_2)$ can be compared against the ROC estimate obtained on $Q_1 \cup Q_2$. When no statistically significant improvement occurs, all feature subsets containing $Q_1 \cup Q_2$ can be removed from consideration. With high probability, the ROC curve on these subsets cannot accurately be obtained, and will lie inferior to the convex hull that can be obtained from the two subsets individually.

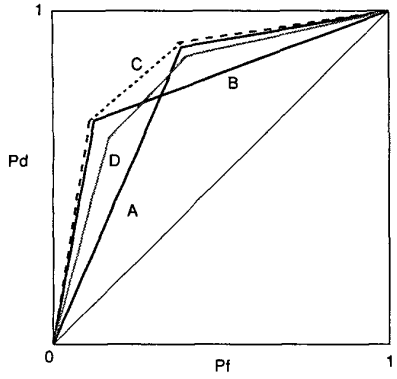

Figure 2. Illustration of the selection procedure: the ROC support classifiers on feature subsets Q_1 (ROC curve A) and Q_2 (ROC curve B) can be combined using sampling to yield an ROC curve C corresponding to the convex hull of $ROC(Q_1)$ and $ROC(Q_2)$. When curve $D = ROC(Q_1 \cup Q_2) \not\succeq C$ all feature subsets containing $Q_1 \cup Q_2$ can be pruned from the search.

To formalize the procedure, we first establish an absolute order $<$ on the feature power set. The ordering $<$ first sorts feature sets by increasing size, then all subsets of the same size are ordered using dictionary ordering of the feature indices. For example, $(x_3, x_4, x_8) < (x_3, x_5, x_8) < (x_1, x_2, x_4, x_5)$. Each subset Q can then be associated with a point in $[0, 1]$ which reflects the fraction f of the total number of subsets that will have been characterized by the search before Q is reached.

The basic algorithm is shown in Table 1. The algorithm first orders all feature subsets according to $<$. The EPC curve is calculated and stored for each subset, starting with the single element subsets. When $l \geq 2$, the current subset Q' is randomly split into two or more separate subsets. The EPC of Q' is compared to the convex hull of the EPCs of the smaller feature subsets. When the EPC of Q' on average lies more than some statistical buffer distance α_0 below the convex hull, Q' as well as all sets Q'' where $Q' \subseteq Q''$ are removed from consideration. The constant $\alpha_0 = \eta\sigma_{P_d|\hat{P}_d}$, where η is an appropriate constant, and $\sigma_{P_d|\hat{P}_d}$ is the standard deviation of the beta distribution (4) on the detection performance estimate. Once the set size reaches some fraction of N, an additional test removes all subsets on which classification performance is significantly worse than that obtained by the best subset obtained so far. This latter test removes most of the random label assignments from consideration in the latter stages of pruning, and improves the rate of convergence towards the end of the run.

3.1 Example: Synthetic Test Case

We illustrate the performance of the algorithm on the simple example introduced in Section 2. However, we add irrelevant features $x_2, x_3, x_4 \ldots x_{l-1}$, where for both classes the additional features are uniformly distributed, white variables. This problem is known to cause worst-case performance in classifier generalization [1]. The ROC curve for each of the subsets containing $(x_0, x_1, \ldots x_{l-1})$ where $l \geq 2$ is therefore equal to the ROC on subset (x_0, x_1). We expect the algorithm to rapidly prune the search space once sets of three or more elements are searched, since the extra features are irrelevant to the classification.

We implemented the algorithm using a dataset of 1500 samples from class 0 and 1200 samples from class 1, assuming a total of ten features. The datasets for the two classes were split into $n_0 = 1000$ and $n_1 = 800$ samples for training, and 500 and 400 samples for testing, respectively. This division allows for estimating the performance of any classifier to within 5% on the test data without using cross validation.

An exhaustive enumeration of all 1024 possible feature subsets was performed, and the EPC and NEPC for each subset were calculated. For each subset the average training set bias ϵ of the classifiers in detection performance, defined as the average vertical distance between the EPC and NEPC, was calculated. Figure 3 shows the 1024 ϵ values on a scatter plot as a function of the fraction f of the power set that has been characterized. An approximate mode is shown. This figure supports the contention that the EPC and the NEPC in general diverge as the number of features

Initialization
ActiveSets $\leftarrow (Q_1, Q_2, \ldots Q_{2^N}), Q_i < Q_{i+1}, Q_i \subseteq Q^*, i = 1, 2 \ldots 2^N - 1$
AreaMax $\leftarrow 0$
Execution
for each Q in ActiveSets
Calculate and store $EPC(Q)$
if($||Q|| > 1$)
Randomly select subsets $Q'_1, Q'_2, \ldots Q'_k$ such that $Q'_1 \cup Q'_2 \cup \ldots Q'_k = Q, Q'_i \subset Q, i = 1, 2, \ldots k$
Calculate convex hull $C(Q)$ of $\{EPC(Q'_1), EPC(Q'_2), \ldots EPC(Q'_k)\}$

$$\epsilon = \int_0^1 \max\{0, C(Q)(P_f) - EPC(Q)(P_f) - \alpha_0\}\, dP_f$$

$$\text{Area}(Q) = \int_0^1 EPC(Q)dx - 0.5$$

if (Area(Q) > AreaMax) AreaMax $\leftarrow$ Area(Q)
if ((((Area(Q) $< \alpha_1$AreaMax)&&($||Q|| > 0.3N$))$||(\epsilon > \alpha_2)$)
for each Q' in (ActiveSets, $Q \subseteq Q'$) ActiveSets $\leftarrow$ ActiveSets$\backslash\{Q'\}$

Table 1. Algorithm for forward filtering of the feature power set using the ROC estimates.

increase.

This result suggests an additional approach to pruning the power set, namely by monitoring and pruning subsets when the average bias exceeds a threshold. While this approach appears to be extremely reliable in practice and will bear investigation in the future, we do not yet have sufficient analytical results to rigorously define this threshold. In contrast, standard statistical tests using the properties of the beta distribution can be used to determine valid thresholds for α_0 and α_1 in the algorithm of Table 1.

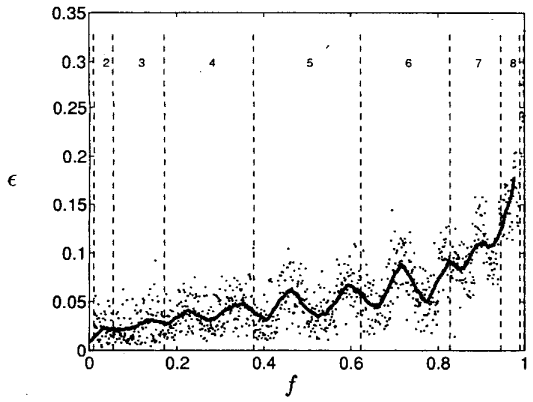

Figure 3. Bias ϵ in P_d, defined as average distance between NEPC and EPC as a function of the fraction f of the feature space characterized. Results for individual subsets are indicated by points; the curve shows the approximate mode. Regions separated by dashed vertical bars correspond to feature subsets with a certain number of elements.

Figure 4 shows the fraction C of the power set that has been characterized as a function of the current search position f. A subset is considered characterized when either a full NP procedure was performed for the subset, or when the set was pruned (filtered) from further consideration due to the EPC falling significantly below the convex hull of the EPCs of its subsets. Due to the high accuracy of the EPC location, the randomizing effect of the features $x_2 \ldots x_9$ is soon detected when three element subsets are considered. Due to the small size of these subsets, they are included in a large number of the larger subsets, and major parts of the power set are rapidly pruned. In this example, the procedure indicates that characterizing subsets with more than four features yields little additional return (note that adding more features will not change this critical size).

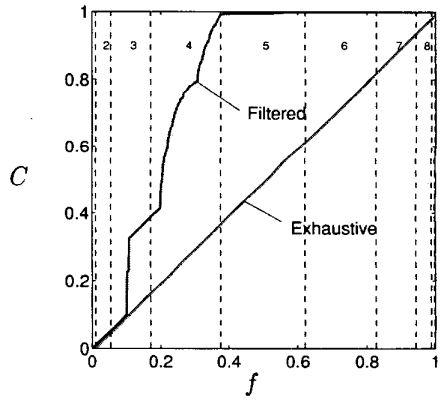

Figure 4. Curve (labeled "Filtered") showing the fraction of the feature power set characterized (C), either via direct calculation or using filter structure elimination, as function of fraction of feature power set f on which NP estimation has been performed. The curve labeled "Exhaustive" is the diagonal and corresponds to an exhaustive search.

Figure 5 shows the cumulative fraction ν of the power set on which a full NP design has been performed, as a function of the current search set f in the power set. The curve shows that once approximately 40% of the feature subsets have been characterized, most of the larger subsets have been eliminated from consideration. In terms of ac-

tual processor computation, even better results are obtained since the number of computations necessary to estimate the ROC of a subset scales exponentially with the size of the subset and hence with f.

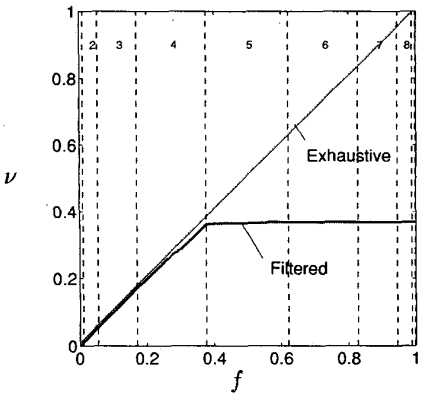

Figure 5. Curve (labeled "Filtered") shows the cumulative fraction of the power set on which a full NP design is performed, as a function of the fraction f of the feature power set that has been characterized. The curve labeled "Exhaustive" is the diagonal referencing a standard exhaustive search. Regions separated by vertical bars correspond to feature subsets with a certain number of elements.

3.2 Complexity and Implementation

The major computational effort involved in performing Neyman-Pearson design on a subset of l features is due to the sorting of the 2^l likelihood ratios of the histogram bins. Once sorted, calculation of the classifier label assignments, EPC, NEPC, and convex hull, is approximately linear in the length of the histogram. The cost of the NP design is therefore of the order $2^l[l \log 2 + c]$ where c is a constant. There are 2^N feature subsets to characterize. Using integer arithmetic, current technology (10^9 integer operations per second) allows for exhaustive characterization to be performed for feature subsets of approximately 20 features within 24 hours of computation.

The pruning of the subspace allows for N to be significantly increased, depending on the critical subset size beyond which the NP design procedure breaks down. There are ${}_N C_l$ subsets of length l; when l is small relative to the total number of features N, ${}_N C_l \simeq N^l/l!$. Therefore, a rough estimate of the calculation complexity of computing the ROC by exhaustive search through all subsets of size $l \ll N$ is

$$N^l 2^l/(l-1)! \tag{5}$$

By pruning the feature power set to only consider subsets of size up to $l \ll N$, the computational problem is therefore changed from an exponential, to an admittedly high-order polynomial problem. When the critical subset size corresponds to $l = 5$ elements, N is limited to approximately 32 features. While this reduction represents an improvement in the size of N of approximately 50%, in practice the size of N that can be considered is still relatively modest. In practice, therefore, when a large number of features exist, the pruning approach has to be combined with some heuristics that partition the features into subsets of manageable size, on each of which the pruning approach can be performed. One such approach is discussed below. However, we note that significant scope exists here for extending existing methods, especially backward approaches such as [7, 10], to exploit the fact that only feature subsets below some critical limit can be searched in a statistically reliable fashion.

4 Query Modification and Text Classification

Query modification arises in the context of building category specific interfaces to databases. Within the interface, user queries are automatically augmented by search terms and operators that effectively restrict query results to a relevant category within the database.

The query modification technique is especially valuable in customizing existing databases, or for integration of multiple distributed databases, as occurs when building category specific metasearch engines. A metasearch engine forwards a query to a large number of secondary search engines, then collates and ranks the results centrally for presentation to the user. By adding query modifications before forwarding the user query to the secondary search engines, fewer irrelevant results are returned to the metasearch engine, which reduces the bandwidth consumption and computational load on the meta engine site. These advantages are especially significant in next-generation search services such as *Inquirus* and *Inquirus 2*, that download the web pages associated with queries returned to the metasearch engine [17, 18, 19, 6].

In a metasearch framework it is important to predict the recall and false alarm rate implied by a query modifier for a particular search engine. This information can be used to allocate server loads, predict the quality and expected quantity of results produced by each search engine, and also order results for the user [3, 4, 5].

Our query modifications use the fact that most search engines support Boolean search. Formally, our query modifications are generated by conjoining the user query A with a disjunction of conjunctive modifiers generated by feature bit strings

$$A \rightarrow A \wedge (x_{i_1} x_{i_2} \ldots x_{i_l} \vee x_{i_{l+1}} x_{i_{l+2}} \ldots x_{l+l'} \vee \ldots) \tag{6}$$

where $x_{i_k} \in \{x_0^*, \ldots x_{N-1}^*, \bar{x}_0^*, \ldots \bar{x}_{N-1}^*\}$, and x_i^* is a document feature.

The feature extraction procedure introduced in the first part of the paper offers a simple and elegant method for computing suitable Boolean query modifications. We note that the binary expression of the true class decision region $\mathcal{L}(\Gamma^k)_1$ of each ROC support classifier Γ^k can be encoded as a modifier of the desired Boolean form. The classifier Γ^k on a subset of l features will correspond to a query modification consisting of k terms, each involving l features.

Further, conditioned on the query A, the probability of retrieving a document in the desired category with maximal recall at a given rate of accuracy is guaranteed by this modifier.

As a concrete example, assume that the class statistics are described by the densities of the simple example problem introduced in Section 3.1. Consider asking for a modifier where 65% recall is expected at 40% false positive retrieval. The true class decision region of the classifier $\Gamma^2(x)$ expressed as a modifier yields

$$A \rightarrow A \wedge (\bar{x}_1\bar{x}_0 \vee \bar{x}_1 x_0) \quad (7)$$

In principle, therefore, given a set of document features and a labeled data set, query modifiers are extracted as follows. First, we use the feature selection procedure to obtain the EPCs of all subsets that can be well characterized (i.e. the EPC is likely close to the ROC). The convex hull of all the EPCs is then constructed. Each point on the resulting curve corresponds to a particular classifier, and hence, query modification. When a query modification is requested by the niche search engine, the query modification closest to the desired operating point is used.

Practical deployment of query modifiers is a major area of investigation by itself, which we will not explore in detail (see [20, 6, 19, 21] for a discussion). However, one major problem is that secondary search engines impose limits on operators and query lengths. We now address this problem in our framework.

In off-line applications, procedures such as the Quine-McCluskey method, or approximations thereof, can be used to simplify modifications by applying Boolean logic [22, 23]. In our example, the expression (7) can be simplified to

$$A \rightarrow A \wedge (\bar{x}_1) \quad (8)$$

Unfortunately, minimizing a lengthy expression of the form (6) is a hard problem. Therefore, when query modifications have to be generated online, such as when the category is defined by tracking a user's online searches, a different approach must be used.

Consider again the conversion of the ROC support classifiers into query modifiers. The number of terms in the query is equal to the ranking j of the classifier Γ^j. Therefore, without Boolean simplification, on the same feature subset classifiers with lower values of P_f will have fewer terms, and shorter lengths. We can therefore group the classifiers according to length, and for each group we generate the convex hull of their operating points. When a search engine requires a query modifier, the shortest modifier closest to the required operating point from each of the groups is submitted. This approach yields highly discriminatory query modifiers with low P_f and low P_d (operating at the bottom left corner of the ROC).

To find more inclusive modifiers, i.e. short modifiers with high P_d, but also higher P_f, the modifiers related to binary encoding of the false class decision region are extracted, and negated to yield a modifier suited to the true class. These modifiers correspond to classifiers operating at the upper right corner of the ROC.

In systems such as *Inquirus* and *Inquirus 2*, the pages that correspond to the URLs returned by the underlying search engines are downloaded in their entirety, and analyzed to improve ranking for the user. In these systems a two stage approach is therefore possible, whereby short query modifiers are used to provide a first pool of relatively high-quality full-text candidate documents, to which more computationally complex classifiers can be applied centrally. In this way, operating points for intermediate values of P_f can be achieved, while retaining part of the benefits of using query modifiers.

4.1 Application: Query Modifiers for Web Search

In this section we illustrate the construction of a set of query modifications for detecting the *"Call For Papers"* sub-section from within conference announcement sites. This detection problem is challenging, since a large number of pages with similar features appear at the same site. As a result, high discrimination performance can be achieved only by taking into account joint statistics of the features. For example, we note that announcements concerning the venue usually do not appear on the call for papers page, unless the call for papers page and the main conference page coincide.

Using a combination of manual and automatic retrieval, a total of 2701 pages related to conferences were retrieved from the web. The pages were manually classified into 2269 false class, and 432 positive class pages. To increase the set of secondary services we can query, we restrict ourselves to searching for text features. The only structural information considered is an indication of whether the text feature occurs in the title of the page or not.

A dictionary was constructed for all the words, bi-grams and tri-grams contained in the documents. The number of documents in each class in which a phrase appears at least once was recorded. The phrases that did not appear in more than 10% of either the true, or the false class documents, were ignored. The phrases were then ranked according to the ratio of the number of times the phrase appears in the true class, to the number of times it appears in the false class.

The top 24 features were used to select feature subsets, calculate EPC curves, and extract query modifiers and classifiers. These features are shown in Table 2. Figure 7 shows the performance of the ROC pruning algorithm on this dataset. Despite the relatively large data set, the algorithm indicates that estimating optimal classifiers for subsets with more than 4 features cannot reliably be performed. Further, the curve suggests that most of the classification performance can be obtained by modeling interactions amongst three or fewer features.

On the subsets that were characterized, we calculated query modifiers of up to six disjunctive terms. We separated the query modifiers into groups according to the number l of features used in the modifier. For each group the convex hull of the operating points was calculated[1]. The result-

[1]We use a modified convex hull algorithm that accounts for statistical variation described by the beta function in deciding which points to include.

No	Value	No	Value	No	Value	No	Value
0	T.cfp	1	F.accepted papers	2	F.invited to submit	3	F.notification of acceptance
4	F.camera ready	5	F.important dates	6	T.call for papers	7	F.notification of
8	T.papers	9	F.submission of	10	F.cfp	11	F.call for papers
12	F.organizing committee	13	F.deadline for	14	F.the proceedings	15	F.program committee
16	F.of computer science	17	F.abstracts	18	F.phone fax	19	F.chair
20	F.tutorials	21	F.further information	22	F.of interest	23	F.held in

Table 2. Table of phrases used in creating top ranked query modifiers. A "T" in front of the modifier indicates the phrase appears in the title; an "F" denotes the phrase occurs in the body text.

No	Pf	Pd	Pd/Pf	Value
1	0.002	0.153	76.4	$+x_0$
2	0.006	0.412	74.5	$+x_4$
3	0.036	0.759	20.9	$+x_{11}$
4	0.002	0.236	118.1	$+x_3\bar{x}_{17}$
5	0.005	0.440	94.7	$+x_0\bar{x}_3 + x_0x_3 + \bar{x}_0x_3$
6	0.006	0.505	91.3	$+x_0\bar{x}_4 + x_0x_4 + \bar{x}_0x_4$
7	0.007	0.565	77.5	$+x_3x_4 + x_3\bar{x}_4 + \bar{x}_3x_4$
8	0.037	0.852	22.9	$+x_3x_{11} + x_3\bar{x}_{11} + \bar{x}_3x_{11}$
9	0.042	0.894	21.4	$+x_5x_{11} + x_5\bar{x}_{11} + \bar{x}_5x_{11}$
10	0.091	0.917	10.1	$+x_{11}x_{15} + x_{11}\bar{x}_{15} + \bar{x}_{11}x_{15}$
11	0.002	0.361	180.6	$+x_0x_3x_{17} + \bar{x}_0x_3\bar{x}_{17} + x_0x_3\bar{x}_{17} + x_0\bar{x}_3\bar{x}_{17} + x_0\bar{x}_3x_{17}$
12	0.007	0.602	82.5	$+x_1x_3x_4 + x_1\bar{x}_3x_4 + \bar{x}_1x_3\bar{x}_4 + \bar{x}_1x_3x_4 + x_1\bar{x}_3\bar{x}_4 + \bar{x}_1\bar{x}_3x_4$
13	0.036	0.843	23.2	$+x_7\bar{x}_{11}x_{15} + x_7x_{11}\bar{x}_{15} + x_7x_{11}x_{15} + \bar{x}_7x_{11}x_{15} + \bar{x}_7x_{11}\bar{x}_{15}$
14	0.037	0.852	22.9	$+x_3x_{11}\bar{x}_{23} + x_3\bar{x}_{11}x_{23} + x_3\bar{x}_{11}\bar{x}_{23} + x_3x_{11}x_{23} + \bar{x}_3x_{11}x_{23} + \bar{x}_3x_{11}\bar{x}_{23}$
15	0.038	0.861	22.6	$+x_5x_{11}\bar{x}_{15} + x_5x_{11}x_{15} + x_5\bar{x}_{11}x_{15} + \bar{x}_5x_{11}x_{15} + \bar{x}_5x_{11}\bar{x}_{15}$
16	0.040	0.880	22.0	$+x_5x_{11}\bar{x}_{20} + x_5x_{11}x_{20} + x_5\bar{x}_{11}\bar{x}_{20} + \bar{x}_5x_{11}x_{20} + \bar{x}_5x_{11}\bar{x}_{20}$
17	0.042	0.894	21.4	$+x_2x_5x_{11} + x_2\bar{x}_5x_{11} + x_2x_5\bar{x}_{11} + \bar{x}_2x_5x_{11} + \bar{x}_2x_5\bar{x}_{11} + \bar{x}_2\bar{x}_5x_{11}$
18	0.091	0.917	10.1	$+x_{11}x_{15}\bar{x}_{23} + x_{11}\bar{x}_{15}x_{23} + x_{11}x_{15}x_{23} + x_{11}\bar{x}_{15}\bar{x}_{23} + \bar{x}_{11}x_{15}x_{23} + \bar{x}_{11}x_{15}\bar{x}_{23}$

Table 3. Selected query modifiers

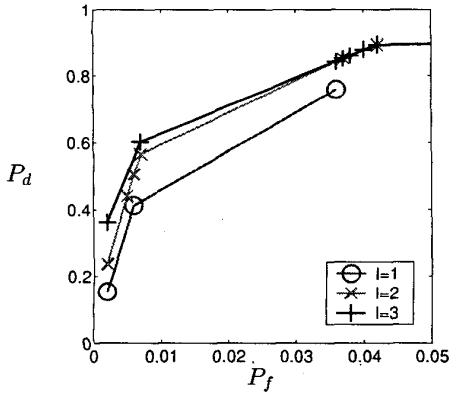

Figure 6. Convex hull and operating points achieved by "Call for Papers" query modifier, grouped by the number l of features in each query modification.

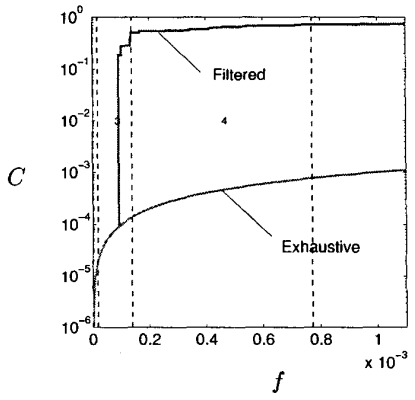

Figure 7. Forward pruning of the feature set for 24 features on the "Call for Papers" problem. The vertical axis shows the fraction of the power set of 2^{24} subsets characterized or eliminated, as a function of the position of the search position f as a fraction of the search space. Note that a logarithmic scale is used due to the large dimension of the space.

ing query modifiers of the convex hull have performance as shown in Figure 6, while the corresponding symbolic query modifiers are shown in Table 3. The query modifiers are not simplified using Boolean rules, so the order in which the histogram bins, and hence the modifier terms, are added, can clearly be seen.

At first glance, the large number of negated elements corresponding to clearly relevant features may appear somewhat counter intuitive. The negation of terms is required in our problem due to the considerable similarity of the false and the true class used in building the modifiers. A better perspective on the role of the negative modifiers results from viewing different terms of a modifier as each describing non-overlapping sub-clusters of relevant documents. For example, consider the tenth modifier listed: $+x_{11}x_{15}+x_{11}\bar{x}_{15}+\bar{x}_{11}x_{15}$. This modifier can be written as $+x_{11}+\bar{x}_{11}x_{15}$. The first term makes a cut that captures all documents containing x_{11}. The second term can be viewed as trying to correct the first assignment by describing a secondary cluster of relevant documents, best described by the criterion x_{15}, contained in the false class of the first feature (x_{11}).

5 Conclusions

Statistically valid feature selection is a computationally intensive problem that requires summarization of the performance of classifiers on each element of the feature power set. We introduced a method for reducing the problem from an exponential to a polynomial problem, while allowing for flexibility in selection of the final operating point. Our approach integrates a number of results. First, we use the fact that the Neyman-Pearson design procedure yields classifiers that have increasingly random bin labels as the feature subset size increases. This randomization results in a shift in true performance toward the $P_d = P_f$ diagonal. Second, we use the directed set relationships satisfied by the elements of the feature power set, and also by the ROC curves on the subsets, to detect when the Neyman-Pearson design procedure breaks down. As a result, characterization can be restricted to statistically valid subsets with relatively small sizes, where computational cost is small.

Our algorithm provides a new method for performing statistically valid dimensionality reduction in discrete spaces. We make no assumptions, explicit or implicit, about the data distributions. The use of ROC curves also offers some protection against changes and uncertainty in the *a-priori* class probabilities. Instead of a single classifier, a set of classifiers with monotonic performance guarantees are produced that can easily be changed by varying a single scalar parameter. As a result, the approach should be particularly useful on the web, where text is often represented using discrete alphabets (such as by thresholding TFIDF values), and it is difficult to accurately model the false class probability. We illustrated one such use, that of extracting query modifiers for metasearch.

Our algorithm at present uses an exhaustive characterization of the subsets below the critical size limit. The approach can be used directly for constructing classifiers and characterizing groups of features of useful size (around thirty features). However, it should be possible to incorporate components of our approach into more sophisticated search approaches that sample the power set, and thereby increase the size of the feature subsets that can be processed.

Existing approaches would in turn benefit significantly from the restriction of the search space resulting from estimating the critical size limit which likely exists for each induction approach. Further, in construction of backward selection procedures, we note that care should be taken not to eliminate features from consideration by ranking performance when the procedure is not statistically warranted.

References

[1] G. H. John, R. Kohavi, and K. Pfleger, "Irrelevant features and the subset selection problem," in *Machine Learning: Proceedings of the Eleventh International Conference*, 1994.

[2] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artificial Intelligence*, pp. 245–271, 1997.

[3] L. Gravano, H. Garcia-Molina, and A. Tomasic, "The effectiveness of GlOSS for the text database discovery problem," in *Proc. ACM SIGMOD*, 1994.

[4] L. Gravano, H. Garcia-Molina, and A. Tomasic, "Precision and recall of GlOSS estimators for database discovery," in *Proceedings of the 3rd International Conference on Parallel and Distributed Information Systems (PDIS'94)*, September 1994.

[5] L. Gravano, K. Chang, H. Garcia-Molina, C. Lagoze, and A. Paepcke, "STARTS: Stanford protocol proposal for Internet retrieval and search." Digital Library Project, 1997.

[6] E. J. Glover, S. Lawrence, W. P. Birmingham, and C. L. Giles, "Architecture of a metasearch engine that supports user information needs," in *Proc. CIKM 1999*, pp. 210–216, 1999.

[7] D. Koller and M. Sahami, "Toward optimal feature selection," in *Proc. 13th International Conference on Machine Learning*, pp. 284–292, Morgan Kaufmann, 1996.

[8] R. Caruana and D. Freitag, "Greedy attribute selection," in *Proc. 11th International Conference on Machine Learning*, pp. 28–36, Morgan Kaufmann, 1994.

[9] H. Almuallim and T. G. Dietterich, "Learning with many irrelevant features," in *Proceedings of the Ninth National Conference on Artificial Intelligence*, pp. 547– 552, 1991.

[10] P. Langley and S. Sage, "Oblivious decision trees and abstract cases," in *Working Notes of the AAAI-94 Workshop on Case-Based Reasoning*, AAAI Press, 1994.

[11] H. L. Van Trees, *Detection Estimation and Modulation Theory*, vol. 1-3. Wiley and Sons, 1971.

[12] R. O. Duda and P. E. Hart, *Pattern Recognition and Scene Analysis*. New York: John Wiley & Sons, 1973. ISBN 0-471-22361-1.

[13] F. Coetzee, S. Lawrence, and C. L. Giles, "Bayesian classification and feature selection from finite data sets," in *Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI-2000)*, (Stanford, CA), pp. 89–97, June 30–July 3 2000.

[14] R. Greiner, "PALO: A probabilistic hill-climbing algorithm," *Artificial Intelligence*, vol. 83, no. 1–2, 1996.

[15] R. Kohavi, "Feature subset selection as search with probabilistic estimates," in *Proc. AAAI Fall Symposium on Relevance*, 1994.

[16] J. R. Munkres, *Topology: A First Course*. Prentice-Hall, 1975. ISBN 0-13-925495-1.

[17] S. Lawrence and C. L. Giles, "Context and page analysis for improved web search," *IEEE Internet Computing*, vol. 2, no. 4, pp. 38–46, 1998.

[18] S. Lawrence and C. L. Giles, "Inquirus, the NECI meta search engine," in *Seventh International World Wide Web Conference*, (Brisbane, Australia), pp. 95–105, 1998.

[19] E. J. Glover, S. Lawrence, M. D. Gordon, W. P. Birmingham, and C. L. Giles, "Web search – your way," *Communications of the ACM*, 1999. accepted for publication.

[20] M. Mitra, A. Singhal, and C. Buckley, "Improving automatic query expansion," in *Proc. SIGIR98*, (Melbourne (AU)), ACM, 1998.

[21] K.-U. Sattler and M. Hoding, "Adapter generation for extracting and querying data from web sources," in *Proc. WebDB*, pp. 49–54, 1999.

[22] E. McCluskey, "Minimization of boolean functions,," *Bell Syst. Tech. J.*, vol. 35, pp. 1417–1444, 1956.

[23] R. Brayton, G. Hachtel, C. McMullen, and A. Sangiovanni-Vincentelli, *Logic minimization algorithms for VLSI minimization*. Kluwer, 1985.