BIOMARKER DISCOVERY AND VALIDATION FOR PROTEOMICS AND GENOMICS: MODELING AND SYSTEMATIC ANALYSIS

A Dissertation

by

ESMAEIL ATASHPAZGARGARI

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Ulisses de Mendonca Braga-Neto
Co-Chair of Committee,	Edward R. Dougherty
Committee Members,	Byung-Jun Yoon
	Jean-Francois Chamberland
	Ivan Ivanov
Department Head,	Chanan Singh

August 2014

Major Subject: Electrical Engineering

ABSTRACT

Discovery and validation of protein biomarkers with high specificity is the main challenge of current proteomics studies. Different mass spectrometry models are used as shotgun tools for discovery of biomarkers which is usually done on a small number of samples.

In the discovery phase, feature selection plays a key role. The first part of this work focuses on the feature selection problem and proposes a new Branch and Bound algorithm based on U-curve assumption. The U-curve branch-and-bound algorithm (UBB) for optimization was introduced recently by Barrera and collaborators. In this work we introduce an improved algorithm (IUBB) for finding the optimal set of features based on the U-curve assumption. The results for a set of U-curve problems, generated from a cost model, show that the IUBB algorithm makes fewer evaluations and is more robust than the original UBB algorithm. The two algorithms are also compared in finding the optimal features of a real classification problem designed using the data model. The results show that IUBB outperforms UBB in finding the optimal feature sets. On the other hand, the result indicate that the performance of the error estimator is crucial to the success of the feature selection algorithm.

To study the effect of error estimation methods, in the next section of the work, we study the effect of the complexity of the decision boundary on the performance of error estimation methods. First, a model is developed which quantifies the complexity of a classification problem purely in terms of the geometry of the decision boundary, without relying on the Bayes error. Then, this model is used in a simulation study to analyze the bias and root-mean-square error (RMS) of a few widely used error estimation methods relative to the complexity of the decision boundary. The

results show that all the estimation methods lose accuracy as complexity increases.

Validation of a set of selected biomarkers from a list of candidates is an important stage in the biomarker identification pipeline and is the focus of the the next section of this work. This section analyzes the Selected Reaction Monitoring (SRM) pipeline in a systematic fashion, by modelling the main stages of the biomarker validation process. The proposed models for SRM and protein mixture are then used to study the effect of different parameters on the final performance of biomarker validation. We focus on the sensitivity of the SRM pipeline to the working parameters, in order to identify the bottlenecks where time and energy should be spent in designing the experiment.

DEDICATION

To Sahba

ACKNOWLEDGEMENTS

I would like to thank my advisors Dr. Ulisses M. Braga-Neto and Dr. Edward R. Dougherty. I am especially grateful to Dr. Braga-Neto for his constant support, guidance and encouragement, and for showing me that a delicate balance between macro-leading and micro-helping is the golden key to fostering a student's research. His attention to details and high working standards has made a lasting impression on my life. He is always open to ideas and discussion, and has helped me to believe more in myself.

My special thanks go to Dr. Dougherty for his guidance throughout my research. Besides showing me how to walk through the path I have chosen, he taught me the importance of choosing the right path. Working with him, I learned not only how to think about better solutions to problems, but also how to ask better questions. His farsighted scientific vision and emphasis on true scientific research are important lessons I have learned from him.

I am also grateful to my committee members, Dr. Byung-Jun Yoon, Dr. Jean-Francois Chamberland and Dr. Ivan Ivanov for their advice and support.

I would like to thank Dr. Charles D. Johnson for his support during my two-year collaboration with Texas A&M AgriLife Genomics and Bioinformatics Service.

Thanks also go to my friends in the GSP lab for great discussions we had on different topics.

I dedicate this dissertation to Sahba, my lovely wife. My debt to her is beyond words. Without her support, this work would have never been completed.

Thanks to my parents, Mousa and Rabieh, and my sister, Masoumeh, for being a generous source of pure love, support, and encouragement throughout my life.

NOMENCLATURE

QQQ	Triple Quadrupole
MS	Mass Spectrometry
SRM	Selected Reaction Monitoring
PSA	Prostate Specific Antigen
CID	Collision Induced Dissociation
PTP	Proteotypic Peptides
TPR	True Positive Rate
QDA	Quadratic Discriminant Analysis
3NN	3-nearest-neighbor
NNet	Two-layer Neural Network
BB	Branch and Bound
UBB	U-curve Branch and Bound

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION*

The identification of biomarkers is a major goal of biomedicine in this century [1], and proteomics using different mass spectrometry (MS) tools has played a key role in this area. One well-known example of peptide biomarker is Prostate Specific Antigen (PSA), which is a marker for early diagnosis of prostate cancer in men. The PSA test is an FDA-approved serum or plasma-based population screening tool, but has very low specificity, resulting in $750 million annual cost for unnecessary medical follow-up. The lack of biomarkers with high specificity shows how challenging the problem of proteomic biomarker identification is and the need for sensitive and accurate instruments, powerful techniques, and careful analysis of proteomics data.

One of the important challenges of biomarker discovery is identification of low-abundance biomarkers. Abundant biomarkers are easy to detect and quantify, but these have already been identified for the most part. The current emphasis is therefore on the discovery of low-abundance biomarkers [1]. Figure 1.1 displays the biomarker identification pipeline and the two main stages in this process, the *discovery* and *validation/qualification* phases. The global discovery phase is done on a small number of samples and then a larger number of samples is used for the validation of potential biomarkers, before going to clinical application [1].

In the discovery phase the selection of best protein biomarkers from a set of proteins is very important. This emphasizes on the role of feature section algorithm. In the first part of this work, in section 2, a feature selection algorithm is proposed for

*Parts of this section are reprinted with permission from "Relationship between the accuracy of classifier error estimation and complexity of decision boundary" by Esmaeil Atashpaz-Gargari, Chao Sima, Ulisses M Braga-Neto, Edward R Dougherty, 2012, *Pattern Recognition*, vol. 46, no. 5,

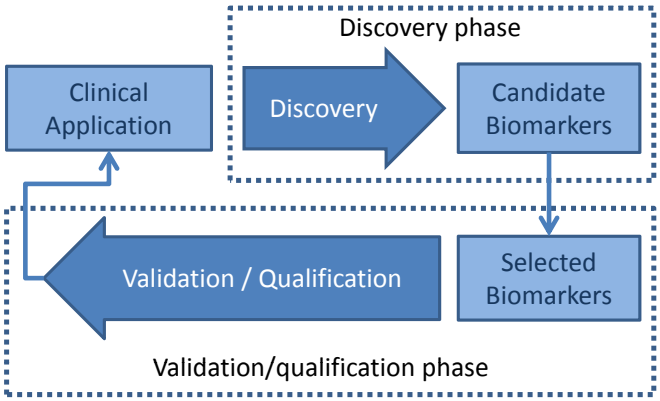


Figure 1.1: Two main stages of biomarker development pipeline. The discovery phase requires MS experiments with high resolution and short duty cycles and typically involves small numbers of samples. Selected biomarkers from the discovery step are validated in the next stage before moving on to further analysis in clinical studies [1] .

selecting global best feature set. The performance of the feature selection algorithms depends heavily on the accuracy of the error estimation methods. Then, in the next chapter, in section 3, we introduce a measure for quantifying the complexity of the decision boundary and use it to study the performance of the widely used error estimation methods as the complexity changes. Modeling the validation pipeline is the focus of the final part of this work, section 4. We study the triple quadrupole based Selected Reaction Monitoring (SRM) pipeline and model the protein quantification process through SRM. Then we use this model to study the effect of different parameters on the performance of the pipeline.

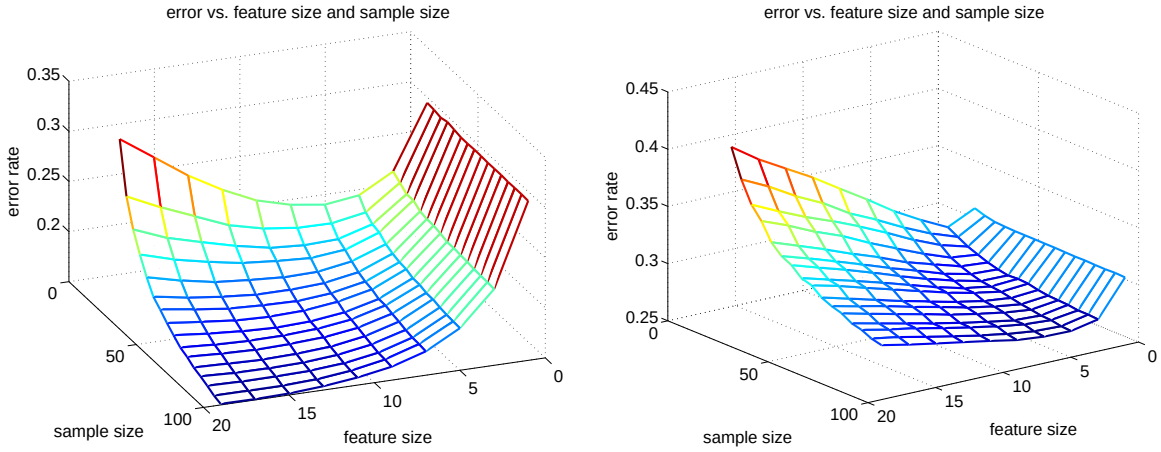
1.1 Biomarker Discovery: Peaking Phenomenon, Feature Selection and Error Estimation

1.1.1 Feature Selection

Given the joint feature label distribution, increasing the number of features decreases the classification error. However, when the classifier is designed using sample

data and the number of samples is small, the classification error does not monotonically decrease. Increasing the number of features used to design the classifier, with a fixed number of samples, makes the expected error of the designed classifier decrease and then increase. This is known as the *peaking phenomenon*, which was first studied in [2].

Figures 1.2(a) and (b) show peaking phenomenon for the Linear Discriminant Analysis (LDA) classification rule. In Figure 1.2(a) the features are slightly correlated. In this case, peaking occurs earlier (i.e., for a smaller number of features) or later depending on the sample size. For example, at sample size 30, peaking occurs with about 6 features, but when sample size increases to 100, peaking occurs at a larger feature size. In Figure 1.2(b) the features are highly correlated. As we see in this case, even for a large sample size, peaking occurs early. Figure 1.2 is based on the work in [3].



(a) (b)

Figure 1.2: Peaking phenomenon. (a) Slightly correlated features. $\rho = 0.125$. (b) Highly correlated features. $\rho = 0.5$.

Feature selection is the problem of finding an optimal subset of a finite set of features that minimizes a cost (criterion) function, often taken to be the classification

error [4]. Determining the optimal set of features can be a complicated task as for a problem with n features, there are $\binom{n}{d}$ possible feature sets of size d and 2^n possible total possible feature sets.

If the criterion function used for feature selection is independent of the classification rule, the method is said to be a *filter approach*. Otherwise a feature selection method is called *wrapper*. Due to the high complexity of feature selection many sub-optimal algorithms have been proposed. The following is the brief description of some of the widely used sub-optimal feature selection methods.

- *Best Individual d Features*: The criterion function is evaluated for each individual feature X_i and the best d features are selected.

- *Sequential Forward Search*: At each step the criterion function is evaluated for for all features and the best feature is selected. This is repeated until d features are selected.

- *Sequential Backward Search*: This algorithm starts with the selection of all the features. At each step the criterion function is evaluated removing each of the individual features. The feature that minimizes the criterion drop is removed from the feature set and the algorithm continues until d features are remaining.

- *Generalized Sequential Forward Search*: At each step of the algorithm, the combination of r features that are not in the current feature set are evaluated. The group for which the addition of the features to the group of the selected features maximizes the criterion value, is selected and added to the group. When $r = 1$ this algorithm is the same as Sequential Forward Search.

- *Generalized Sequential Backward Search*: At each step of the algorithm, the combination of r features that are in the current feature set are evaluated.

The group for which the removal of its features from the group of the selected features minimizes the criterion value drop is eliminated from the set of selected features. When $r = 1$ this algorithm is the same as Sequential Backward Search.

- *Plus-l Take-r Search*: Depending on the values of l and r this can be both bottom-up and top-down search. If $l > r$ at each step, using sequential forward search, l features are added and then using the sequential backward search, r features are removed. If $r > l$ at each step, using sequential backward search, r features are removed and then using the sequential forward search, l features are added.

- *Generalized Plus-l Take-r Sequential Search*: This method uses generalized sequential forward search and generalized sequential backward search instead of sequential forward search and sequential backward search, respectively.

- *Floating Search*: This the Plus-l Take-r search method, where the values of l and r are allowed to vary.

Besides the sub-optimal feature selection methods, some algorithms have been proposed that use heuristics to attempt to find the optimal feature set in fewer evaluations than exhaustive search. Among these are feature selection algorithms based on the well-known Branch-and-Bound (BB) paradigm for discrete and combinatorial optimization [5, 6]. A BB algorithm uses some property of the criterion function, such as monotonicity, to accomplish a systematic enumeration of the features sets in the form of a tree. At each step of the algorithm, the *Bound (B)* is defined as the cost of the best feature set found until that step. If the cost c of a node is smaller than the bound, its successor nodes are explored further and B is updated. If $c > B$

for a node, then the successors of that node are discarded. If the tree is organized in such a way that large sections of it can be pruned en masse, then the BB algorithm is successful. Different improvements have been proposed to enhance the performance of the basic BB algorithm [7]. Yu and Yuan [8] suggest avoiding the evaluation of intermediate single-branching nodes by obtaining a "minimum search tree." Also ordering the nodes in the tree based on the significance of the features is used in some of the variants of the BB algorithm [7]. To avoid evaluation of the cost, some algorithms try to estimate it and use the estimated value to construct the tree [9].

Now, due to the peaking phenomenon, discussed previously, the classification error is likely to display a U-shaped behavior along a chain of increasing nested feature sets. In this case, the original BB algorithm, or its variants mentioned previously, are not suitable, as all of these algorithms assume that the cost function is monotonic. Hence, the solution found by these algorithms will not necessarily be the global best possible feature set. The original proposal of a feature selection algorithm based on a U-shaped cost function was made by Ris and colleagues in [10]. They called this algorithm the U-curve Brand-and-Bound (UBB) feature selection algorithm. The purpose of the first part of the work, in section 2 is to propose and analyze a fast algorithm to implement the UBB method, which outperforms the original algorithm.

In small sample feature selection problems, the performance of the feature selection depends heavily on the accuracy of the error estimation algorithm. The focus of the section 3 of this dissertation will be the analysis of the performance of the error estimation methods. In this section, we will study the effect of the complexity of the decision boundary on some of the widely used error estimators.

1.1.2 Error Estimation and Complexity of Decision Boundary

Small-sample classifier design has become a paramount issue in functional genomics. The problem of choosing a classification rule and feature selection is exacerbated by the difficulty of error estimation with small samples, where one is compelled to train and test a classifier on the same data. Cross-validation seems to be the preferred choice of many investigators owing to its approximate global unbiasedness, but the variance of the cross-validation error estimation is typically very high, which can make it unreliable [11, 12]. As for bootstrap estimation, it has reduced variance in comparison to cross-validation, but at an increased computational cost. Bolstered Resubstitution [13] tries to achieve a reasonable compromise to the bias, variance and complexity trilemma.

Error estimation performance is generally a function of classification rule, sample size, dimensionality and feature-label distribution. In section 3 of this work, we analyze the performance of error estimation methods as a function of complexity of decision boundary. In this section, we will quantify the effect of distribution complexity on the RMS, for several data-efficient popular error estimators, including resubstitution, leave-one-out, cross-validation, bootstrap, and bolstering. Several classification rules are considered: Quadratic Discriminant Analysis (QDA), 3-nearest-neighbor (3NN) and neural networks (NNet). The analysis in this study extends to the error estimation problem some of the ideas in [14], where the true classification error was studied as a function of the complexity of the feature-label distribution and of the sample size.

The issue studied in part is critical to experimental design. If one makes no modeling assumptions with small-sample classifier design, then, virtually nothing can be said about the error of the designed classifier and hence nothing can be said

about the scientific content of the classifier – it is epistemologically vacuous [15]. On the other hand, in defining distributional complexity we want to differentiate between complexity and separability of the classes. Specifically, we want a measure of complexity that will not be related to the Bayes error but will be related to the complexity of the Bayes decision boundary. The classes may be multimodal, with different "modes" being highly interwoven in Euclidean space, but without overlap among the class-conditional densities. In this case the Bayes error will be zero, but the Bayes classifier may possess a complex decision boundary. Despite the fact that, in principle, such a situation involves perfectly separable classes, it presents one, in practice, with a difficult problem for both classifier design and error estimation. Our interest here is not with error-estimation RMS as a function of Bayes error, but as a function of distribution complexity, and thus complexity of the decision boundary, and our proposed definition of complexity reflects this fact.

In the following, we review some of the well known error estimation methods. We will compare the performance of these methods for different levels of the complexity of the decision boundary.

1.1.3 Error Estimation Methods

In two-group statistical pattern recognition, there is a *feature vector* $X \in \mathbb{R}^p$ and a *label* $Y \in \{0, 1\}$. The pair (X, Y) has a joint probability distribution $\mathbf{F}$, which is unknown in practice. Hence, one has to resort to designing classifiers from *training data*, which consists of a set of n independent observations, $S_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$, drawn from $\mathbf{F}$. A *classification rule* is a mapping $g : \{\mathbb{R}^p \times \{0, 1\}\}^n \times \mathbb{R}^p \to \{0, 1\}$. A classification rule maps the training data S_n into the *designed classifier* $g(S_n, \cdot) : \mathbb{R}^p \to \{0, 1\}$. The *true error* of a designed classifier is its error rate given the training

data set:

$$\epsilon_n[g|S_n] = P(g(S_n, X) \neq Y) = E_{\mathbf{F}}(|Y - g(S_n, X)|), \tag{1.1}$$

where the notation $E_{\mathbf{F}}$ indicates that the expectation is taken with respect to $\mathbf{F}$; in fact, one can think of (X, Y) in the above equation as a random test point (this interpretation being useful in understanding error estimation). The expected error rate over the data is given by

$$\epsilon_n[g] = E_{\mathbf{F}_n}(\epsilon_n[g|S_n]) = E_{\mathbf{F}_n} E_{\mathbf{F}}(|Y - g(S_n, X)|), \tag{1.2}$$

where $\mathbf{F}_n$ is the joint distribution of the training data S_n. This is sometimes called the *unconditional error* of the classification rule, for sample size n.

Were the underlying feature-label distribution $\mathbf{F}$ known, the true error could be computed exactly, via (1.1). In practice, one is limited to using an *error estimator*. Ideally, this estimate should be fast to compute and as close as possible to the true error, for the given training data. Most error estimators used in practice implement some form of sample-mean-like approximation using test points. The error estimator is unbiased, with respect to the unconditional error, if the test points come from independent samples not used to design the classifier.

1.1.3.1 Resubstitution

The simplest and fastest way to estimate the error of a designed classifier in the absence of test data is to compute its error directly on the sample data itself:

$$\hat{\epsilon}_{\text{resub}} = \frac{1}{n} \sum_{i=1}^{n} |y_i - g(S_n, x_i)|. \tag{1.3}$$

This *resubstitution estimator*, attributed to [16], is very fast, but is usually optimistic (i.e., low-biased) as an estimator of $\epsilon_n[g]$ [11]. For some classification rules, resubstitution can be severely low-biased, an extreme case being one-nearest-neighbor classification, in which the resubstitution estimator is identically equal to zero. Typically, the more complex is the classifier, the more optimistic is resubstitution, since complex classifiers tend to overfit the data, especially with small samples [17].

1.1.3.2 Cross-Validation

Cross-validation removes the optimism from resubstitution by employing test points not used in classifier design [18]. In *k-fold cross-validation*, the data set S_n is partitioned into k *folds* $S_{(i)}$, for $i = 1, \ldots, k$ (for simplicity, we assume that k divides n). Each fold is left out of the design process and used as a test set, and the estimate is the overall proportion of error committed on all folds:

$$\hat{\epsilon}_{\text{cvk}} = \frac{1}{n} \sum_{i=1}^{k} \sum_{j=1}^{n/k} |y_j^{(i)} - g(S_n \backslash S_{(i)}, x_j^{(i)})|, \tag{1.4}$$

where $(x_j^{(i)}, y_j^{(i)})$ is a sample in the i-th fold. The process may be repeated: several cross-validation estimates are computed using different partitions of the data into folds, and the results are averaged. A k-fold cross-validation estimator is unbiased as an estimator of $\epsilon_{n-n/k}[g]$. The most well-known cross-validation method, usually attributed to [19], is the *leave-one-out estimator*, whereby a single observation is left out each time:

$$\hat{\epsilon}_{\text{loo}} = \frac{1}{n} \sum_{i=1}^{n} |y_i - g(S_{n-1}^i, x_i)|, \tag{1.5}$$

where S_{n-1}^i is the data set resulting from deleting data point i from the original data set S_n. This corresponds to n-fold cross-validation. The leave-one-out estimator is unbiased as an estimator of $\epsilon_{n-1}[g]$. Cross-validation estimators are often

pessimistic, since they use smaller training sets to design the classifier. Their main drawback is their variance [12, 11]. They can also be quite slow to compute when the number of folds or samples is large.

1.1.3.3 Bootstrap

The bootstrap error estimation technique [20, 21] is based on the notion of an "empirical distribution" $\mathbf{F}^*$, which serves as a replacement to the original unknown distribution $\mathbf{F}$. The empirical distribution puts mass $\frac{1}{n}$ on each of the n available data points. A "bootstrap sample" S_n^* from $\mathbf{F}^*$ consists of n equally-likely draws with replacement from the original data S_n. Hence, some of the samples will appear multiple times, whereas others will not appear at all. The actual proportion of times a data point (x_i, y_i) appears in S_n^* can be written as $P_i^* = \frac{1}{n}\sum_{j=1}^{n} I_{(x_j^*,y_j^*)=(x_i,y_i)}$, where $I_S = 1$ if the statement S is true, zero otherwise. The basic *bootstrap zero estimator* [22] is written in terms of the empirical distribution as $\hat{\epsilon}_0 = E_{\mathbf{F}^*}(\,|Y - g(S_n^*, X)|\colon (X,Y) \in S_n \setminus S_n^*\,)$. In practice, the expectation $E_{\mathbf{F}^*}$ has to be approximated by a Monte-Carlo estimate based on independent replicates S_n^{*b}, for $b = 1, \ldots, B$ (B between 25 and 200 being recommended [22]):

$$\hat{\epsilon}_0 = \frac{\sum_{b=1}^{B}\sum_{i=1}^{n} |y_i - g(S_n^{*b}, x_i)|\, I_{P_i^{*b}=0}}{\sum_{b=1}^{B}\sum_{i=1}^{n} I_{P_i^{*b}=0}}. \tag{1.6}$$

The bootstrap zero estimator works like cross-validation: the classifier is designed on the bootstrap sample and tested on the original data points that are left out. It tends to be high-biased as an estimator of $\epsilon_n[g]$, since the amount of samples available for designing the classifier is on average only $(1 - e^{-1})n \approx 0.632n$. The estimator

$$\hat{\epsilon}_{\mathrm{b632}} = (1 - 0.632)\,\hat{\epsilon}_{\mathrm{resub}} + 0.632\,\hat{\epsilon}_0. \tag{1.7}$$

tries to correct this bias by doing a weighted average of the bootstrap zero and resubstitution estimators. It is known as the *.632 bootstrap estimator* [22], and has been perhaps the most popular bootstrap estimator in data mining [23]. It has low variance, but can be extremely slow to compute. In addition, it can fail when resubstitution is too low-biased [12].

1.1.3.4 Bolstered Resubstitution

As mentioned, the *empirical feature-label distribution* $\mathbf{F}^*$ is a discrete distribution that puts mass $\frac{1}{n}$ on each of the n available data points. The resubstitution estimator can be written in terms of the empirical feature-label distribution as

$$\hat{\varepsilon}_{\text{resub}} \;=\; E_{\mathbf{F}^*}[|Y - \psi_n(X)|] \tag{1.8}$$

Relative to $\mathbf{F}^*$, no distinction is made between points near or far from the decision boundary. If one spreads the probability mass of the empirical distribution at each point, then variation is reduced because points near the decision boundary will have more mass on the other side of the boundary than will points far from the decision boundary. Consider a probability density function $f_i^{\diamondsuit}$, for $i = 1, \ldots, n$, called a *bolstering kernel*, and define the *bolstered empirical distribution* $F^{\diamondsuit}$, with probability density function given by

$$f^{\diamondsuit}(X) \;=\; \frac{1}{n}\sum\nolimits_{i=1}^{n} f_i^{\diamondsuit}(X - X_i) \tag{1.9}$$

The *bolstered resubstitution* estimator [13] is obtained by replacing $\mathbf{F}^*$ by $F^{\diamondsuit}$ in Eq. (1.8) to obtain

$$\hat{\varepsilon}_{\text{bolst}} \;=\; E_{F^{\diamondsuit}}[|Y - \psi(X)|] \tag{1.10}$$

Bolstering can be applied to other error estimators; however, we only use bolstered

resubstitution, the bolstering method used the most to date.

The bolstered resubstitution estimator is given by

$$\hat{\varepsilon}_{\text{bolst}} = \frac{1}{n}\sum_{i=1}^{n}\Bigg(I_{y_i=0}\int_{A_1} f_i^{\diamondsuit}(x-x_i)\,dx + I_{y_i=1}\int_{A_0} f_i^{\diamondsuit}(x-x_i)\,dx\Bigg) \tag{1.11}$$

where $A_j = \{x \mid \psi(x) = j\}$. The integrals are the error contributions made by the data points, according to whether $y_i = 0$ or $y_i = 1$. If the classifier is linear, then the decision boundary is a hyperplane and it is usually possible to find analytical expressions for the integrals; otherwise, Monte-Carlo integration can be employed.

The amount of bolstering determines the variance and bias properties (hence, RMS also) of the bolstered estimator. As a general rule, wider bolstering kernels lead to lower-variance estimators, but after a certain point this advantage becomes offset by increasing bias. A zero-mean, spherical Gaussian bolstering kernel $f_i^{\diamond}$ with covariance matrix of the form $\kappa_i^2\mathbf{I}$, where $\mathbf{I}$ is the identity matrix, has been proposed [13], and has been shown to work well in low-dimensional feature spaces. The standard deviation κ_i is based on a non-parametric sample-based estimator, $\hat{d}_y$, of the mean minimum distance between points belonging to class y. Specifically, $\kappa_i = \hat{d}_{y_i}/\alpha_p$, where α_p is a correction constant that depends only on the dimension p. We refer to [13] for details on computing $\hat{d}_y$ and α_p.

1.2 Biomarker Validation: Selected Reaction Monitoring

Earlier in this chapter we reviewed the two main stages of biomarker development pipeline, the discovery phase and the validation phase, and mentioned that the selected biomarkers from the discovery step are validated in the next stage before moving on to further analysis in clinical studies.

Different Mass Spectrometry (MS) modes with high speed and high resolution are used successfully for the discovery stage which results in a set of candidate biomarkers. The validation phase requires MS modes with high sensitivity, such as Selected Reaction Monitoring (SRM). SRM is a tandem MS mode that exploits unique capabilities of the triple-quadrupole (QQQ) system to reach high precision in detecting biomarkers in complex samples. The first and third quadrupoles act as filters to select a predefined m/z value, while the second quadrupole works as a collision induced cell. The two stages of mass selection make SRM the right choice for quantitative analysis of low-abundance biomarkers.

Figure 1.3 displays the idealized schematics of SRM analysis on QQQ MS. The co-eluting analytes that enter the first quadrupole are filtered based on predefined m/z values and enter the second quadrupole for collision induced dissociation. The resulting fragment ions are then filtered by third quadrupole passing the preset m/z values for the desired fragment ions. The two stages of mass filtering in SRM and its targeted nature lead to an increased sensitivity by one or two orders of magnitude compared with usual full scan methods. It is worthy mentioning that the term "Multiple Reaction Monitoring" (MRM) has been used to describe parallel acquisition of SRM for measurement of several target ions. However, to avoid ambiguity between the number of transitions monitored and number of stages used in the mass spectrometry analysis (MS^n), its use is deprecated by IUPAC [24].

In this work, in section 4, SRM-based biomarker validation pipeline is modeled. Protein mixture model, sample complexity and purification, peptide mixture model, peptide ionization efficiency, transition, and peptide modification are the most important part of the model. The model developed in this study is then deployed to analyze the performance of biomarker validation workflow using SRM experiments, with different model parameter settings. Figure 4.2 displays the simulation process.

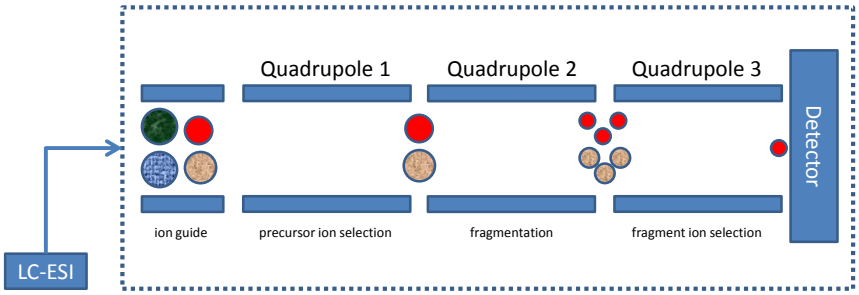


Figure 1.3: Idealized schematics of QQQ MS used in SRM analysis. The first quadruple filters out most co-eluting ions from the chromatographic system. However, interfering ions may pass Q1 and enter Q2. Ions in Q2 are fragmented and form the input of the Q3. Ideally the specific m/z selection in Q3 passes only fragments of the desired ion and eliminates interfering ions.

The list of candidate biomarkers generated based on the protein mixture model is the input of the SRM pipeline. In different stages of this process, the protein mixture data is affected by different noise sources depending on the experiment setting. The model parameters are changed during the simulation and for each parameter setting the average performance is found.

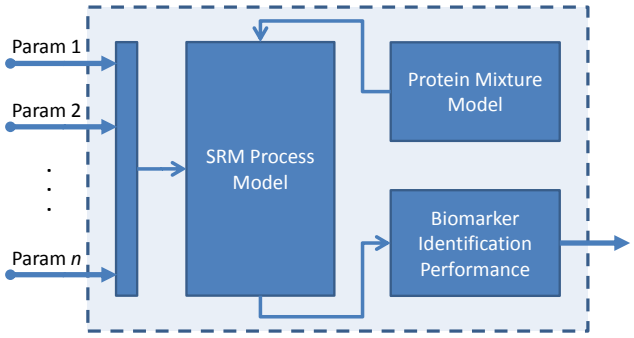


Figure 1.4: The entire simulation process. The protein abundance mixture data enters the SRM process and is affected by different noise sources in different levels of the process. The noisy data enters the biomarker validation block, where the ranking power and true positive rate are used to measure the performance of the overall biomarker validation process.

2. A FAST ALGORITHM FOR U-CURVE BRANCH-AND-BOUND FEATURE SELECTION

Feature selection is the problem of finding an optimal subset of a finite set of features that minimizes a cost function, often taken to be the classification error [4]. For a problem with n features, there are $\binom{n}{d}$ possible feature sets of size d and 2^n possible total possible feature sets. This makes the feature selection problem a hard task in dealing with large number of candidate features n.

Many algorithms have been proposed that use heuristics to find the suboptimal feature set in fewer evaluations than exhaustive search. Sequential Forward Selection (SFS) and Sequential Backward Selection (SBS) are two well-known suboptimal search algorithms. However, if the goal is to find the global best features, all the feature sets should be evaluated or as another solution, the assumptions of the search problem can be used to discard the solutions which do not have the potential to be the global best feature set. The well-known Branch-and-Bound (BB) algorithm proposed in [5] uses the monotonicity property of the criterion function to systematically enumerate all candidate solutions, where large subsets of fruitless candidates are discarded.

The original BB algorithm assumes a monotonic criterion function. This assumption does not hold in practice. For example, with peaking phenomenon the true error of a classifier designed using small number of samples, decreases and then increases by using more features. Recently Ris and colleagues in [10] proposed a U-curve Brand-and-Bound (UBB) which extends the idea of BB algorithm to the U-curved criterion functions. The purpose of the current part of the dissertation is to propose and analyze a fast algorithm to implement the UBB method, which outperforms the

original algorithm. A cost function is proposed in this part of the work to model the feature selection problem with peaking phenomenon. The model is able to control the delay in peaking using the parameters which determine the number of features at which peaking happens. This enables us to study the features selection algorithms in dealing with problems with different peaking behaviors.

2.1 U-curve feature selection

In this section, we introduce the U-curve optimization problem formally [10]. Let $\mathcal{P}(S)$ denote the power set of the finite non-empty set S.

Definition 1 (Chain)**.** *A chain is a collection* $\{X_1, X_2, ..., X_k\} \subseteq \mathcal{P}(S)$*, such that* $X_1 \subseteq X_2 \subseteq ... \subseteq X_k$*.*

Definition 2 (U-shaped curve)**.** *Let* $\mathcal{X} \subseteq \mathcal{P}(S)$ *be a chain. A function* $f : \mathcal{X} \to \mathbb{R}$ *describes a U-shaped curve if, for any* $X_1, X_2, X_3 \in \mathcal{X}$*, then* $X_1 \subseteq X_2 \subseteq X_3$ *implies that* $f(X_2) \leq \max\{f(X_1), f(X_3)\}$*.*

Definition 3 (Decomposability in U-shaped curves)**.** *A cost function* $c : \mathcal{P}(S) \to \mathbb{R}$ *is decomposable in U-shaped curves if, for each chain* $\mathcal{X} \subseteq \mathcal{P}(S)$*, the restriction of* c *to* $\mathcal{X}$ *describes a U-shaped curve.*

Definition 4 (Minimum cost)**.** *Let* X^* *be an element of* $\mathcal{X} \subseteq \mathcal{P}(S)$ *and let* c *be a function defined on* $\mathcal{P}(S)$*. If there does not exist another element* $X \in \mathcal{X}$ *such that* $c(X) < c(X^*)$*, then* X^* *is of minimum cost in* $\mathcal{X}$*. If* $\mathcal{X} = \mathcal{P}(S)$*, then we say that* X^* *is of minimum cost.*

The U-curve problem can be defined as follows: Given a non-empty set S and a cost function $c : \mathcal{P}(S) \to \mathbb{R}$ that is decomposable in U-shaped curves, find an element $X^* \in \mathcal{P}(S)$ of minimum cost. Figure 2.1 (a) illustrates the U-curve problem. In this

figure, four different chains are highlighted in red. Figures 2.1 (b,c) display 2D and 3D plots of the problem in figure 2.1 (a).

2.1.1 U-curve Branch and Bound

The UBB algorithm proposed in [10] uses the U-curve structure of the cost function to find the minimum cost element of S without evaluating all the elements in S. Through a recursive enumeration scheme, it first constructs a tree T and then uses it as the search space. The fact that c is decomposable in U-shaped curves is used to prune the tree during the search. The algorithm branches until the cost of a visited element starts to increase. Observing the first increase in the cost function, the tree is pruned. The algorithm continues the search on the pruned tree. Figure 2.2 shows the process of UBB for selecting the optimal solution of a U-curve problem. More details about UBB and its pesudocode can be found in [10].

2.2 Proposed Algorithm

The results in [10] show that UBB requires fewer function calls compared to Exhaustive Search (ES) in finding the global best solution of the U-curve problem. However, UBB's number of function calls is still high (about half of ES's function calls). To tackle the high number of function calls of UBB, an improved algorithm is proposed here, which consists of two main innovations:

1) **Iterative updating of optimal chains.** Through an enumeration process, UBB constructs a tree structure as the search space of the optimization problem. The tree is pruned when the cost of an element in the search chain starts increasing. Although in the first iterations of the UBB algorithm, finding a minimum element in a chain leads to removal of many elements in the tree, in the next steps the search chains are not the best possible chains in the search space and the pruning becomes very slow. This is one of the reasons that the number of function calls in

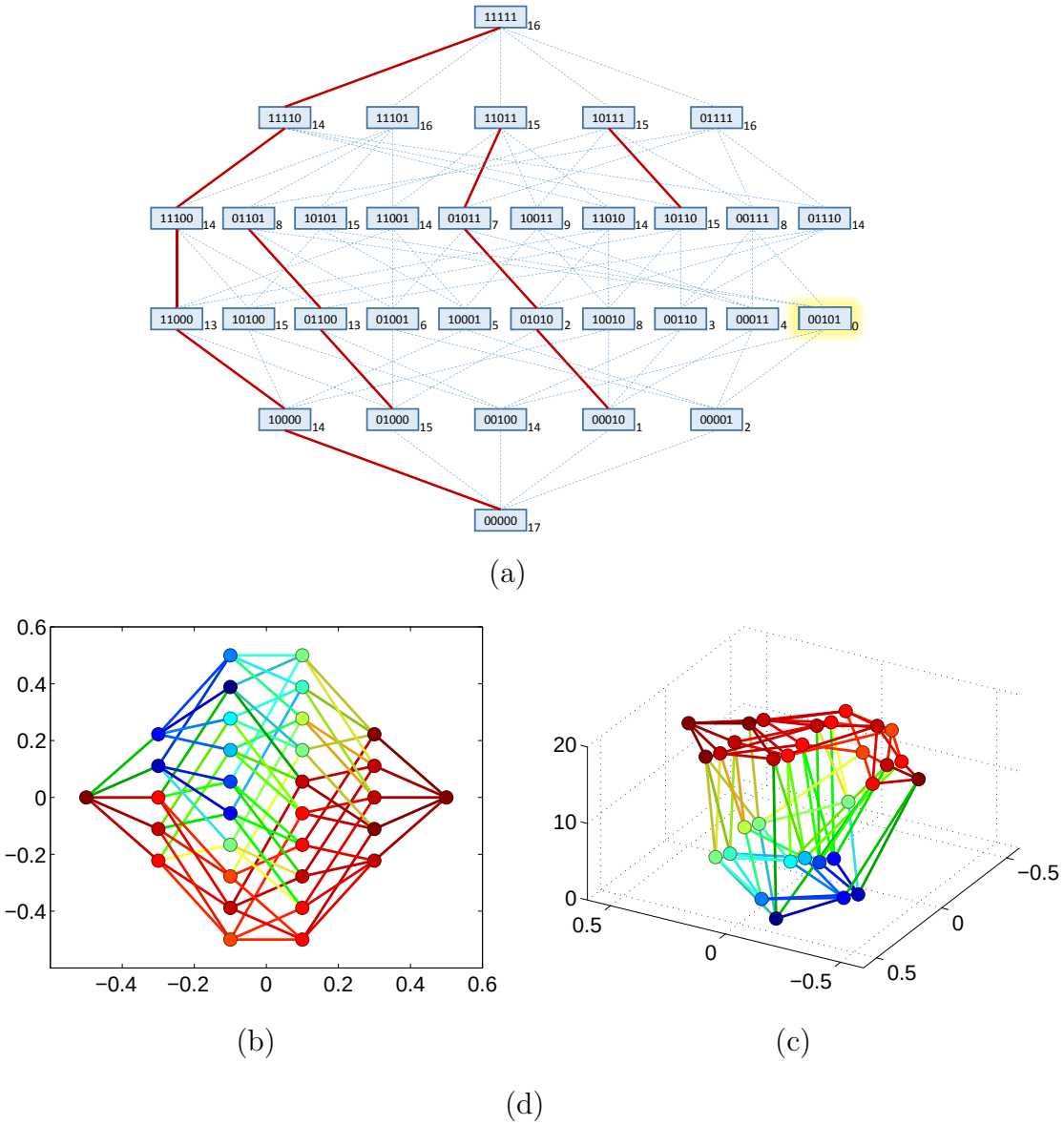


Figure 2.1: (a) U-curve feature selection with $n = 5$, highlighting 4 chains. (b) 2D plot (c) 3D plot. The element with the yellow shade is the global minimum.

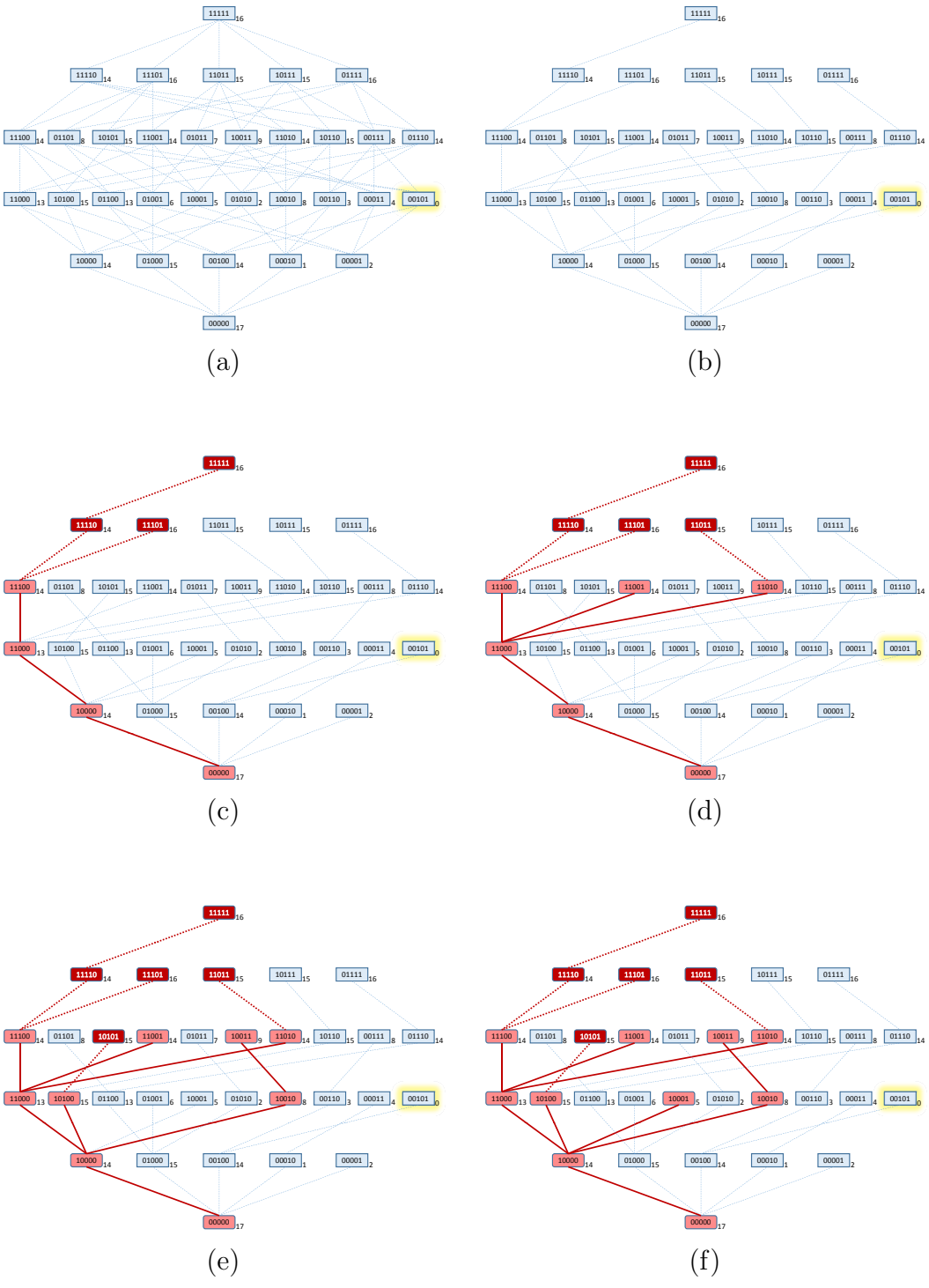


Figure 2.2: UBB algorithm. (a) Search space. (b) the tree produced by enumeration scheme. (c-f) Four steps of the algorithm.

UBB is high. To improve the pruning process, instead of limiting the search to the tree structure constructed through the enumeration process of UBB, at each step of IUBB we determine the optimal chain in the search space which leads to pruning the most elements of the search space. This is done as follows. For a feature selection problem with n features, the search space can be represented by a Boolean lattice $\mathcal{L}$ of degree n. Let L_l be organized in layers, as represented in Figures 2 and 3, where L_l denotes the l-th layer, for $0 \leq\leq n$, that is, let L_l contain all possible feature sets of size l. In addition, let each feature set X be represented by a binary string of size n, where a "1" at the ith position indicates that feature set i belongs to X, again as represented in Figures 2 and 3. For $0 \leq l \leq n-1$, we define $\mathbf{R}_l = [r_{ij}]$, a matrix of size $\binom{n}{l} \times \binom{n}{l+1}$, with i, j-element general element given by

$$r_{ij} = \begin{cases} 1, & \text{if } |X_j^{l+1} - X_i^l| = 1 \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

where X_i^l and X_j^{l+1} are ith and jth elements of L_l and L_{l+1}, respectively. In other words, r_{ij} is the Hamming distance between X_j^{l+1} and X_i^l. If $r_{ij} = 1$, then the two feature sets are on the same chain. Let $\mathbf{C}_l$ be the vector of row sums of $\mathbf{R}_l$, and define the vectors

$$\begin{aligned} \mathbf{T}_n &= 0\,, \\ \mathbf{T}_{n-1} &= \mathbf{C}_{n-1}\,, \\ \mathbf{T}_l &= \mathbf{C}_l + \mathbf{R}_l \times \mathbf{T}_{l+1}\,, \quad l = n-2, n-3, ..., 0\,. \end{aligned} \tag{2.2}$$

The ith element of $\mathbf{T}_l$ indicates the pruning gain of its corresponding element in the search space. We use $\mathbf{T}_l$ to find chains for which finding the minimum element results

in maximum pruning of the search space. At each step of the algorithm, an optimal chain $\mathcal{X}^*$ is found, and after determining the minimum element of the chain, all the states connected to the optimal element are removed from the search space. Then the matrices $\mathbf{R}_l$, $\mathbf{C}_l$ and $\mathbf{T}_l$ are updated, and the algorithm continues until there is no remaining element in the search space.

2) **Use of bisection to find chain minimum.** The main drawback of UBB is that in finding the minimum element of a chain, it searches all the elements before reaching the element that shows increasing cost value. That is, for a chain $\mathcal{X} = \{X_1, X_2, ..., X_k\}$, if the U-curve cost function has a minimum $X^* = X_{i^*}$, the algorithm evaluates the cost function $(i^* + 1)$ times to find X^*. When dealing with a large number of features, the algorithm will tend to need a large number of unnecessary function calls to find X^*. To use the U-curve assumption efficiently, as an alternative, faster methods can be used to find the minimum element of the chain. We use bisection in the proposed algorithm. This changes the complexity of finding the minimum of the chain $\mathcal{X}$ from $O(|\mathcal{X}|)$ to $O(\log(|\mathcal{X}|))$. Figure 2.3 shows the number of the function calls required to find the minimum element X^* of the chain when i^* is uniformly distributed in the set $2, \ldots, |\mathcal{X}|-1$ and the cost function is $c(i) = (i - i^*)^2$. As we see, at $|\mathcal{X}| = 500$, bisection requires on average 17 function evaluations, while $|\mathcal{X}|/2 = 250$ function evaluations are needed on average by the method in UBB to find the minimum element X_i^*.

The pseudocode of the algorithm is shown in Algorithm 1.

Compared to the original UBB algorithm, the proposed IUBB algorithm uses the U-assumption efficiently by first using a different search structure which focuses on an optimal chain $\mathcal{X}^*$ in the search space at each step of the algorithm. Then using the U-assumption for not only pruning the search space, but also for finding the minimum element X^* of each chain (`Bisection` module). This improved and

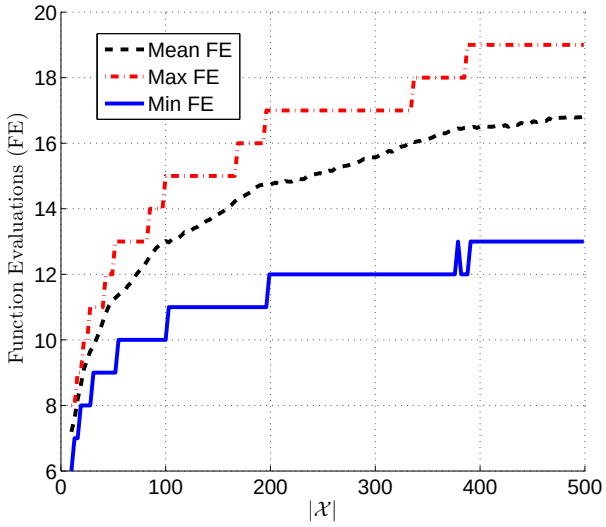


Figure 2.3: Number of function evaluations required to find the minimum element of the chain $\mathcal{X}$ by bisection vs. $|\mathcal{X}|$.

efficient use of U-curve assumption enables the proposed algorithm to outperform UBB in most the search problems. In the next section we will compare the two algorithms for a few different feature selection experiments.

2.3 Experimental Results

In this section the performances of the proposed IUBB algorithm and the original UBB algorithm are compared. The analysis is broken into two different parts. First, the algorithms are compared for a set of synthetic benchmark U-curve problems. The parameters in the synthetic problems allows us to study the effect of varying structure of the U-shape feature selection problem and analyze the behavior of the algorithms while the complexity of the problem changes over the variation of the parameters. Next, the UBB and IUBB algorithms are applied in feature selection for a real classification problem and their performances are compared.

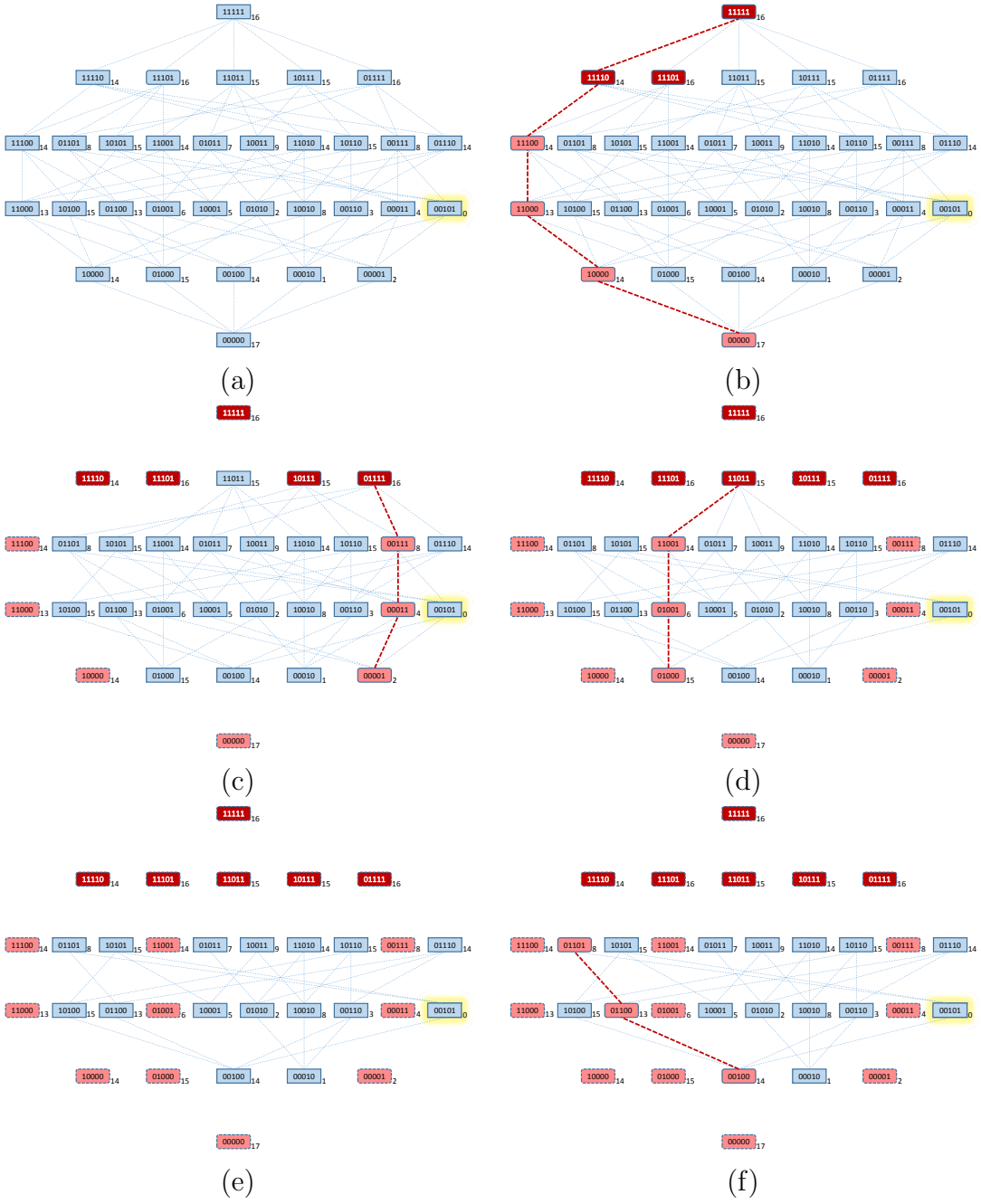


Figure 2.4: IUBB process. (a) The original search space. (b-f) Five steps of the algorithm. The elements with pink background indicate the the feature sets evaluated. Red background elements are the ones that are removed from the search space without evaluation of the cost value. The blue elements are the feature sets that are not evaluated nor removed due to the U-assumption. The selected optimal chain is shown by a red dashed line in all the diagrams.

Algorithm 1 IUBB Algorithm

Initialize Boolean lattice $\mathcal{L}$ of degree n.
$N_r \Leftarrow$ Number of elements is the search space that are not visited.
$c^{opt} \Leftarrow \infty$
while $N_r \geq 0$ **do**
  Find the optimal chain $\mathcal{X}^*$
  `Bisection`$(\mathcal{X}^*) \Rightarrow X^*$
  if $c(X^*) < c^{opt}$ **then**
    $c^{opt} \Leftarrow c(X^*)$
    $X^{opt} \Leftarrow X^*$
  end if
  `Prune`$(\mathcal{L}, \mathcal{X}^*, X^*) \Rightarrow (\mathcal{L}, N_r)$
end while
return X^{opt} and c^{opt}

2.3.1 Synthetic Benchmark U-curve Problem

In this section, a model is introduced for the U-curve feature selection problem, which allows us to change the structure of the problem and observe the performance of the algorithms while the difficulty of the feature selection problem varies. The parameters in the model control the number of features in the optimal feature set, representing problems ranging from early peaking to late peaking. This model can be used as a benchmark U-curve cost function to study other feature selection methods. The model for the cost function is given by:

$$z = f(\mathbf{x}|\mathbf{c}, \mathbf{W}, z_{sup}) = z_{sup}[1 - exp(\frac{1}{2}(\mathbf{x} - \mathbf{c})^T \mathbf{W}(\mathbf{x} - \mathbf{c}))] \tag{2.3}$$

where

$$
\begin{aligned}
&n : \text{Dimension of the feature selection problem} \\
&\mathbf{x} \in \{0,1\}^n : \text{Binary Feature Vector} \\
&\mathbf{c} \in \{0,1\}^n : \text{Center (Global Minimum of Cost Function)} \\
&\mathbf{W} \in R^{(n \times n)} : \text{Positive Definite Weighting Matrix (Shaping Matrix)} \\
&z_{sup} : \text{Cost Supremum, Cost Scale or Ideal Maximum Value of Cost}
\end{aligned}
\tag{2.4}
$$

The parameter $0 \leq \alpha \leq 1$ controls the number of the features in the optimal set of the features, i.e.

$$\alpha = \frac{1}{n} \sum i = 1nc_i \tag{2.5}$$

The parameter α controls the peaking "delay" and the number of features present in the optimal feature set.

To have a better view of the proposed model, in figure 2.5 we plot two samples of u-curve model problem for two different settings of parameters, ($\alpha = 0.3$, $n = 7$), ($\alpha = 0.6$ and $n = 7$). Figure 2.5 (a) shows a U-curve problem with early peaking. The number of features in the optimal set of the features in the problem depicted in Figure 2.5 (b) is almost 60% of the total number of features and peaking happens late. When $\alpha = 1$ the cost function is monotonic which models a feature selection problem with large sample size with monotonically decreasing classification error as more sample are used in the classifier design..

These two classes of problems with different characteristics can be used to see how the shift in peaking affects the algorithms. The structure of the problems depicted in figure 2.5 shows that the model proposed for U-curve problem model is flexible enough to generate a vast range of feature selection problems with peaking phenomena. To study the robustness of the algorithms when the U-curve assumption

does not hold, in the next sections we will add parameters to the model to simulate the real feature selection problems where the estimated error over the elements of the chains does not hold the U-assumption completely.

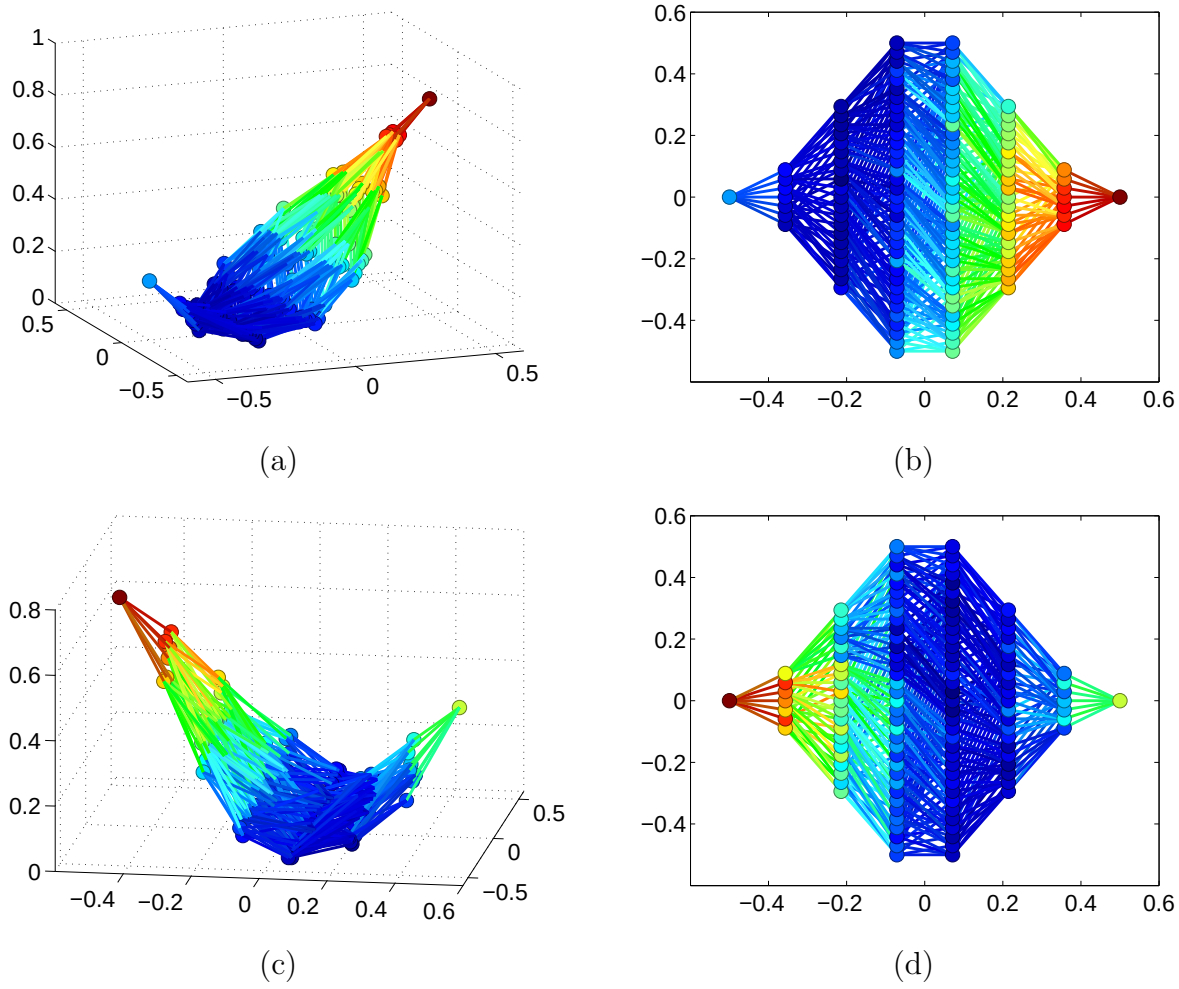


Figure 2.5: The u-curve problem model for $n = 7$. (a and b) 3D and 2D representation of the cost for $\alpha = 0.3$. (c and d) 3D and 2D representation of the cost for $\alpha = 0.6$. Increasing α shifts the peaking from few number of features to selection of more features in the optimal feature set.

The u-curve problem model proposed here, can be used as a benchmark for the study of the other algorithms that address this class of optimization problems. The parameters in the cost model enables us to change the structure of the feature se-

lection problem and see how the behavior of the algorithms change over a set of different problems.

2.3.2 Results

A set of different test U-curve feature selection optimization problems of size n and different structures resulting from different sets of model parameters, are used to assess the performance of each method. Different measures are used to compare the two feature selection algorithms. Before introducing these measures, we define the following terms.

- n_{FE}: the number of function evaluations. This is the number of feature sets visited and evaluated by an algorithm.
- n_{PR}: the number of feature sets pruned by applying the U-assumption on the search structure.
- n_{RM}: the number of feature sets removed from search for reasons other that the Pruning. For example removal of some of the feature sets from a chain using Bisection to find the minimum element.
- n_{DD}: the number of feature sets Pruned or Removed ($n_{DD} = n_{PR} + n_{RM}$).
- n_{UD}: the number of feature sets not Visited, Pruned or Removed ($n_{UD} = 2^n - n_{FE} - n_{PR} - n_{RM}$).

Definition 5 (Search Efficiency)**.** *For a global search algorithm, at each n_{FE}, the Search Efficiency is defined as:*

$$SE = \frac{n_{FE} + n_{DD}}{n_{FE}}, SE \geq 1 \tag{2.6}$$

Search Efficiency shows the ability of the two algorithms in using the U-assumption for discarding undesired solutions from search space. For Exhaustive search the $SE = 1$ as there is no use of U-assumption in the search process and there is no additional gain in evaluating a feature set ($n_{DD} = 0$). However, IUBB and UBB try to use the information of each function evaluation to discard some of the solutions. Comparing the search efficiency of the two algorithms will allow us to see how efficiently the two algorithms use the U-assumption in finding the global best feature sets. On the other hand SE indicates the level of the trust over a solution if the algorithms stop in a fixed n_{FE} due to limited computational resources. For example if we stop the search at $n_{FE} = 100$, then $SE = 2$ means that the best solution found at $n_{FE} = 100$ represents $2 \times n_{FE}$ candidate solutions of the search space. A low value of SE will show that the algorithm uses the assumption inefficiently and on the other hand a solution found by the algorithm is only representing the part of search space that has directly been visited and evaluated.

We will use the following indices for comparing the two algorithms.

- **Search Efficiency**: Will show the efficiency of the algorithms in using the U-assumption.
- n^0_{FE}: The number of function evaluations required to find the optimal feature sets.
- **Best Cost**: The cost of best feature set found by each algorithm will be used to compare two algorithms.

Using the mentioned measures, we compare two algorithms in figures 2.6 and 2.7. Figure 2.6 (a) shows the best cost found by each algorithm in n_{FE} number of function evaluations. Also figure 2.6 (b) depicts number of feature sets in the search

space that need to be evaluated vs. n_{FE}. On the other hand, figure 2.6 (c) indicates number of feature sets in the search space that are pruned or removed from search space vs. n_{FE}. Also figure 2.6 (d) shows the ratio of the plots in Figure 2.6(c) vs. n_{FE}. In figure 2.7 (a) we see the ratio plot of the number of feature sets in the search space that need to be evaluated vs. n_{FE}. Also figure 2.7 (b) shows the percentage of feature sets in the search space that need to be evaluated vs. n_{FE}. The difference between percentage of feature sets in the search space that need to be evaluated vs. n_{FE} is shown in figure 2.7. Finally figure 2.7 (d) depicts search Efficiency of two algorithms vs. n_{FE}.

IUBB has a high search gain in low NFEs which very important as this represents a real case where the computational resources limit the number of estimation of the error of the candidate feature sets and searching the entire search space and even a small percentage of it is computationally intensive.

Figures 2.6 and 2.7 compared two algorithms for a single application of UBB and IUBB on feature selection problems with different dimensions. In the next section we will evaluate the two algorithms based on their average performance in dealing with U-curve feature selection problems with different structures. To do this, we will limit the number of function evaluation of each algorithm to a fixed number (5% and 10% of the search entire search space) and will use the cost of best feature set found by each algorithm and the search efficiency at these fixed n_{FE}s. This models a more realistic scenario where search is limited by the limitations of the computational resources.

2.3.3 Peaking Delay

In this part, fixing the dimension of the feature selection problem to 15 and 17, we want to study how the variation of the parameters in the model problem, will

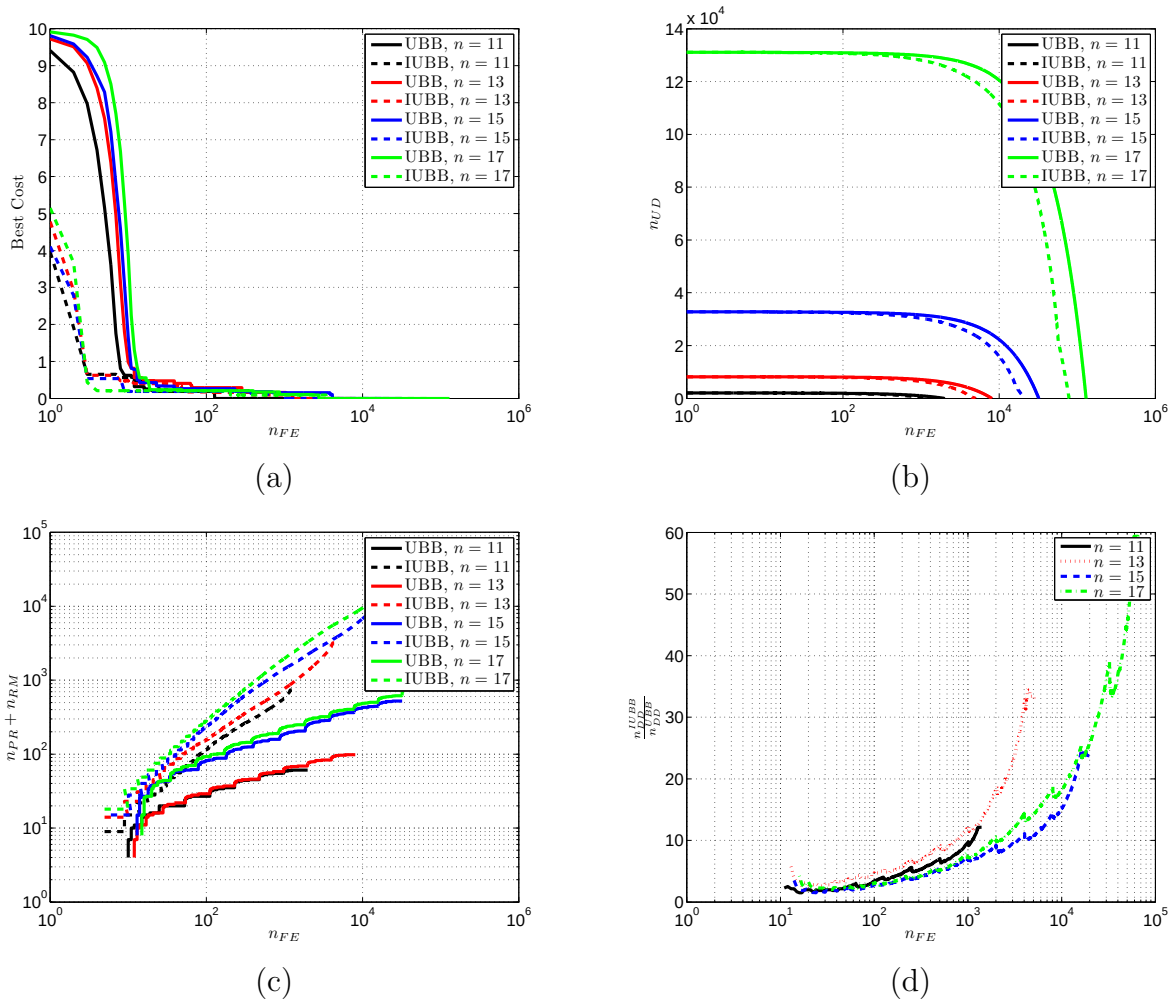


Figure 2.6: Comparing a single run of two algorithms ($\alpha = 0.75$) for different values of n. (a) The best cost found by each algorithm in n_{FE} number of function evaluations. (b) Number of feature sets in the search space that need to be evaluated vs. n_{FE}. (c) Number of feature sets in the search space that are pruned or removed from search space vs. n_{FE}. (d) The ratio of the plots in (c) vs. n_{FE}.

affect the performance of IUBB, and UBB. Figure 2.8 (a, b) show the effect of the model parameter α on the performance of the algorithms when dimension n is set to 15.

As we see in Figure 2.8 (a) the increase of the model parameter α from 0 to 0.5 increases the number of cost evaluation required by each algorithm. This behavior is not beyond our expectations, as the increase of alpha, will move the minimum elements of the chains from the selection of fewer features towards the selection of more features and this will lower the pruning potential of the algorithms, delaying their success in finding the best feature sets. On the other hand, when α increases the chance of finding global minimum in the initially constructed chains decreases. The algorithms have the worst performance when α is about 0.5. Increasing α in the interval $[0.5, 1]$ improves the performance of the two methods. When α is about 1, this means that the optimal solution is in the first selected chain and the two algorithms perform well.

Figure 2.8 (b) shows the ratio of the number of function evaluations required to find the best features. As we see UBB has a better performance for $\alpha \in [0, 0.5]$. For $\alpha \geq 0$ IUBB has better performance. The ratio plot shows that the UBB might require about 40 times more evaluations for a problem with late peaking.

Figure 2.9 compares the two algorithms using their Avg Cost, SE and SE ratio plots for $NFE = 5\%$ and $NFE = 10\%$ of the search space. As we see, if the two algorithms are provided with a fixed number of possible function evaluations, IUBB has a better performance in terms of the cost of best feature set and the search efficiency, over the entire range of possible αs. Also we see that the two algorithms have high search efficiency for low values of αs and as α increases the search efficiency decreases very fast for both of the algorithms. This shows the importance of the performance of the algorithms in higher values of α. In the next section we will

study the effect of dimension of the feature selection problem on the performance of the two algorithms. Figure 2.10 shows the same results for $n = 17$

2.3.4 Dimension

One of the main drawbacks of the Branch-and-Bound algorithms is their weak performance as the dimension of the feature selection problem increases. The results show that UBB has inherited this weakness as its performance in high dimensions decreases. In this section the proposed U-curve benchmark problem model is used to compare the algorithms against the increase of the problem dimension. The dimension n changes from 10 to 17 and performance of the two algorithms are evaluated based on two different criteria.

Figure 2.11 (a) shows a plot of the average number of function calls required by two algorithms to find best feature sets. Figure 2.11 (b) depicts the ratio of the number of the function calls used by the two algorithms to find the global minimum. In figure 2.11 (c) the average percentage of the search space evaluated by each algorithm to find best features is depicted. Figure 2.11 (d) shows the difference between the two plots in figure 2.11 (c). The results will be described in more details. Also smaller error bars in figures 2.11 (a and c) indicate that the IUBB is more robust than UBB to the changes in the structure of the problem.

To have a better understanding of the results of the two algorithms, figure 2.12 (a and b) compares the average cost of the two algorithms when the 5% and 10% of the search space is evaluated by two algorithms. Figures 2.12 (c and d) depict the average search gain at 5% and 10% of the search space. on the other hand figures 2.12 (e and f) show the ratio of the average search gain at 5% and 10% of the search space. As we see in these figures limiting the search to the 5% and 10% of the search space which is the case when dealing with real feature selection problems

where the limitation in the computational resources confine the number of function evaluation, IUBB has a better performance. The average cost of the best feature set found by IUBB is less than that of UBB. On the other hand the search efficiency plots shows that IUBB outperforms UBB. Higher value of search gain besides low average cost value indicates that not only IUBB results in a better set of features, but also its results are more trustworthy as they represent a larger portion of search space. The ratio of the search gain plot shows that IUBB uses the U-curve assumption more efficiently than the original UBB.

The result in Figures 2.12 were for $\alpha = 0.7$. We observed that with this value of α the search efficiency of the UBB converges to 1 (Exhaustive search) as n increases. To study the behavior of the two algorithms in more details, we change the value of the α and see its effect on the performance of the two algorithms as the dimension n increases.

Figure 2.13 (a) shows a plot of the average number of function calls required by two algorithms to find best feature sets. Figure 2.13 (b) the ratio of the number of the function calls used by the two algorithms to find the global minimum. In figure 2.13 (c) the average percentage of the search space evaluated by each algorithm to find best features is depicted. Figure 2.13 (d) shows the difference between the two plots in figure 2.13 (c). The results will be described in more details. Also smaller error bars indicate that the IUBB is more robust than UBB to the changes in the structure of the problem.

Figure 2.14 (a and b) compares the average cost of the two algorithms when the 5% and 10% of the search space is evaluated by two algorithms. Figures 2.14 (c and d) depicts the average search gain at 5% and 10% of the search space. on the other hand figures 2.14 (e and f) show the ratio of the average search gain at 5% and 10% of the search space between two algorithms.

2.3.5 Validation of U-curve Assumption

The previous results compared two algorithms in finding the global minimum of a set of U-curve optimization problems. In this section we study the performance of the algorithms in cases where the U-curve assumption is not held. We add new parameters to the model that control deviation from U-curve assumption. The mathematical model of the modified cost function follows:

$$z = f(\mathbf{x}|\mathbf{c}, \mathbf{W}, z_{sup}) = z_{sup}[1 - exp(\frac{1}{2}(\mathbf{x} - \mathbf{c})^T \mathbf{W}(\mathbf{x} - \mathbf{c})) + A\cos(2\pi f \bar{\mathbf{x}})] \quad (2.7)$$

where

$$\begin{aligned} &\bar{\mathbf{x}} : \text{mean of } \mathbf{x} \\ &A : \text{Amplitude of the } cos \text{ term} \\ &f : \text{Frequency of the } cos \text{ term} \end{aligned} \quad (2.8)$$

The cos term in the cost function controls the deviation of the problem from U-curve assumption. If A is set to 0 the problem is a U-curve. If $A > 0$ then the problem does not satisfy the U-curve assumption and each chain might have more than one local minimum. The value of f controls the number (frequency) of local minimums and A controls the depth of the local minimums.

To see the effect of changing A and f, figure 2.15 shows the 3D and 2D plot of the cost value of the different elements of the search space for two different sets of the parameters. In figure 2.15 the parameters A and f are set to 0.2 and 2 respectively. Figure 2.15 shows the cost where $A = 0.4$ and $f = 3$. As we see A and f control the level of deviation from the assumption and can be used to study the performance of the algorithms when the problem is not u-curve. Increasing A increases the depth of the local minimums in the chains. A proper algorithm should be able to avoid these

minimums for a large enough range of A. On the other hand increasing f generally increases the frequency of the local minimums, making it harder for the algorithms to find the global minimum of a chain and prune the search space.

Figure 2.15 shows two samples of u-curve model problem for two different settings of parameters, ($A = 0.2$, $f = 2$) and ($A = 0.4$, $f = 3$). In figure 2.15 (b) the deviation from U-curve assumption is higher compared to problem depicted in figure 2.15 (a). As we see the two parameters A and f, enrich our model and make it flexible for generating problems that do not hold the U-curve assumption completely.

Figure 2.16 shows the cost of the best feature sets found by two algorithm as the value of A increases. As we see, for $A \leq 0.075$ the two algorithms are able tolerate the deviation from U-curve assumption. However as A becomes greater than 0.075, UBB loses its performance suddenly while IUBB is robust to the increases of the value of A.

Figure 2.17 shows the same results as figure 2.16 for $f = 3$. As we see, in this case, even a small value of deviation from U-assumption is enough for UBB to get stock in local minimums of the function.

2.3.6 Classification Problem

After comparing the two algorithms over a range of different U-curve problems, in this section we apply the algorithms on real feature selection problems. A set of different feature selection problems are generated based on the method and data model described in the next section. The estimated value of the true error using some of the well-known error estimation methods is used by the algorithms to select the optimal set of the features.

2.3.6.1 Data Model

This section describes the model used for the data. The features are divided into two different groups, markers and non-markers. We use the model proposed in [25] where a Gaussian block model is used for the abundance of markers and non-markers, with the latter group being divided into two groups, high-variance and low-variance. More details are given next.

There are altogether D_{gm} global markers. The class-conditional distributions are D_{gm}-dimension Gaussian: $N(\mu_0^{gm}, \Sigma_0^{gm})$ for class 0 and $N(\mu_1^{gm}, \Sigma_1^{gm})$ for class 1, where μ_0^{gm} and μ_1^{gm} are the mean vectors of class 0 and 1, respectively, and Σ_0^{gm} and Σ_1^{gm} are the covariance matrices.

The means are set to $\mu_0^{gm} = (0, 0, ..., 0)$ and $\mu_1^{gm} = (1, 1, ..., 1)$, while a block-based structure is used to define the covariance matrices, whereby markers are divided into groups of D_m markers each. Markers from different groups are uncorrelated and markers of the same group possess same correlation ρ between each other. Specifically, we define Σ_0^{gm} and Σ_1^{gm} as $\Sigma_0^{gm} = \sigma_0^2 \times \Sigma$ and $\Sigma_1^{gm} = \sigma_1^2 \times \Sigma$, with

$$\Sigma = \begin{bmatrix} R_\rho & 0 & \cdots & 0 \\ 0 & R_\rho & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_\rho \end{bmatrix}, \tag{2.9}$$

where R_ρ is a $D_m \times D_m$ matrix with 1's on the diagonal and ρ's elsewhere.

Non-markers are features that provide no discriminating power between two classes. *High-variance* non-markers are uncorrelated. For any feature, the distribution will be a mixture of Gaussians, $N(0, \sigma_0)$ and $N(1, \sigma_1)$, where σ_0 and σ_1 are the same values used in markers. The number of high-variance non-markers is de-

noted by D_{hv}.

Low-variance non-markers are also uncorrelated. For any feature, the distribution is $N(0, \sigma_0)$. The number of Low-variance non-markers is $D_{lv} = D - D_{gm} - D_{hv}$. Figure 2.18 shows the peaking phenomenon for LDA classifier designed using the samples generated from the model.

2.3.6.2 Results

Using the data model a set of different feature selection problems are generated. These problems are used to compare the two algorithms for feature selection in a set of real problems. Table 2.1 shows the summary of the parameter values used in this section. Figures 2.19 and 2.20 show the application of UBB and IUBB on a set of feature selection problems generated using the model. The results show that how the accuracy of the error estimation algorithm affects the feature selection algorithms.

Table 2.1: Summary of parameters

Parameter		**Value**
Classification rule		LDA
No. of sample size	nTr	40
No. of total feature dimensions	n	15
No. of global markers	D_{gm}	10
Blok Size	D_m	2
	ρ	0.5
	σ_0	0.3

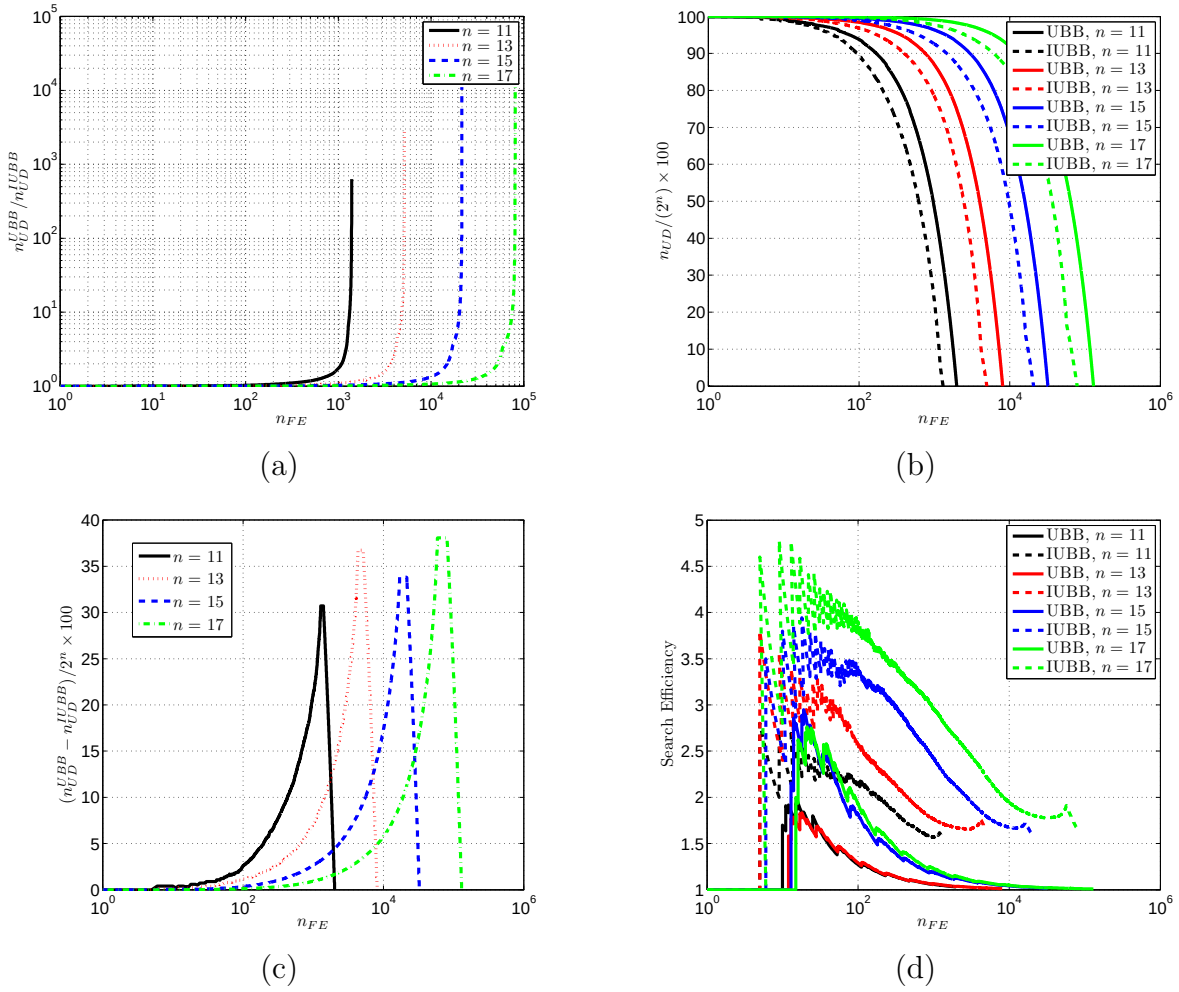


Figure 2.7: Comparing a single run of two algorithms ($\alpha = 0.75$) for different values of n. (a) The ratio plot of the number of feature sets in the search space that need to be evaluated vs. n_{FE}. (b) The percentage of feature sets in the search space that need to be evaluated vs. n_{FE}. (c) The difference between percentage of feature sets in the search space that need to be evaluated vs. n_{FE}. (d) Search Efficiency of two algorithms vs. n_{FE}.

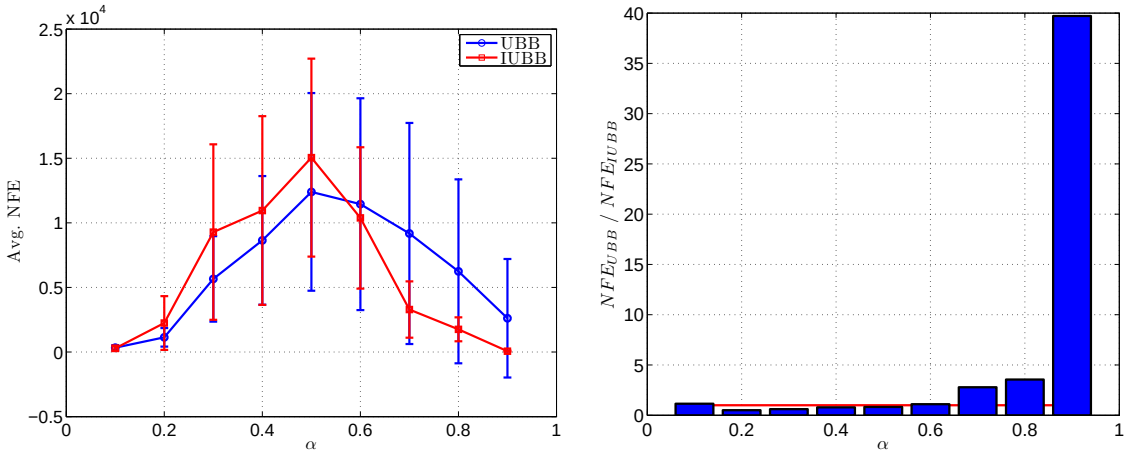


Figure 2.8: (a) Plot of the average function evaluations used by each algorithm to find the optimal feature sets, with standard deviation bars. (b) Barplot of the average gain in efficiency displayed by IUBB over UBB in terms of function evaluations required to find the global best feature set. (a,b) $n = 15$.

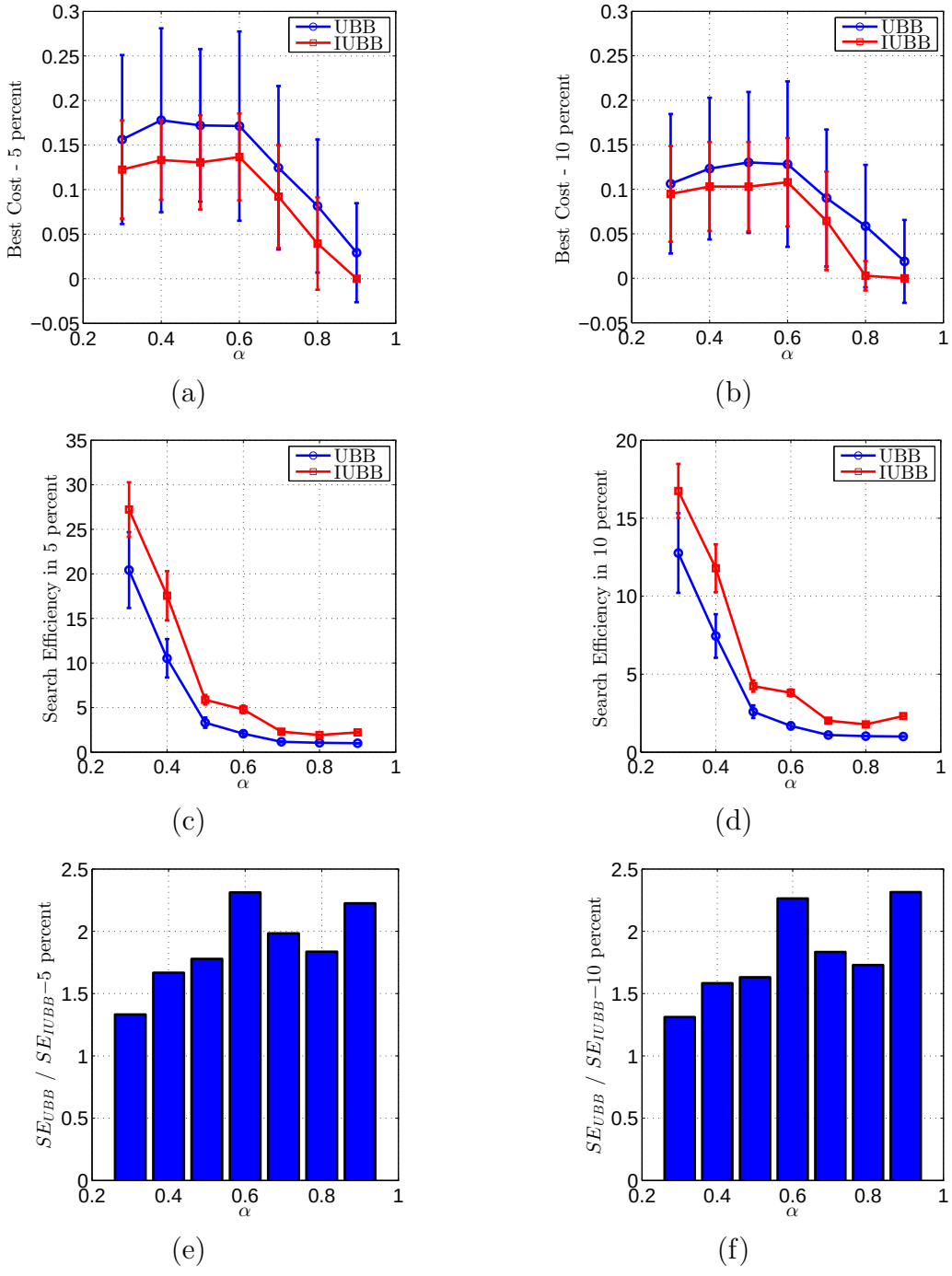


Figure 2.9: (a,b) Average best cost vs. α. (c,d) Search efficiency vs. α. (e,f) The ratio of search efficiency vs. α. (a,c,e) $n_{FE} = 2^n \times 5/100$. (b,d,f) $n_{FE} = 2^n \times 10/100$. (a-f) $n = 15$.

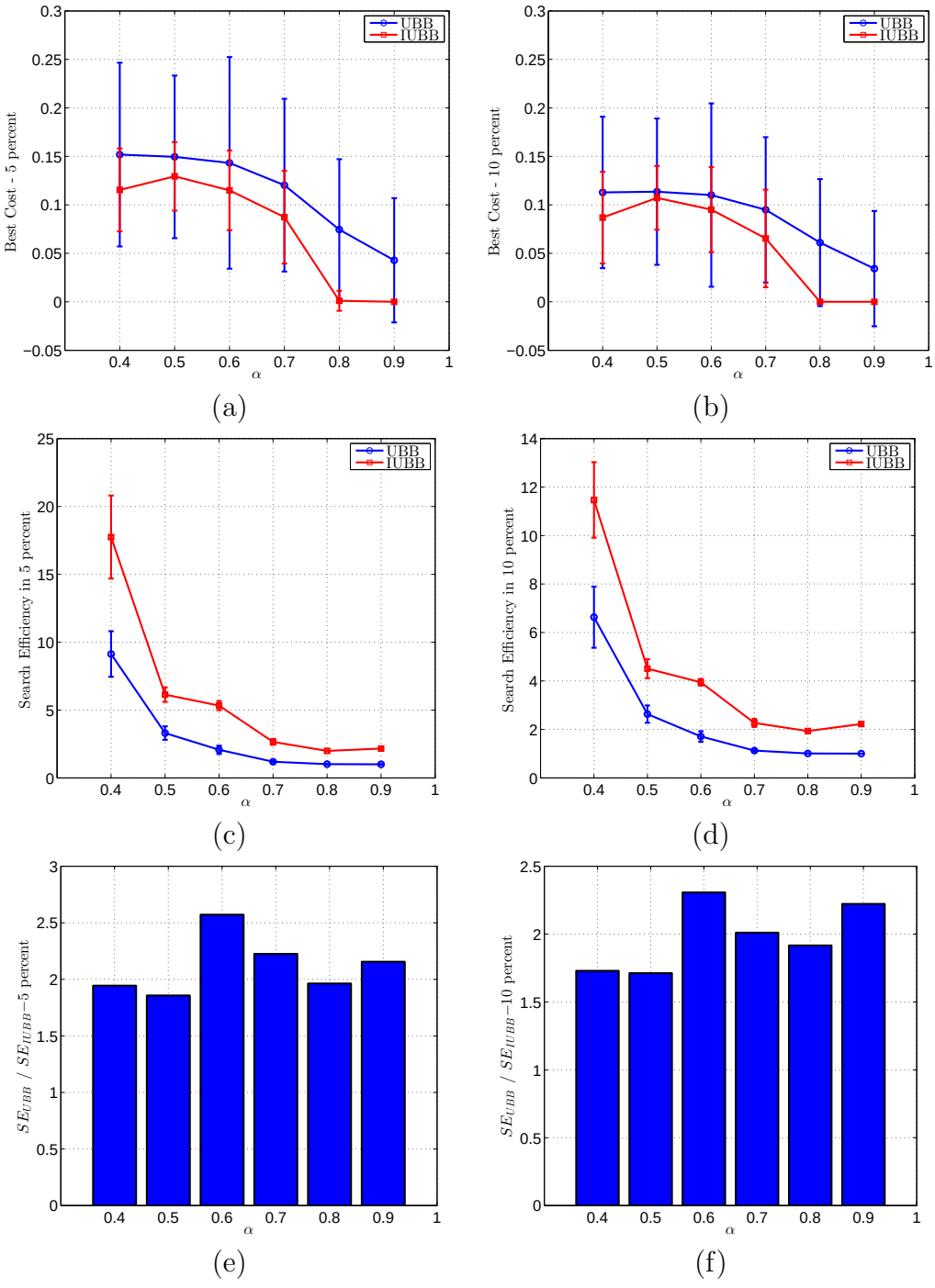


Figure 2.10: (a,b) Average best cost vs. α. (c,d) Search efficiency vs. α. (e,f) The ratio of search efficiency vs. α. (a,c,e) $n_{FE} = 2^n \times 5/100$. (b,d,f) $n_{FE} = 2^n \times 10/100$. (a-f) $n = 17$.

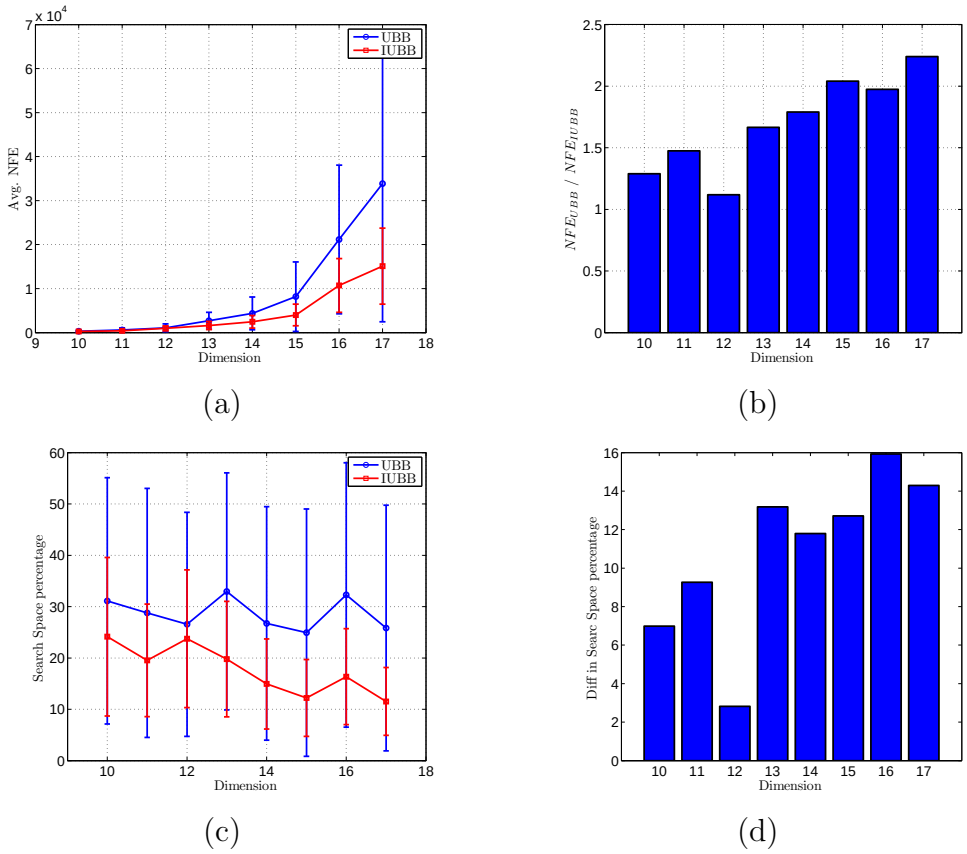


Figure 2.11: (a) Plot of the average function evaluations used by each algorithm to find the optimal feature sets, with standard deviation bars. (b) Barplot of the average gain in efficiency displayed by IUBB over UBB in terms of function evaluations required to find the global best feature set. (c) Average percentage of the search space evaluated by each algorithm to find the optimal feature sets, with standard deviation bars. (d) Difference between the average percentage of the search space evaluated by each algorithm to find the optimal feature sets, with standard deviation bars. (a-d) $\alpha = 0.7$.

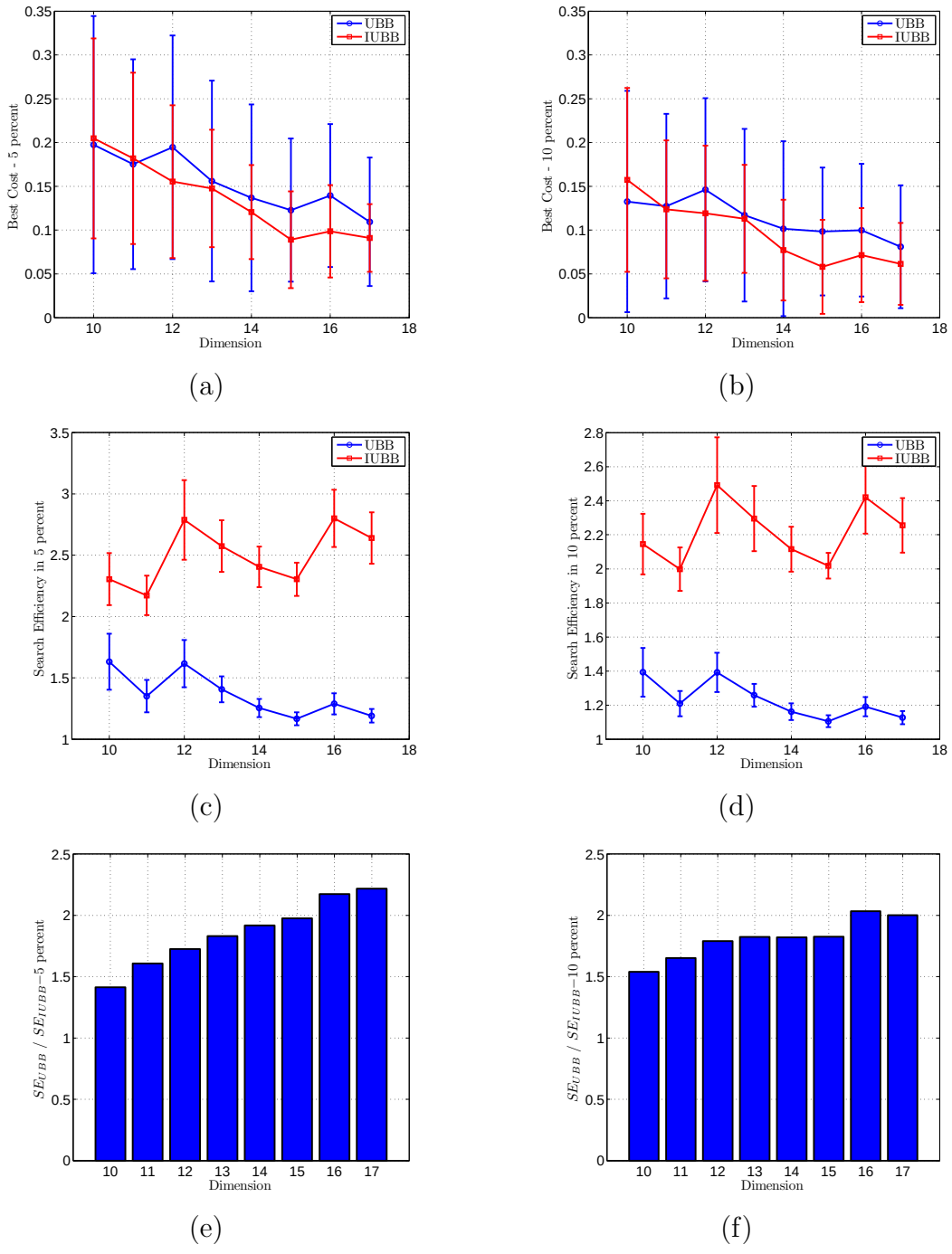


Figure 2.12: (a,b) Average best cost vs. α. (c,d) Search efficiency vs. α. (e,f) The ratio of search efficiency vs. α. (a,c,e) $n_{FE} = 2^n \times 5/100$. (b,d,f) $n_{FE} = 2^n \times 10/100$. (a-f) $\alpha = 0.85$.

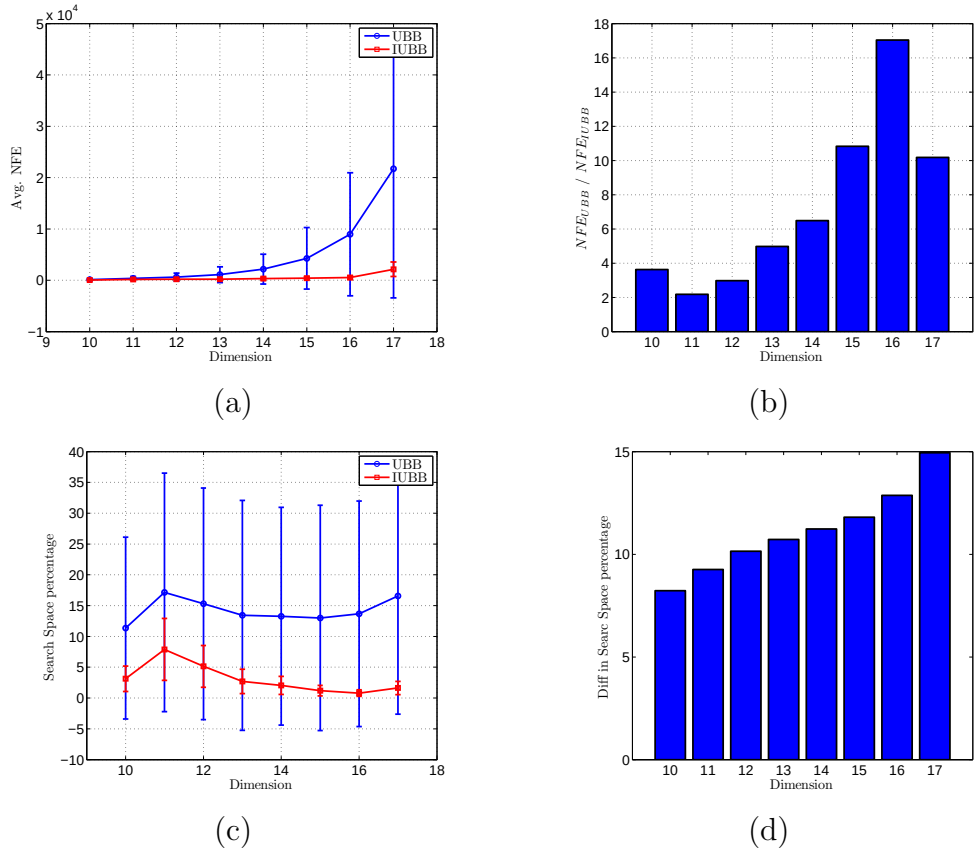


Figure 2.13: (a) Plot of the average function evaluations used by each algorithm to find the optimal feature sets, with standard deviation bars. (b) Barplot of the average gain in efficiency displayed by IUBB over UBB in terms of function evaluations required to find the global best feature set. (c) Average percentage of the search space evaluated by each algorithm to find the optimal feature sets, with standard deviation bars. (d) Difference between the average percentage of the search space evaluated by each algorithm to find the optimal feature sets, with standard deviation bars. (a-d) $\alpha = 0.85$.

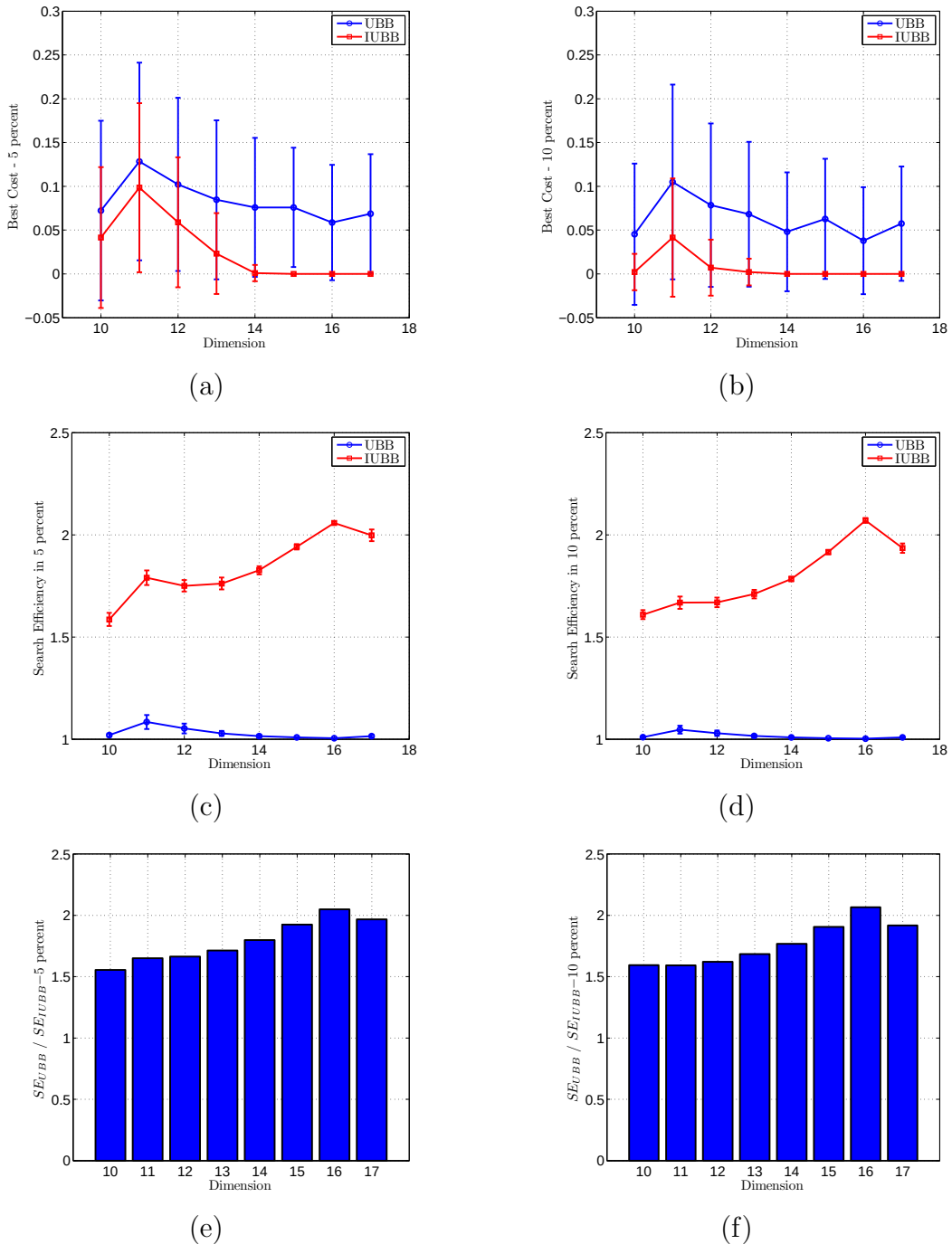


Figure 2.14: (a,b) Average best cost vs. α. (c,d) Search efficiency vs. α. (e,f) The ratio of search efficiency vs. α. (a,c,e) $n_{FE} = 2^n \times 5/100$. (b,d,f) $n_{FE} = 2^n \times 10/100$. (a-f) $\alpha = 0.85$.

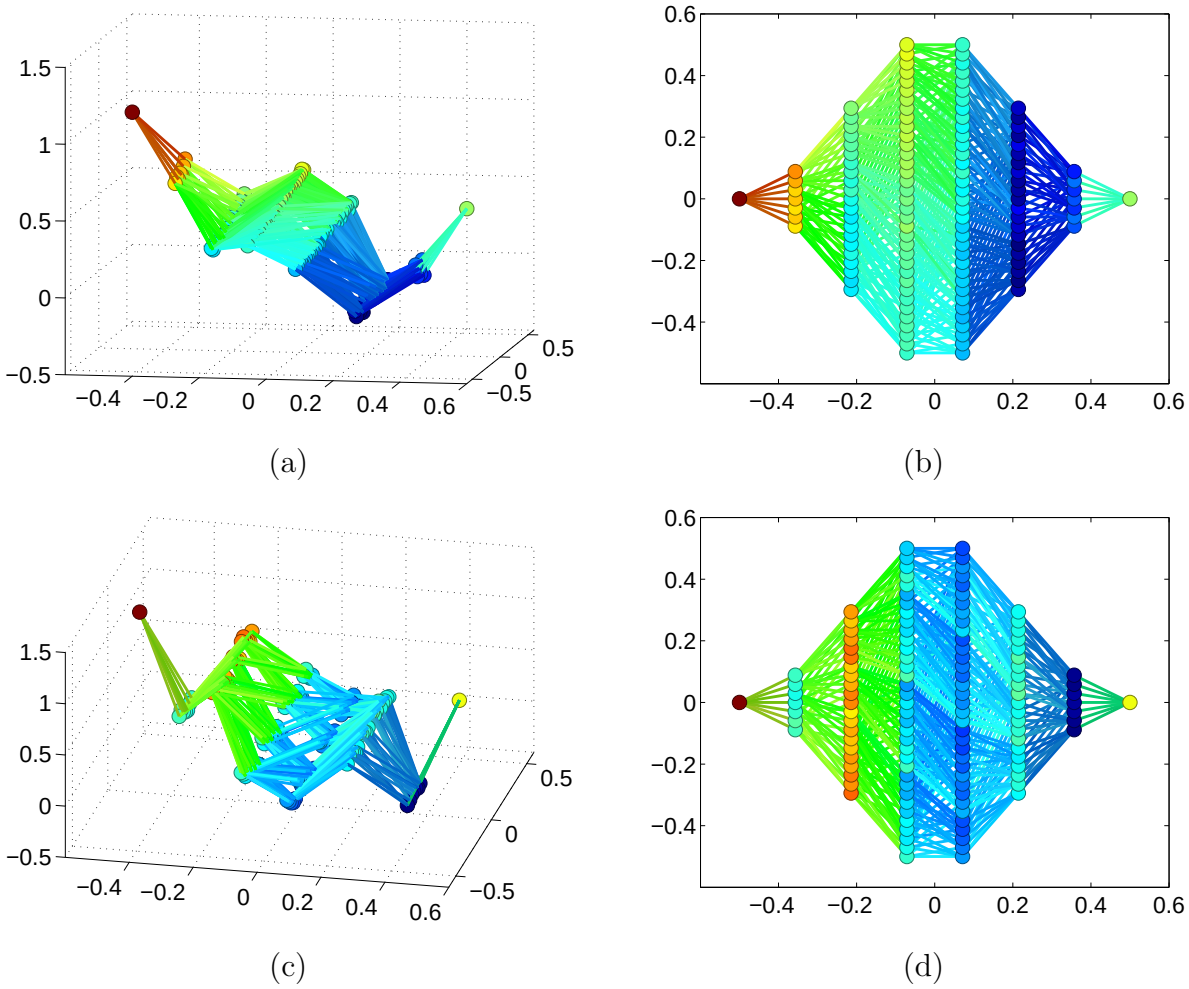
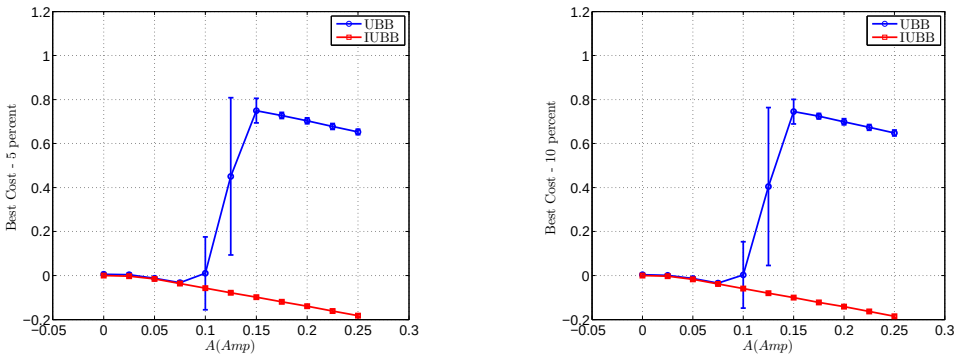
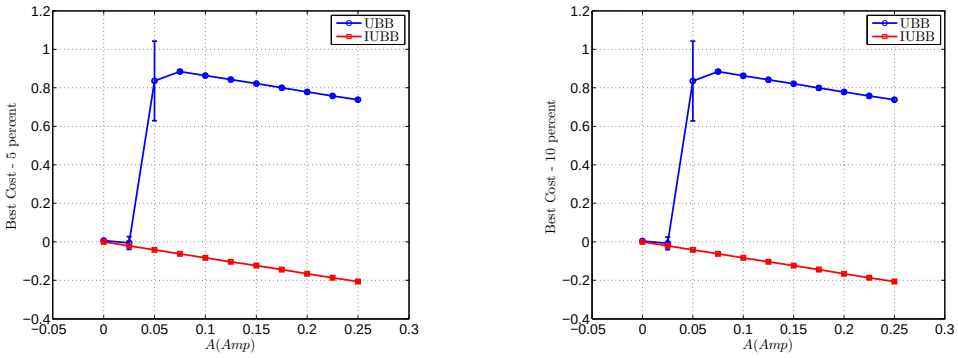


Figure 2.15: The u-curve problem model for $n = 7$ and $\alpha = 0.7$. (a and b) 3D and 2D representation of the cost for $A = 0.2$ and $f = 2$. (c and d) 3D and 2D representation of the cost for $A = 0.4$ and $f = 3$. Increasing A increases the depth of the local minimums. f control the frequency of the local minimums.



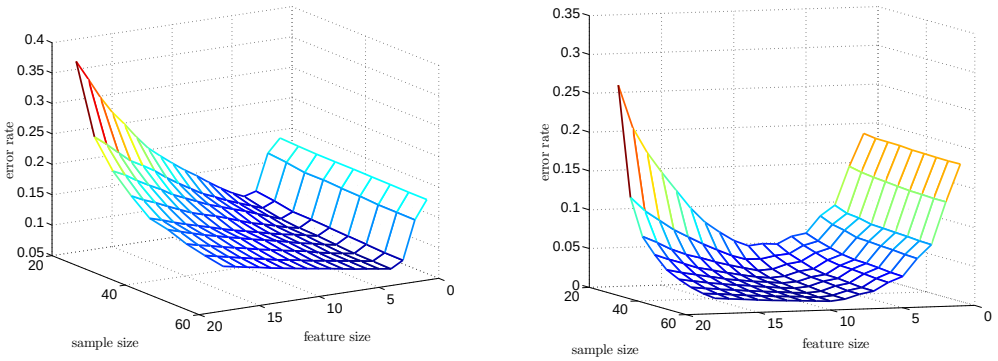
(a) (b)

Figure 2.16: Average cost vs. A. (a) n_{FE} is 5% of the search space. (b) n_{FE} is 10% of the search space. (a,b) $\alpha = 0.75$, $n = 15$ and $f = 2$.



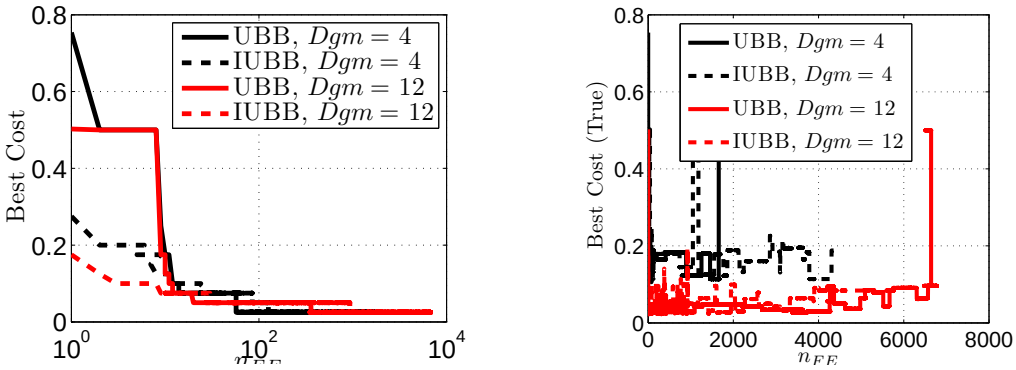
(a) (b)

Figure 2.17: Average cost vs. A. (a) n_{FE} is 5% of the search space. (b) n_{FE} is 10% of the search space. (a,b) $\alpha = 0.85$, $n = 15$ and $f = 3$.



(a) (b)

Figure 2.18: Peaking phenomenon for LDA classifier designed using the samples generated from the model



(a) (b)

Figure 2.19: Average best cost vs. n_{FE} (a) Estimated error used in feature selection (b) True error corresponding to the estimated error.

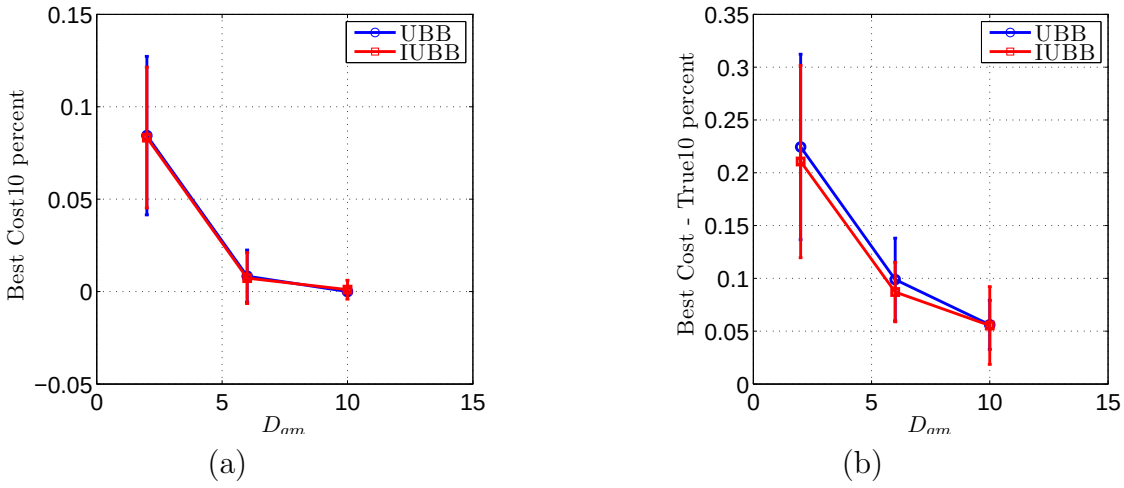


Figure 2.20: Average best cost vs. D_{gm} at 10 percent. (a) Estimated error used in feature selection (b) True error corresponding to the estimated error.

3. RELATIONSHIP BETWEEN THE ACCURACY OF CLASSIFIER ERROR ESTIMATION AND COMPLEXITY OF DECISION BOUNDARY*

Since the scientific content of any model depends on its predictive capacity, the most important attribute of any classifier is its error, that being the probability of misclassification. Since the feature-label distribution is generally unknown when designing a classifier, its error must be estimated by an error-estimation rule, so that the validity of the classifier model, consisting of both the classifier and its error, depends on the accuracy of the error-estimation rule. If the sample is large, one can split the data into training and test data, design the classifier on the training set, and estimate its error on the test set. In this case, there exists a distribution-free bound on the root mean square (RMS) error of the error estimator, namely, $RMS \leq 1/2\sqrt{m}$, where m is the number of data points in the test set [11]. When the sample is small, splitting the data results in poor classifier design, so that *data-efficient* error estimators must be used, i.e., error estimators that operate on the same data used for designing the classifier. Given an error-estimation rule, an obvious question concerns when it performs well, meaning that it has an acceptable RMS. This depends strongly on the error-estimation rule, the classification rule used to design the classifier, the feature-label distribution, and the sample size.

In this part of the dissertation we study the performance of the error estimators as the complexity of the decison boundary changes. A measure is defined for quantifying the complexity. This chapter is organized as follows. Section 3.1 describes in

*Parts of this section are reprinted with permission from "Relationship between the accuracy of classifier error estimation and complexity of decision boundary" by Esmaeil Atashpaz-Gargari, Chao Sima, Ulisses M Braga-Neto, Edward R Dougherty, 2012, *Pattern Recognition*, vol. 46, no. 5, © 2012 Elsevier.

detail the Beta Mixture Model (BMM) used in the chapter. Section 3.2 presents the simulation study results. A method to compute the complexity of a distribution is found in the section 3.3.

3.1 Model for Distributional Complexity

For clarity, we describe the model in two dimensions. The generalization of the model to three or more dimensions is straightforward. To begin, we define the configuration matrix

$$H = \begin{bmatrix} h_{11} & h_{12} & ... & h_{1m} \\ h_{21} & h_{22} & ... & h_{2m} \\ . & . & . & . \\ h_{m1} & h_{m2} & ... & h_{mm} \end{bmatrix},$$

where $h_{ij} \in \{0, 1\}$, and $i, j \in \{1, 2, ..., m\}$. Next, we consider the set of two-dimensional square cells of side $\ell > 0$,

$$D_{ij} = [(i-1)\ell, i\ell] \times [(j-1)\ell, j\ell] \,, \quad i, j \in \{1, 2, ..., m\} \,.$$

On each cell D_{ij}, we consider the joint density f_{ij} of two independent, identically distributed Beta random variables with parameters $a > 0$, $b > 0$:

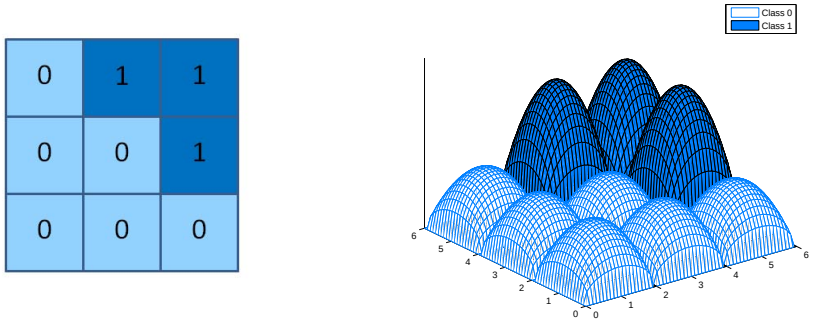
$$f_{ij}(x_1, x_2) = \frac{1}{C}(x_1 - (i-1)\ell)^{a-1}(i\ell - x_1)^{b-1}(x_2 - (j-1)\ell)^{a-1}(j\ell - x_2)^{b-1} \,,$$

for $(x_1, x_2) \in D_{ij}$, with $f_{ij}(x_1, x_2) = 0$ for $(x_1, x_2) \notin D_{ij}$, where $C > 0$ is a normalization constant to make the density integrate to 1. Now, for any given matrix H, let $R_k = \{(i, j) \,|\, h_{ij} = k\}$ for $k = 0, 1$. We define the class conditional density

$f_H(x_1, x_2 \,|\, k)$ for our model as the mixture

$$f_H(x_1, x_2 \,|\, k) \;=\; \frac{1}{|R_k|} \sum_{(i,j) \in R_k} f_{ij}(x_1, x_2)\,, \quad k = 0, 1\,.$$

The feature-label distribution is given by the combination of the two class-conditional densities: $f_H(x_1, x_2) \;=\; c_0 f_H(x_1, x_2 \,|\, 0) + (1 - c_0) f_H(x_1, x_2 \,|\, 1)$, where c_0 is the prior probability of class 0. We refer to this model as a Beta Mixture Model (BMM). An example is shown in Fig. 3.1.



(a) (b)

Figure 3.1: (a) Sample matrix H. (b) Class-conditional densities $f_H(x, x_2 \,|\, 0)$ and $f_H(x_1, x_2 \,|\, 1)$.

For any given matrix H, and corresponding feature-label distribution $f_H(x_1, x_2)$, the Bayes error is zero, and the data produced by the model are perfectly separable. This is crucial to our approach, as we want the complexity of the distribution to arise exclusively from the complexity of the Bayes decision boundary, which for this model consists of a combination of line segments and rays. See Figure 3.2 for an illustration. We define the *distributional complexity* $\chi(H)$ to be the total number of the line segments and rays in the Bayes decision boundary. For example, In Fig. 3.2, the distributional complexity is $\chi(H) = 5$. A simple method to compute

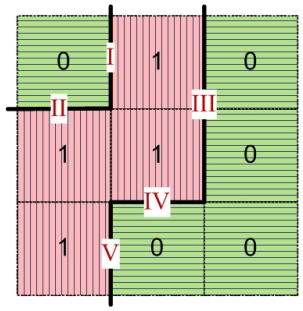


Figure 3.2: Example of Bayes decision boundary and decomposition by line segments and rays.

the complexity for any given configuration is presented in section 3.3.

The collection of all configurations of a given complexity χ will be denoted by M_χ. For example, with $\chi = 1$, i.e., a Bayes decision boundary consisting of a single line, we have

$$M_1 = \left\{ \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \right.$$
$$\left. \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix} \right\}.$$

Each configuration in M_1 leads to a distinct classification problem and decision boundary; however, from the perspective of the difficulty of classifier design or error estimation, complementation of the labels 0 and 1 and rotation produce equivalent configurations. This defines an equivalence relation, and our concern is with the equivalence classes $\tilde{M}_\chi$. For example, taking the standard approach of listing a

single representative of each equivalence class, for $\chi = 1, 2, 3$ we have

$$\tilde{M}_1 = \left\{ \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right\},$$

$$\tilde{M}_2 = \left\{ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} \right\},$$

$$\tilde{M}_3 = \left\{ \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \right\}.$$

3.2 Simulation Study

Based on the Beta Mixture Model and definition of complexity proposed in the previous section, we carry out a detailed simulation study to evaluate the impact of distributional complexity on the performance of different error estimation methods. We consider dimensionality $p = 2, 3$. In the case $p = 2$ we obtain results for each of the $2^8 - 2$ possible 3×3 configurations (leaving out complements and trivial cases), whereas in the case $p = 3$, results are obtained for a random sampling of the possible $3 \times 3 \times 3$ configurations at each level of complexity. In all cases, the cells have size $\ell = 1$, and the beta parameters are set to $\alpha = \beta = 1.5$. Results for other choices of parameters are similar, and can be found on the companion website at http://gsp.tamu.edu/Publications/supplementary/atashpaz12a. Three classification methods are employed: Quadratic Discriminant Analysis (QDA), 3-Nearest-

Neighbors (3NN) and two-layer Neural Networks (NNet). The error estimators considered in this study are resubstitution (resub), 10-fold cross validation with repetition (cv10r), leave-one-out (loo), bootstrap .632 (boot), and bolstered resubstitution (bolstrd). Following the recommendation of [13], we utilize in the 3NN case a variation of bolstering, called semi-bolstered resubstitution, whereas for QDA and NNet, the standard bolstered resubstitution estimator is employed. A description of all error estimators used in this study is given in section 3.3.

The resubstitution estimator is only plotted in the case $p = 2$, as its bias becomes disproportionate in the case $p = 3$, making the differences among the other estimators difficult to visualize if they are plotted together. We present here results for sample size $n = 60$; in addition, the companion website contains results for $n = 120$, which were observed to be similar, with the exception that true classification errors are smaller, as expected. The number of Monte-Carlo runs is set to 10,000 in each experiment. Based on this large number of repetitions, the expected true error of each classification rule is computed along with the bias and RMS of the several error estimators; these are used to analyze their performance as a function of distributional complexity.

3.2.1 Expected True Error

Before analyzing the performance of the error estimators, it is worth considering briefly the behavior of the true classification error. Figure 3.3 displays the true expected error of the various classification rules as a function of distributional complexity. As we see in the plots, the expected true error monotonically increases as the complexity increases. All the classification rules show good performance for low-complexity models. Interestingly, QDA outperforms 3NN and NNet for low-complexity models, but its performance quickly degrades as complexity increases.

This occurs because the simple structure of QDA makes it unsuitable for application to complex models. Across moderate and large complexities, 3NN is the best classification rule. This occurs because, compared to QDA, 3NN is more flexible and can create decision boundaries of different complexities. On the other hand, although NNet is capable of constructing complex decision boundaries, it requires large sample sizes for training, and this requirement can be seen to increase sharply with larger distributional complexity.

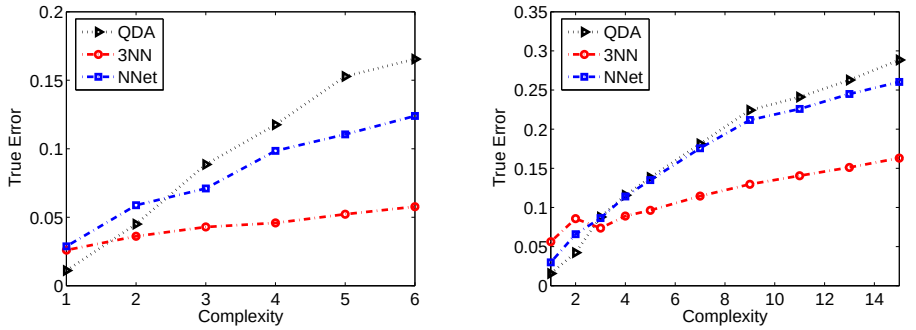


Figure 3.3: Expected true error of different classification rules vs. complexity, with $\alpha = \beta = 1.5$. (a) $p = 2$. (b) $p = 3$.

3.2.2 Performance of Error Estimators

Here we analyze the performance of the different error estimators, in terms of bias and RMS, as a function of distributional complexity, for the 3NN, QDA, and NNet classification rules.

3.2.2.1 3NN

Figure 3.4 displays the bias and RMS of the different error estimators for the 3NN classification rule. As mentioned previously, resubstitution is not plotted in the case $p = 3$ due to its disproportionate bias. We can see that performance degrades as distributional complexity increases. The RMS plot shows that semi-bolstered resub-

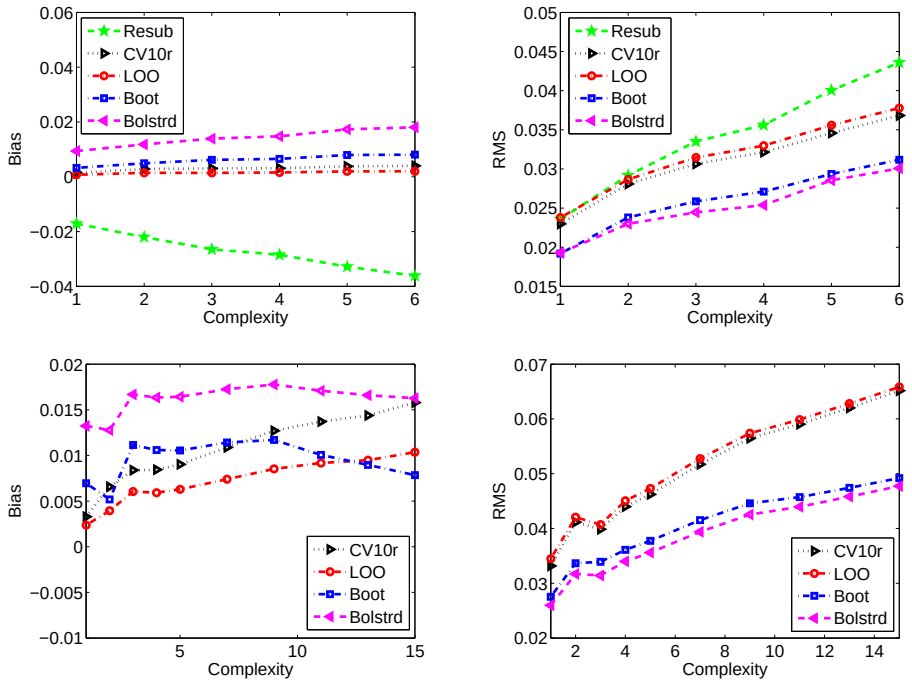


Figure 3.4: Performance of different error estimation methods vs. distributional complexity for the 3NN classification rule. Top plots: $p = 2$. Bottom plots: $p = 3$. Bias is shown on the left, whereas RMS is shown on the right.

stitution outperforms the other error estimation methods across the entire complexity range, followed closely by bootstrap. The cross-validation estimators cv10r and loo show acceptable performance for lower complexities. The worst error estimation method is resub, due to its bias, which rapidly increases as the complexity increases. Nevertheless, for low complexities, where the decision boundary is simple, the RMS of resub and the cross-validation estimators are essentially equal.

3.2.2.2 QDA

Figure 3.5 displays the bias and RMS of the different error estimation methods for the QDA classification rule. Once again, we observe that the performance of all error estimators become worse as distributional complexity increases. The bolstered resubstitution error estimator outperforms the others, followed closely by bootstrap,

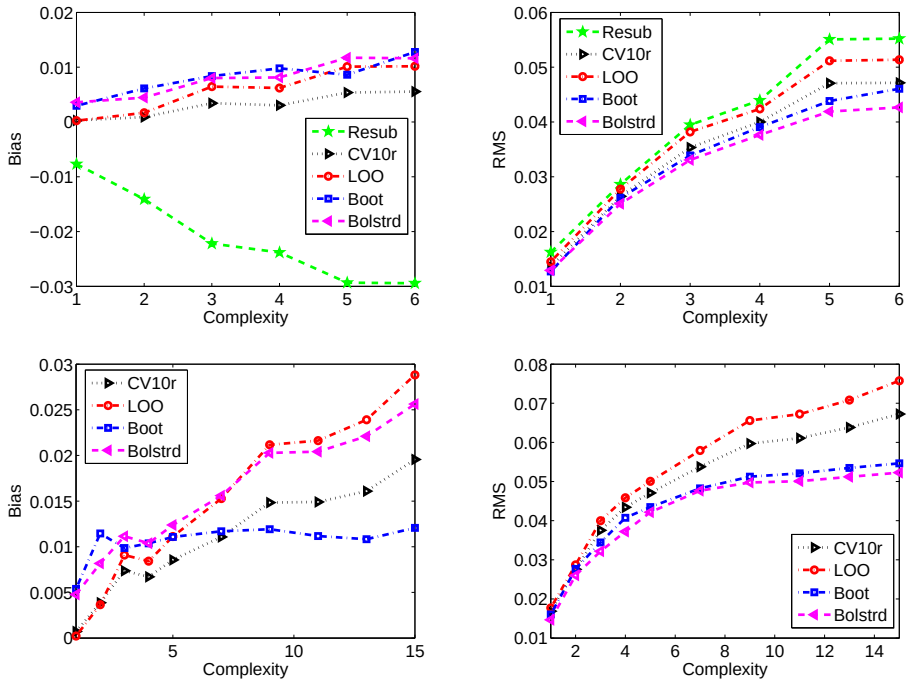


Figure 3.5: Performance of different error estimation methods vs. distributional complexity for the QDA classification rule. Top plots: $p = 2$. Bottom plots: $p = 3$. Bias is shown on the left, whereas RMS is shown on the right.

with cross-validation estimators lagging behind, even though they are the latter have the least bias. Resubstitution is extremely optimistically biased, which makes its RMS largest.

3.2.2.3 NNet

Figure 3.6 displays the bias and RMS of the different error estimation methods for the NNet classification rule. The general trends described earlier hold. Of note however is the very large bias displayed by resubstitution, which occurs due to the fact that NNet is more prone to overfitting than 3NN and QDA.

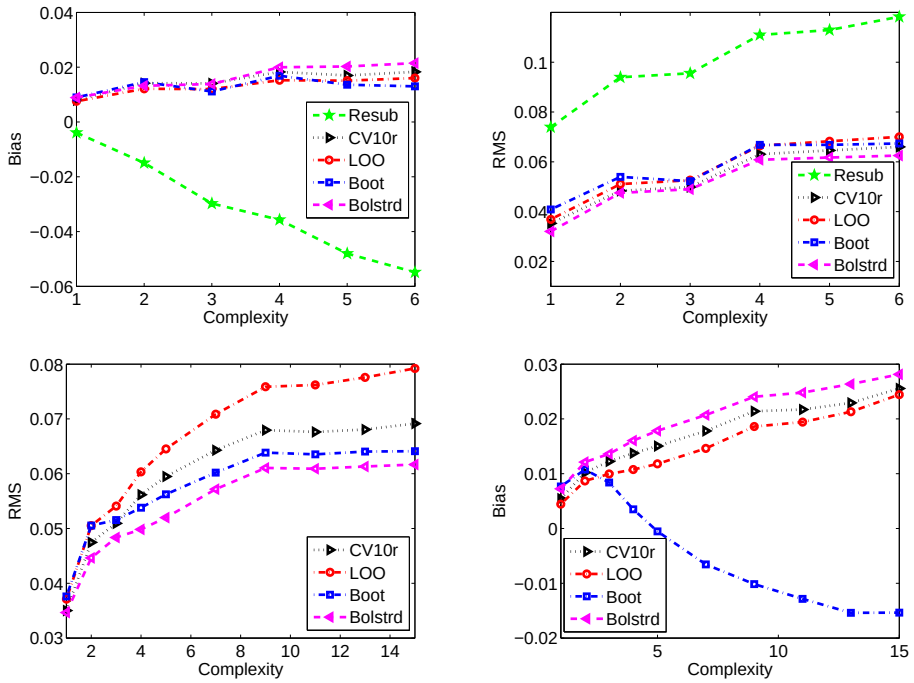


Figure 3.6: Performance of different error estimation methods vs. distributional complexity for the NNet classification rule. Top plots: $p = 2$. Bottom plots: $p = 3$. Bias is shown on the left, whereas RMS is shown on the right.

3.3 Complexity Computation

As mentioned in section 3.1, in the $p = 2$ case, the complexity of the distribution is defined to be the total number of lines, line segments and rays in the Bayes optimal piecewise linear classifier. In the $p = 3$ case, the Bayes piecewise linear classifier will be a combination of planes, half-planes, quarter planes and plane segments.

This can be extended to obtain the definition of complexity for general dimensionality $p \geq 2$ as follows. Let H be a p-dimensional configuration matrix of size $m \times m \times \cdot \times m$. Let L_j^i be a $(p-1)$-dimensional matrix obtained from H by fixing the i-th coordinate at value j, where $i \in \{1, \ldots, p\}$ and $j \in \{1, \ldots, m\}$. Define

$(p-1)$-dimensional matrices $D^d_{j,j+1}$ by:

$$D^i_{j,j+1} = L^i_j - L^i_{j+1},$$

where $i = 1, \ldots, p$ and $j = 1, \ldots, m-1$. The complexity $\chi(H)$ is the total number of connected components of nonzero terms in the $p \times (m-1)$ matrices $D^d_{j,j+1}$, where two cells are defined to be connected if they share a "face" in $(p-1)$-dimensional space.

These definitions can be best understood by means of examples in the $p = 2$ and $p = 3$ cases. We will assume in both cases that $m = 3$, as in the examples given in Section 3.1.

3.3.1 2D Case

In the $p = 2$ case, given a configuration matrix

$$H = \begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} \\ h_{2,1} & h_{2,2} & h_{2,3} \\ h_{3,1} & h_{3,2} & h_{3,3} \end{bmatrix},$$

the matrices $D^1_{1,2}$, $D^1_{2,3}$, $D^2_{1,2}$ and $D^2_{2,3}$ are given by:

$$D^1_{1,2} = \begin{bmatrix} h_{1,1} - h_{2,1} & h_{1,2} - h_{2,2} & h_{1,3} - h_{2,3} \end{bmatrix},$$

$$D^1_{2,3} = \begin{bmatrix} h_{2,1} - h_{3,1} & h_{2,2} - h_{3,2} & h_{2,3} - h_{3,3} \end{bmatrix},$$

$$D^2_{1,2} = \begin{bmatrix} h_{1,1} - h_{1,2} \\ h_{2,1} - h_{2,2} \\ h_{3,1} - h_{3,2} \end{bmatrix},$$

$$D^2_{2,3} = \begin{bmatrix} h_{1,2} - h_{1,3} \\ h_{2,2} - h_{2,3} \\ h_{3,2} - h_{3,3} \end{bmatrix}.$$

The complexity $\chi(H)$ is the total number of connected components of nonzero terms in the previous matrices. This is illustrated in Figure 3.7, with a matrix H for which $\chi(H) = 5$.

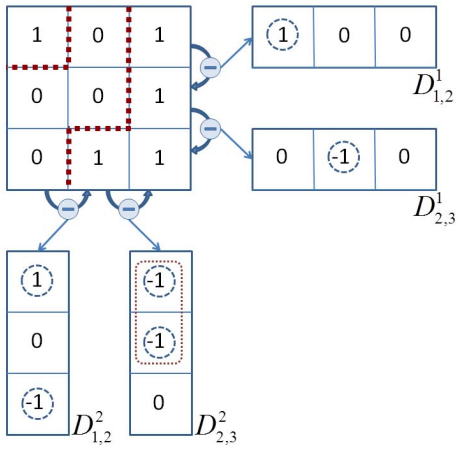


Figure 3.7: The process of finding the complexity $\chi(H)$ in the case $p = 2$. Here, $\chi(H) = 5$.

3.3.2 3D Case

In the $p = 3$ case, given a $3 \times 3 \times 3$ configuration matrix $H = [h_{ijk}]$, the matrices $D^1_{1,2}$, $D^1_{2,3}$, $D^2_{1,2}$, $D^2_{2,3}$, $D^3_{1,2}$ and $D^3_{2,3}$ are given by:

$$D^1_{1,2} = [h_{1jk} - h_{2jk}], \qquad D^1_{2,3} = [h_{2jk} - h_{3jk}],$$

$$D^2_{1,2} = [h_{i1k} - h_{i2k}], \qquad D^2_{2,3} = [h_{i2k} - h_{i3k}],$$

$$D^3_{1,2} = [h_{ij1} - h_{ij2}], \qquad D^3_{2,3} = [h_{ij2} - h_{ij3}].$$

The complexity $\chi(H)$ is the total number of connected components of nonzero terms in the previous matrices. This is illustrated in Figure 3.8, with a matrix H for which $\chi(H) = 9$.

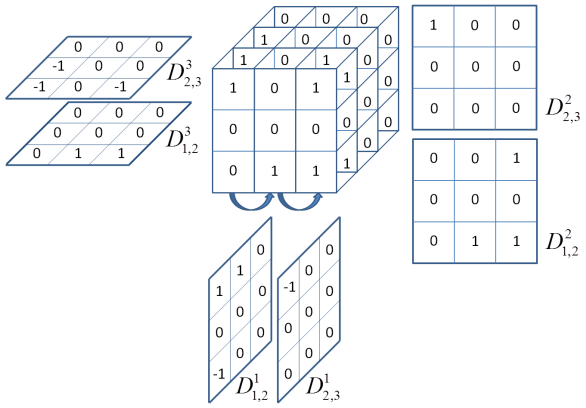


Figure 3.8: The process of finding the complexity $\chi(H)$ in the case $p = 3$. Here, $\chi(H) = 9$.

3.3.3 Discussion

The idea of using the grid-based configuration for modeling the complexity of decision boundary is taken from [14], where a definition of complexity is proposed based on the Bayes tree classifier designed for each configuration. There, the distributional complexity of a feature-label distribution is defined to be "the minimal number of hyperplanes necessary to achieve the Bayes classifier if the Bayes classifier is achievable by a finite number of hyperplanes, and in infinity otherwise". Whereas that former definition links the complexity to the characteristics of the Bayes tree classifier, our definition deals exactly with the final outcome of this classifier, which is the Bayes decision boundary. Figure 3.9 shows a configuration in the 2D case which has complexity 5 according to our new definition. Based on the previous definition, the complexity of this configuration is 4, as we need four hyperplanes to

make the Bays tree classifier. The measure defined in this part of the dissertation focuses on the decision boundary, assuming completely separable classes. Fixing the value of Bayes error to zero, this definition enables one to directly study the effect of the geometric complexity of the decision boundary on the performance of error estimators.

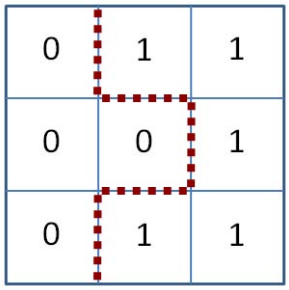


Figure 3.9: A sample configuration in 2D. Based on definition proposed in this chapter, the complexity of this configuration is 5, while Attoor's definition of complexity assigns 4 for this configuration.

4. MODELING AND SYSTEMATIC ANALYSIS OF BIOMARKER VALIDATION USING SELECTED REACTION MONITORING

Proteomics deals with the study of gene and cellular function at the protein level. Microarrays, 2D Gel Electrophoresis, and Mass Spectrometry (MS) are the most widely used technologies for high-throughput proteomics. Among these technologies, MS has increasingly become the method of the choice for analysis of complex protein samples [26]. Among its unique advantages are unsurpassed molecular specificity and very high detection sensitivity [27]. MS analysis is composed of thee major steps: 1) *Ionization*: Conversion of the analyte molecules or atoms into gas-phase ionic species. 2) *Mass Analysis*: Separation and mass analysis of ions on the basis of their mas-to-charge (m/z) ratio. 3) *Detection*: Detection and measurement of the mass-separated ions.

Time of flight (TOF), Linear quadrupole/3D-quadrupole ion trap, Fourier Transform Ion Cyclotron Resonance (FT-ICR) and Orbitrap are some of the main mass analyzers used in MS instruments. Application of two or more stages of mass analysis leads to Tandem Mass Spectrometry (MS/MS) which enables us to examine selectively the fragmentation of particular ions in a mixture of ions [28]. Selected reaction monitoring (SRM) is a specific mode of Tandem Mass Spectrometry, which is widely used for quantitative measurement of analytes present in complex mixture and for validation of low-abundance biomarkers. In this part of the work, we model the SRM-based biomarker validation pipeline. The proposed model is then used to study the effect of the different parameter setting on the overall performance of biomarker validation process.

4.1 Selected Reaction Monitoring (SRM)

For over 30 years, SRM has been the method of choice for doing mass spectrometry on small molecules in order to study drug metabolism. However, its application to protein identification and quantification was limited by the low mass range of the instruments used for metabolite identification. The introduction of the quadrupole instrument with extended mass range removed this restriction in the application of SRM for studying proteins and peptides [1, 29, 30]. Although SRM can be done on some of the other tandem MS instruments (e.g., EB- and BE-Magnetic sector tandem MS), it is preferably implemented on triple-quadrupole, due to low cost, linear mass scale, operational simplicity and straightforward scan laws. The first and third quadrupoles in the QQQ system act as mass filters to specifically select a predefined m/z values, controlled by direct-current (dc) and radio-frequency (rf) potentials. The second quadrupole in SRM operates as rf-only quadrupole passing all ions. In fact, this quadrupole acts as the collision induced dissociation (CID) unit. This is done in two steps: *collision activation* and *collisionally activated dissociation* and is performed in the *high-* and *low-energy* regimes. The later is the mode that is preferably implemented in quadrupole. One of the main disadvantages of CID over other ion activation and dissociation methods is that ion-dissociation efficiency gradually falls off as the precursor ion's weight increases.

A prototypical SRM experiment consists of three major steps. First, a list of candidate proteins is determined. The list of proteins of interest is determined based on previous knowledge from discovery studies and the scientific literature. The available information about the potentially relevant proteins (e.g., ProteinAtlas) can also be employed in this step. In the next stage, for each candidate protein, a set of proteotypic peptides (PTPs) should be identified and targeted to determine the presence

of the protein and to quantify it. PTPs of a specific protein should be able to uniquely identify that protein or one of its isoforms as well as have a good ionization efficiency. Moreover, their mass to charge ratio should be in the mass range of the MS instrument. Besides these general characteristics, in a quantitative experimental workflow, PTSs should be fully recovered in the sample preparation and also present good chromatographic behavior to reduce the chemical background [31]. Furthermore, post-translational and chemically induced modifications of the peptides should be taken into account. These types of peptide modifications are described in more detail in next section, where they form a part of the model for the SRM process. Along with experimental methods, computational tools are also used to select MS-observable peptides for proteins. In the third step, for each selected peptide, the fragment ions that can unambiguously represent the targeted peptide from others should be identified. Based upon the experiments on the QQQ instrument or data from previously done shotgun experiments, 2-4 fragment ions are selected for each PTP. For example, being integrated with PeptideAtlas [32], TIQAM [33] can be used in this step [34].

Determination of the pairs of m/z values for the first and third quadrupole is referred to as selection of a "*transition.*" [35]. The selection of transitions are of high importance for reaching high quantification accuracy red and different factors such as ionization and fragmentation conditions should be taken into account. Fragmentation conditions and specially the distribution of fragment ion intensities depends on the type of instrument and the operating parameters. In the QQQ system, singly charged y-type ions are the predominant type of fragments generated by CID in a linear collision cell, as b-type ions and doubly charged fragments are significantly less stable than their y-type N-terminal counterparts. [35, 36]. On the other hand, tryptic peptide ions are predominantly doubly or triply charged with one charge at

each terminus. Therefore, the single-charge fragments will generally have a larger m/z value than the precursor value. On the other hand, single-charged chemical background will produce fragments with smaller m/z than the precursor. Therefore, the selection of transitions for which fragments have larger m/z than the precursor is essential for transition selectivity and high signal to noise ratios [35].

In spite of the two narrow filtering stages in SRM, the selected transitions may not be specific for the peptide of interest in a complex sample. This lack of specificity can result in false quantification values for the targeted peptide. Several methods are used to validate selected transitions before using them in SRM. Spiking heavy isotope-labelled peptides to the sample, which match the sequence of the target peptide, can help distinguishing the effect of unspecific signals. However, the cost of using heavy labelled peptides is high for quantification of large number of proteins, and usually other methods (e.g., SRM-triggered MS/MS scanning) are used, but those are unable to validate the transitions for low-abundance proteins in the detection limit of SRM [35]. Figure 4.1 summarizes the main steps in an SRM experiment.

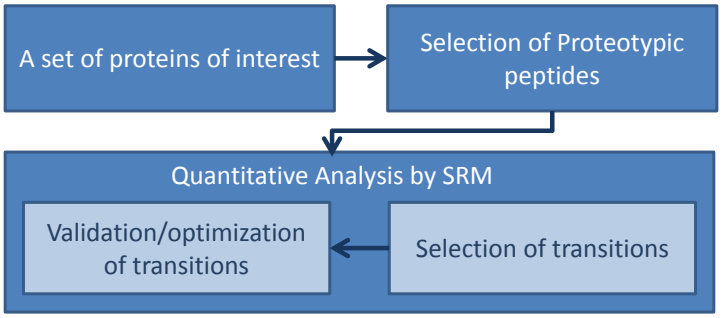


Figure 4.1: Workflow of an SRM experiment. First, a set of proteins of interest are determined for a specific study. Then, for each protein, some proteotypic peptides are found. In the next step, for each PTP, those fragments that are able to discriminate the peptide from others are found. The transitions (pairs of m/z values for precursor/fragment ions) are then validated to decrease the effect of unspecific signals.

4.2 Methods

In spite of the widespread application of SRM in the protein biomarker validation process, there is little work on integration of the different modules in SRM workflow and their systematic study for to assess the impact of different parameters on the overall biomarker validation pipeline. A model-based approach toward the SRM experiment will help us to have a better understanding of the characteristics of the different modules of the SRM-based biomarker validation process. Here, the SRM pipeline is modeled as a noisy channel affecting the underlying protein abundance signal; a model for the noise channel is proposed and used to analyze the effect of different parameters and experimental settings on the final performance of the SRM-based biomarker validation pipeline and the ability of SRM to detect true biomarkers among a set of candidate ones. Although the aim of the SRM model proposed here is not to determine the exact value of each parameter, it will be useful in providing a systematic view towards studying the individual components of the SRM experiment.

4.2.1 Protein Mixture Model

The first major component of the model is the *protein mixture model*. This part models the abundance of the proteins in the actual SRM experiment. *Marker* and *non-marker* proteins, as well as low-abundance and high-abundance proteins, are modeled in this part. The list of candidate biomarkers in the biomarker validation stage, enter the SRM pipeline described in the previous section. As mentioned previously, there are different sources of error in the SRM workflow that result in false quantification values for the protein abundance. The situation is exacerbated when dealing with low-abundance protein biomarkers. Background high-abundance proteins, inefficiency of peptide ionization, chemically induced modification and transition noise are the most widely quoted sources of error in SRM experiments [31, 35, 1].

In a typical experiment, the total set of samples are divided into two sample classes (e.g. *control* vs. *treatment*). There are a total number of N_a^{pr} proteins in the mixture, among which there are N_c^{pr} candidate proteins going through the validation stage ($N_a^{pr} > N_c^{pr}$). Based on the observations reported in [37], the protein concentration in the pooled sample can be modeled by a Gamma distribution[38].

$$\eta_i \sim \text{Gamma}(t, \theta), \qquad i \in \{1, 2, ..., N_a^{pr}\} \tag{4.1}$$

where t and θ are shape and scale parameters, and as an example $t = 2$ and $\theta = 1000$ present a realistic model with dynamic range of approximately 4 orders of magnitude.

As mentioned in Background section, many of the high-abundance protein biomarkers are already found by shotgun experiments and the focus of the SRM experiment is on validation of low-abundance candidate biomarkers. In order to model the concept of low-abundance and high-abundance proteins, we use two different Gamma distributed concentration models. For all the N_a^{pr} proteins, and $i \in \{1, 2, ..., N_a^{pr}\}$,

$$\eta_i \sim \begin{cases} \text{Gamma}(t_c, \theta_c), & i \in \{1, 2, ..., N_c^{pr}\} \\ \text{Gamma}(t_a, \theta_a), & i \in \{(N_c^{pr} + 1), (N_c^{pr} + 2), ..., N_a^{pr}\} \end{cases} \tag{4.2}$$

where t_c, θ_c, t_a and θ_a are are shape and scale parameters for candidate list and background proteins respectively. This reflects the nature of a real SRM experiment where the goal is to validate a set of low-abundance biomarkers among a complex set of high-abundance ones. We denote the number of true biomarkers in the set of N_c^{pr} candidate list and N_a^{pr} all proteins in the list by N_c^m and N_a^m respectively. The values of t_c, θ_c, t_a and θ_a are given in Table 4.1.

Biomarkers are proteins in the sample for which the expression level in the treatment and control sample differ significantly. The difference between markers and

non-markers in the expression level can be modeled by fold change [38]:

$$f_i = \begin{cases} a_i, & \text{if protein } i \text{ is over-expressed} \\ \frac{1}{a_i}, & \text{if protein } i \text{ is under-expressed} \\ 1, & \text{otherwise} \end{cases} \tag{4.3}$$

where the fold change parameter, a_i, is uniformly distributed in [1, h], $h > 1$. This results in a distribution that is approximately log-normal for the fold change itself. The value of h used in the simulations is specified in Table 4.1.

The sample variation of proteins in the mixture is modeled by a Gaussian distribution as proposed in [25], where a block model is used for the covariance matrix. The following multivariate Gaussian is used to model the concentration of the protein $i \in \{1, 2, ..., N_a^{pr}\}$ in class $j \in \{0, 1\}$ and the interaction among all the proteins in the sample:

$$C_{ij}^{pr} \sim \begin{cases} \mathcal{N}([\eta_1, \eta_2, ..., \eta_{N_a^{pr}}], \Sigma), & j \in \text{class } 0 \\ \mathcal{N}([f_1\eta_1, f_2\eta_2, ..., f_{N_a^{pr}}\eta_{N_a^{pr}}], \Sigma), & j \in \text{class } 1 \end{cases} \tag{4.4}$$

The covariance matrix Σ has a block structure, such that

$$\Sigma = [\sigma_{ij}^2]_{N_a^{pr} \times N_a^{pr}}$$

$$\sigma_{ij}^2 = \sigma_{ii}\sigma_{jj}\lambda_{ij} \tag{4.5}$$

$$\sigma_{ii} = \phi_i \times \eta_i$$

where the constant ϕ is the coefficient of variation and the correlation matrix Λ is

defined as:

$$\Lambda = [\lambda_{ij}] = \begin{bmatrix} R_\rho & 0 & \cdots & 0 \\ 0 & R_\rho & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_\rho \end{bmatrix}, \tag{4.6}$$

where R_ρ is a $b \times b$ matrix with 1's on the diagonal and ρ's elsewhere. The block-based structure of the covariance matrix represents the real interaction among the proteins. The proteins in each block (e.g., proteins within a pathway) are correlated, while there is no interaction among the proteins of different blocks [25]. The correlation ρ and block size b control the level of interaction among the proteins and their corresponding value used in simulations are specified in Table 4.1.

4.2.2 Sample Complexity and Purification

Many of the biomarkers with high abundance have already been found and the main interest in SRM-based biomarker validation process is in the quantification of low-abundance proteins. In biological samples, there is a wide dynamic range in protein abundance ($> 10^{10}$), which is much larger than the dynamic range of many MS instruments. For example, while interleukin has very low abundance, albumin makes up more than 50% (about 60%) of human plasma protein (30-50 g/L for albumin compared to below 100 pg/L for interleukin) [39].

Presence of high-abundance proteins interfering with the low-abundance ones biases the detection and quantification of biomarkers in complex samples. For example, due to suppression of their ionization by high-abundance proteins, low-abundance proteins escape detection. This makes purification and removal of high abundant proteins an important stage of biomarker validation workflow. Purification removes

background noise in the data, i.e. the nonspecific contributions of proteins not being evaluated as the candidate markers [40, 27]. There are different commercial and non-commercial options for the enrichment of samples for low-abundance proteins and the amount of energy that is put in this step greatly affects the overall performance of biomarker identification in the SRM process. For example, albumin precipitation, size exclusion, and immuno-depletion are strategies that have been developed to eliminate some of the most abundant proteins from blood serum. As an specific example, Seppro® IgY12, removes twelve high-abundance proteins from human biological fluids such as serum, plasma, and cerebral spinal fluid (CSF) [41].

In this part, we model purification by removing a set of high-abundance proteins from the protein mixture model. The parameter p_p controls the purification in the model by indicating the percentage of high abundance proteins that are successfully removed. Denoting the set of proteins selected for purification by $\mathcal{G}_p$, we have:

$$\hat{C}_{ij}^{pr} = \begin{cases} \gamma_i C_{ij}^{pr}, & \text{if protein } i \in \mathcal{G}_p \\ C_{ij}^{pr}, & \text{otherwise} \end{cases} \tag{4.7}$$

where $\gamma_i \sim U(0, \beta_\gamma)$. The value used for β_γ $(0 < \beta_\gamma << 1)$ in the simulations is given in Table 4.1.

4.2.3 Peptide Mixture Model

As mentioned in Background section, for each protein in the list of candidate biomarkers, a set of proteotypic peptides (PTPs) is identified and targeted to determine the presence of the protein and to quantify it. PTPs should uniquely identify the proteins, have good ionization efficiency, be fully recovered during sample preparation, and also present good chromatographic behavior to reduce the chemical background [31].

The molar concentration of C_i^{pp} of peptide i in each sample, in class j is given by

$$C_{ij}^{pp} = \sum_{k \in \Omega_i} \hat{C}_{kj}^{pr}, \quad i \in \{1, 2, ..., N_c^{pp}\}, j \in \{0, 1\} \tag{4.8}$$

where Ω_i is the set of all proteins sharing peptide species i and N_c^{pp} is the number of peptides. In an usual SRM experiment, for each protein, 1-2 PTPs are used. Denoting the number of peptides per protein by N_{pp}, then N_c^{pp} is equal to $N_{pp} \times N_a^{pr}$. In the results reported in this section of the work, we set $N_{pp} = 2$. In the ideal case, the cardinality of the set Ω_i is 1, that is C_i^{pp}, the concentration of peptide i, is related to only one protein. Equation (4.8) can be rewritten as following.

$$C_{ij}^{pp} = \sum_{k=1}^{N_a^{pr}} \xi_{ik} \hat{C}_{kj}^{pr}, \quad i \in \{1, 2, ..., N_c^{pp}\}, j \in \{0, 1\} \tag{4.9}$$

where for $i \in \{1, 2, ..., N_c^{pp}\}$ and $k \in \{1, 2, ..., N_a^{pr}\}$, ξ_{ik} is as following:

$$\xi_{ik} = \begin{cases} 1, & \text{Protein } k \text{ has peptide } j \\ 0, & \text{otherwise} \end{cases} . \tag{4.10}$$

In an ideal SRM experiment, each peptide is specific to one protein and then the peptide-protein relation matrix $\Xi = [\xi_{ik}]_{N_c^{pp} \times N_a^{pr}}$ has only one element equal to 1 in each row. In real SRM experiments, the complexity of the sample increases the possibility of having target peptides as a part of other proteins. To model this fact, we define s_i, the specificity of the ith PTP, as

$$s_i = 1 - P(\left|\Xi_i^1\right| \neq 1) \tag{4.11}$$

where $|S|$ shows the cardinality (the number of elements) of the set S and Ξ_i^1 is the

set of non-zero elements of the ith row of PTP-protein relation matrix Ξ. A peptide among the list of PTPs is called *specific* if its share in the sample is created by only its parent target protein. The specificity s_i of a specific PTP is then equal to 1. However, in real SRM experiments, this idealized situation does not occur and for some of the proteotypic peptides, the specificity will be less than 1.

There are many factors that should be considered in choosing the PTPs for each protein. For example, for each PTP, MS properties, uniqueness, and chemical behavior should be taken into account[35]. Increasing the number of proteins exacerbates the problem of finding PTPs that are specific to the target proteins and comply with other PTP selection criteria. On the other hand, we are not interested in the exact specificity value of each PTP, but rather want to observe the general effect of PTP specificity on the overall performance of biomarker validation process by SRM experiment. We thus define s as the average specificity over all peptides and study its effect on the identification of low-abundance protein biomarkers.

4.2.4 Peptide Ionization Efficiency

The abundance of a peptide is represented by the ion abundance in MS data. The abundance of a peptide i in class j is modeled by

$$\mu_{ij} = \kappa\, e_i\, C_{ij}^{pp}\,, \tag{4.12}$$

where e_i is the peptide efficiency factor, similarly to [42], and κ represents the instrument response factor, being the ratio between the ion current signal and the original analyte concentration.

The efficiency of different peptides in passing through the liquid chromatography column is mainly controlled by their hydrophobicity[27], followed by ionization efficiency, which is affected by sample complexity, peptide concentration, and char-

acteristics such as polarity of side chains, molecular bulkiness, and so on [43, 38]. Efficiency is also affected by the destabilizing effect of some amino acids at the N-terminal end of peptides. Some methods have been proposed for prediction of e_i for different peptides. However, these methods fail to address the complexity issue and dependence of the efficiency on not only the underlying peptide, but also on the other peptides present[38].

This makes the prediction of e_i for all the peptides problematic. Here, instead of the exact value of e_i, we are more interested in its effect on the overall performance of the SRM experiment. In the ideal case, e_i is 1 for all peptides. A model based on the uniform distribution $U(\alpha_{pe}, 1)$ models the variation of the peptide efficiency. The parameter α_{pe} controls the dispersion of the ionization efficiency and in the Results section, we analyze the model over a wide range to observe the effect of this parameter on the performance of the biomarker validation process.

4.2.5 Transition

In a complex sample, a particular precursor/fragment combination may not be specific to a targeted peptide, and other peptides with precursor/fragment ion pairs of similar masses might create unspecific signals. In the case that SRM is used to target low-abundant peptides, such unspecific signals might still be well above the detection limit and might be easily mistaken as being derived from the targeted peptide and thus lead to misquantifications [35]. Validation methods are used to ensure that the origin of the quantified signal is the targeted peptide. SRM-triggered MS/MS scanning is the method of choice in different studies. However, this method is challenging when used for the most low-abundance peptides [44]. Spiking heavy isotope labeled peptides into the sample is an alternative for the use of SRM-triggered MS/MS. But the costs of such method can be very high for projects targeting a large

number of proteins. In addition, the application of stable isotopes is limited by the resolution of the quadrupole as isotope labelling should introduce a sufficiently large mass difference between precursor and fragment ions[35]. Using smaller mass differences in isotope-labelling requires a higher resolution for the quadrupole, which in turn decreases the sensitivity. Low resolution has been reported in many papers as a source of error for SRM experiments using triple quadrupole mass spectrometers in complex samples[1].

The effect of transitions from background high-abundance peptides is considered as a significant source of error in quantification of the low-abundant peptides. Unspecific signals are created from other peptides with ion pairs of similar masses with the targeted peptide. By increasing the measured abundance of the targeted peptide, the unspecific signals create misquantification. Therefore, the noise is always positive. The exponential distribution is a simple and adequate choice to model this kind of unipolar additive noise:

$$\zeta_{ij} = \mu_{ij} + \epsilon_{ij}^{t}\,, \tag{4.13}$$

where

$$\epsilon_{ij}^{t} \sim \exp(\mu_{tran}\mu_{ij})\,. \tag{4.14}$$

4.2.6 Peptide Modification

Standard sources of error, including variation in experimental conditions, instrument variance, and thermal noise, can affect the accuracy of quantitative MS experiments. Besides these general factors, peptide modification is reported as one of the important causes of misquantification in SRM experiments [35].

Some peptides contain amino acids with high propensity to chemical modifica-

tions and can bias the quantification. Cysteine alkylation, methionine oxidation, asparagine deamidation, and N-terminal cyclization of glutamic are some of the chemically induced modification of peptides [31]. Oxidation, for example, is reported to inversely affect the performance of MS experiments for quantification of peptides [45]. Since a part of the targeted peptide is converted into the modified form during the process, chemically induced modification is reported to be a potential source of error in quantitative MS experiments [35, 31].

The Gaussian distribution is the standard model for the cumulative effect of independent additive disturbances (distinct noise sources). In [46], a Gaussian noise model with quadratic dependence of the variance on the expected abundance of peptide is used to model the overall effect of different noise sources affecting the actual abundance of a peptide in LC-MS. Likewise, we propose to use the Gaussian noise to model the effect of peptide modification as well as the other sources of error with significant impact on modifying the actual abundance of the peptide in SRM (LC-MS-MS). We have

$$\nu_{ij} = \zeta_{ij} + \epsilon_{ij}^{m}\,, \tag{4.15}$$

where

$$\epsilon_{ij}^{m} \sim \mathcal{N}(0, \alpha_{pm}\nu_{ij}^{2} + \beta_{pm}\nu_{ij})\,. \tag{4.16}$$

The two parameters α_{pm} and β_{pm} control the severity of the noise. In [46] a replication analysis is proposed to estimate the values of these two parameters. The values of α_{pm} and β_{pm} used in simulations are specified in Table 4.1. Having fixed β_{pm}, we will investigate the effect of α_{pm} on the performance of the biomarker validation in the next section.

4.3 Results and Discussion

The previous modelling strategy is used to analyze the performance of biomarker validation workflow using SRM experiments, using different model parameter settings. Figure 4.2 displays the simulation process. The list of candidate biomarkers generated based on the protein mixture model is the input of the SRM pipeline. In different stages of this process, the protein mixture data is affected by different noise sources depending on the experiment setting. Then, the output of the SRM process enters the validation block. Ranking power [47] and percentage of true biomarkers are used as the metrics to assess the performance of the biomarker identification process. The model parameters are changed during the simulation and for each parameter setting the average performance is found. The ranking power is described in the next section.

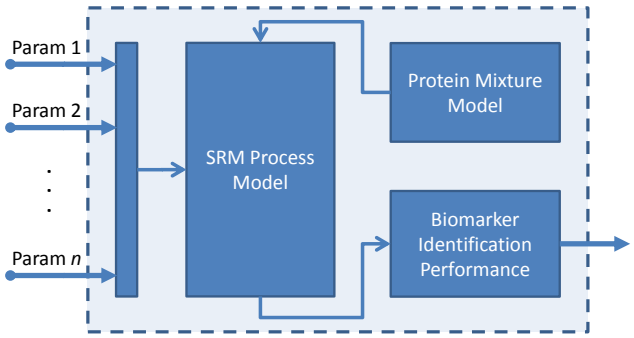


Figure 4.2: The entire simulation process. The protein abundance mixture data enters the SRM process and is affected by different noise sources in different levels of the process. The noisy data enters the biomarker validation block, where the ranking power and true positive rate are used to measure the performance of the overall biomarker validation process.

4.3.1 Experimental Setup

We perform a total of 5000 Monte-Carlo runs in this experimental study, using the parameter settings given in table 4.1, and compute average performance metrics over all the runs. The performance metrics used to evaluate SRM performance

Parameter	**Defaults value**
Number of classes	2
Sample size	$n = 80$
Block size	$b = 5$
Block correlation	$\rho = 0.8$
Fold change	$h = 2$, $a_i \sim$Unif$(1, 2)$
Modification noise	$\alpha_{pm} = 0.03, \beta_{pm} = 3.6$
Peptide efficiency factor	$\alpha_{pe} = 0.5, e_i \sim U(0.5, 1)$
Gamma parameters	$t_c = 2, \theta_c = 100, t_a = 5, \theta_a = 10e6$
Purification	$\beta_\gamma = 10e - 6$
Protein mixture	$N_a^{pr} = 250, N_c^{pr} = 40$
Ranking power	$d = 2$, $r = 0.01$

Table 4.1: Parameter settings in simulation of biomarker validation model

are the percentage of peptides correctly identified and the *ranking power* [47]. The former is computed by applying the t-test as a feature selection method to find the best discriminant set of features, and computing the ratio of true biomarkers detected in that list. The latter defines a measure of goodness based on how close the estimate-based feature sets are to optimality. Let A_{best} be the best feature set relative to the feature-label distribution, ε_0 be the true error of the classifier for A_{best} designed on the sample, and $A_{(1)}, A_{(2)}, ..., A_{(m)}$ be a list of feature sets ordered by the classification errors $\epsilon_1, \epsilon_2, ..., \epsilon_m$, sorted from lowest to highest. The ranking power of the list is defined by

$$\Delta_{D,d}^{n,r} = P(\epsilon_1 - \epsilon_0 < r)\,, \tag{4.17}$$

for $r > 0$. The ranking power gives the probability that at least one feature set in the list has error within r of the best feature set. The closer $\Delta_{D,d}^{n,r}$ is to 1, the better

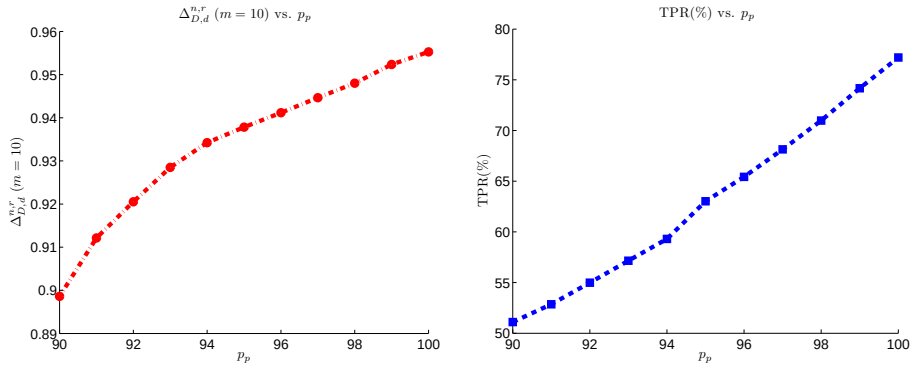
the performance is (as long as m is small; here, $m = 10$ is used). The pseudocode for computing the power rank is described in Algorithm 2.

Algorithm 2 Power rank computation algorithm

Set up data model M and determine A_{best}.
for $i = 1$ **to** N **do**
  1) Generate n-point sample T for M.
  2) Compute the true error, ϵ_0, for A_{best} using the samples from M.
  3) For every feature set of size d, design a classifier from T.
  4) Compute the true and estimated errors for the classifiers from step (3).
  5) Rank all the feature sets by their estimated errors to get the top m estimated-error list.
  6) Select the feature set in the list with the lowest true error, ϵ_1.
  if $\epsilon_1 - \epsilon_0 \leq r$ **then**
    $count \Leftarrow count + 1$
  end if
end for
$\Delta_{D,d}^{n,r} \Leftarrow count/N$
return $\Delta_{D,d}^{n,r}$;

4.3.2 Effect of Purification

Figure 4.3 displays the effect of purification on the performance of the SRM biomarker validation process. We can see that increasing the purification factor from 90% to 99% increases the ranking power by 7%. Increasing the purity from 90% to 99% translates into the increase of TPR from 50% to 80%. Although our purpose is not to focus on the exact value of each parameter in the model, the results show how purification is an important step in the SRM experiment. This confirms the fact that purification strategies, such as albumin precipitation, size exclusion, and immuno-depletion, directly control the accuracy of the SRM-based biomarker validation.



(a) (b)

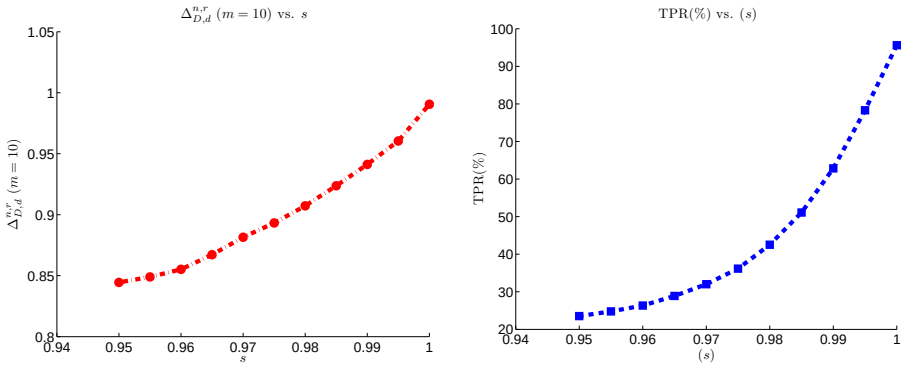
Figure 4.3: Effect of purification on the the SRM model on the performance of the biomarker validation pipeline. (a) $\Delta^{n,r}_{D,d}$ at list size $m = 10$ vs. purification. (b) TPR vs. purification.

4.3.3 Effect of Peptide Specificity

Figure 4.4 shows the effect of peptide specificity on the performance of SRM biomarker validation process. The results show that a very small amount of decrease in the specificity factor can bias the quantification of the low-abundance proteins to a great extent. For example, decreasing the specificity from 1 to 0.95 decreases the TPR by about 75%. These results indicate the importance of the selection of proper set of proteotypic peptides emphasizing on the fact that PTPs of a specific protein should be able to uniquely identify the protein (being specific peptides).

4.3.4 Effect of Peptide Efficiency

Although the exact distribution of the peptide efficiency is not known, observing its effect on the overall performance of the biomarker validation process provides us with a good insight into the effect of this parameter on the SRM experiment. This effect can be seen in Figure 4.5. The variation of peptide efficiency factor, α_{pe} (the lower bound of e_i), in the interval $[0, 1]$ changes the TPR by 6%, increasing it from 45% at $\alpha_{pe} = 0$ to 51% at $\alpha_{pe} = 1$. Based on the ranking power plot, we observe a



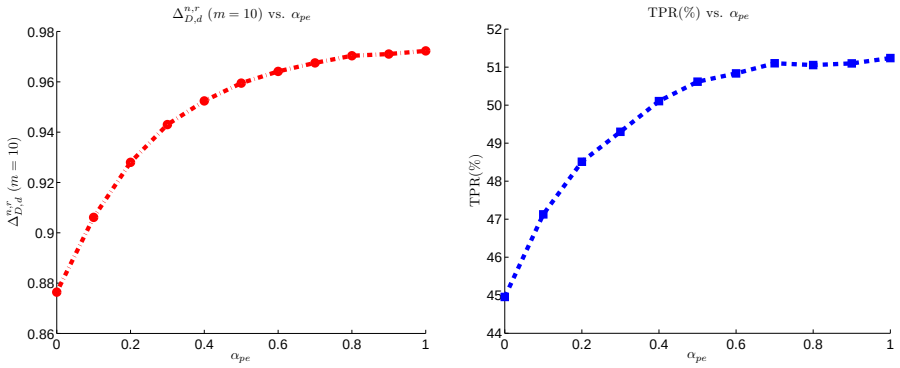
(a) (b)

Figure 4.4: Effect of peptide specificity on the the SRM model on the performance of the biomarker validation pipeline. (a) $\Delta^{n,r}_{D,d}$ at list size $m = 10$ vs. peptide specificity. (b) TPR vs. peptide specificity.

similar trend: $\Delta^{n,r}_{D,d}$ increases from 0.88 to 0.97 by increasing the peptide efficiency factor from 0 to 1. These results agree with our expectations as the increase of the peptide efficiency reduces the transmission loss.

4.3.5 Effect of Transition Noise

Figure 4.6 shows the effect of transition noise on the performance of SRM biomarker validation process. Both the ranking power and TPR curves show that an increase of the transition noise decreases the overall performance of the biomarker validation. For example the ranking power is 0.96 when the effect of this noise is set to zero. However by increasing the noise factor to 2, $\Delta^{n,r}_{D,d}$ reduces to 0.91. We observe a similar behavior, looking at TPR curve, where the rate decreases by 7% as the transition noise increases. This emphasizes the importance of applying the proper methods for validation of the transitions to increase the confidence on the origin of the quantified signal. Based on the experiment constraints, methods such as SRM-triggered MS/MS scanning and spiking of heavy isotope labelled peptides should be used to prevent the contribution of unspecific signals in the quantification of the proteins of



(a) (b)

Figure 4.5: Effect of peptide efficiency on the the SRM model on the performance of the biomarker validation pipeline. (a) $\Delta^{n,r}_{D,d}$ at list size $m = 10$ vs. peptide efficiency. (b) TPR vs. peptide efficiency.

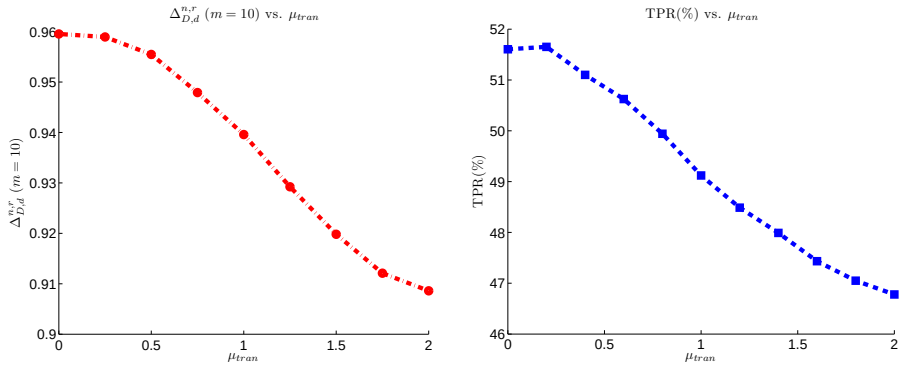
interest.

4.3.6 Effect of Modification

Figure 4.7 displays the effect of modification noise on the performance of the SRM biomarker validation process. Increasing the modification noise factor α_{pm} from 0 to 0.5 reduces the TPR value by 17%. On the other hand, the ranking power plot behaves the same by decreasing α_{pm} from 0.96 to 0.8. Decreasing the modification noise from 0.2 to 0 dramatically increases the ranking power value, emphasizing the fact that reduction of this source of error in quantification of the low-abundance biomarkers is crucial for a successful SRM experiment. This also shows that one should avoid using peptides with high tendency for chemical modifications in the list of PTPs.

4.3.7 Effect of Sample Size

Compared to the discovery stage of biomarker development, where thousands of analytes are measured, a validation experiment deals with the quantification of a limited list of analytes, meaning that the sample size requirement is less demanding.



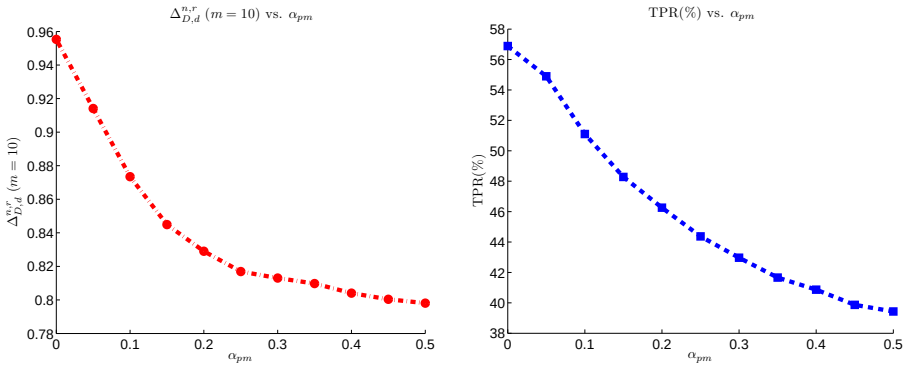
(a) (b)

Figure 4.6: Effect of transition noise on the the SRM model on the performance of the biomarker validation pipeline. (a) $\Delta^{n,r}_{D,d}$ at list size $m = 10$ vs. transition noise. (b) TPR vs. transition noise.

However, the time and cost of the experiment as well as the challenges of finding patients with correct demographics for the disease of interest, with proper medical history and lifestyle, still restricts the number of samples in a biomarker validation experiment to the "small-sample" region [48]. Observing the effect of the number of samples on the performance of the biomarker validation process will be beneficial to the selection of the right amount of replicates considering the limitations on the time and cost of the experiment. Figure 4.8 shows the effect of sample size on the performance of SRM biomarker validation process. Both TPR and ranking power plots show that these two performance indices are greatly affected by the increase of the sample size. Increase of the sample size from 40 to 100 results in 10% increase in the TPR value. The similar change in the sample size translates into the increase of ranking power value by 0.07.

4.3.8 Summary

General facts can be gleaned from the results reported above on the relative importance of each parameter to the sensitivity of biomarker validation performance



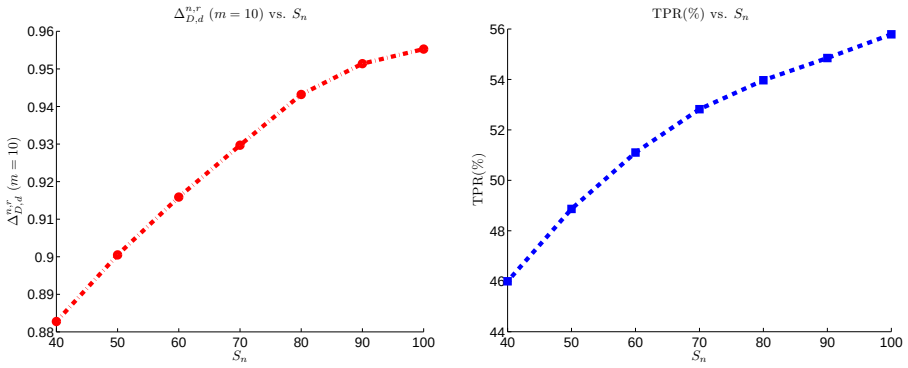
(a) (b)

Figure 4.7: Effect of modification noise on the the SRM model on the performance of the biomarker validation pipeline. (a) $\Delta^{n,r}_{D,d}$ at list size $m = 10$ vs. modification noise. (b) TPR vs. modification noise

using the QQQ-based SRM system.

- Purification critically increases the efficiency of the whole pipeline by reducing the background high-abundance proteins.
- On the other hand, peptide ionization efficiency also plays an important role in the success of biomarker validation experiment.
- A high value of modification noise can greatly compromise the performance of the system, as measured by the decreases of the TPR and ranking power value.
- Likewise, a decrease of peptide specificity reduces the TPR and ranking power to a great extent.

The results emphasize the importance of the correct selection of peptides in an SRM experiment. If the selected peptides are not unique to the targeted protein, it is hard to have high-precision quantification of the abundance of the targeted peptides, which will show itself in the unsuccessful protein validation results. An additional



(a) (b)

Figure 4.8: Effect of sample size on the the SRM model on the performance of the biomarker validation pipeline. (a) $\Delta^{n,r}_{D,d}$ at list size $m = 10$ vs. sample size. (b) TPR vs. sample size.

factor is of course sample size, which not surprisingly showed a clear effect on the performance of the biomarker discovery pipeline.

5. CONCLUSION*

As the first part of this dissertation, the U-curve feature selection problem was studied. UBB is a recently proposed branch-and-bound algorithm for solving the U-curve optimization problem. This algorithm requires fewer number of function calls compared to exhaustive search. However, the experiments and simulations showed that this algorithms fails in using the U-curve assumptions efficiently. In an attempt to overcome the weaknesses of this algorithm and to use the U assumption efficiently, we proposed a new optimization algorithm for feature selection.

To analyze the efficiency of the algorithms in facing feature selection problems with different structures, a feature selection cost model was introduced in this part of the work. This cost model, is used as a benchmark in this work for comparing the two algorithms. The parameters of the cost function turns it into a flexible realistic cost model for feature selection in small sample problems. This cost model can independently be used in any other study about comparing feature selection methods.

Different indices were used to evaluate the performance of the algorithms. The number of function calls needed to find the best feature set, the best cost vs. number of function calls and the search efficiency were three major indices used to compare the algorithms. Among these indices, search efficiency shows the efficiency of the two algorithms in using the U-curve assumption for finding the best feature set in the search space.

*Parts of this section are reprinted with permission from "Relationship between the accuracy of classifier error estimation and complexity of decision boundary" by Esmaeil Atashpaz-Gargari, Chao Sima, Ulisses M Braga-Neto, Edward R Dougherty, 2012, *Pattern Recognition*, vol. 46, no. 5, © 2012 Elsevier.

The results showed that the proposed algorithm outperforms the original UBB in terms of number of function evaluations needed to find the global best features. Also with the same number of function evaluations, IUBB generally reaches a feature set with lower cost value compared to UBB. The most important factor in comparing the two algorithms is the search efficiency. The results showed that IUBB has a higher value of search efficiency in almost all the number of function calls. Meaning that the algorithm is capable of searching more candidate solutions in the search space with a fixed number of cost function evaluations. This not only justifies the better performance of the algorithm in finding the feature sets with lower cost, but also indicates the level of trust on the results of the two algorithms.

To test the performance of the algorithms when the U-curve assumption does not hold completely, the cost model was modified to generate feature selection problems which are deviated from the U-assumption. The results show that IUBB is robust to the changes in the structure of the problem.

To evaluate the performance of the algorithms in dealing with real feature selection problems, the algorithms were used to solve the feature selection problem, where the cost value of each set of feature is the estimated error of the classifier designed for different sets of data generated based on the model described in section 2.3.6.1. The results showed that IUBB performs better than UBB. Also the we observed that accuracy of error estimation is crucial in having a successful feature selection.

In next step of this work, we developed a model for distributional complexity and studied the behaviour of different error estimation methods as a function of complexity for three different classification rules using the proposed model. The model was based on a mixture of beta distributions with a Bayes error of zero, so that only the complexity of the decision boundaries comes into play. Simulation results showed that the increase of distributional complexity leads to increasing degrada-

tion in error estimation performance. The best error estimator according to RMS was observed to be bolstered resubstitution, across the entire range of complexities considered, followed closely by bootstrap, with cross-validation estimators lagging behind. Resubstitution tends to present a dramatic increase in bias with increasing complexity, making this estimator unsuitable for complex distributions.

It has been observed and analytically demonstrated that cross-validation error estimation performance degrades with increasing Bayes error, in particular, for Gaussian class-conditional densities [49, 50]. The difference between degradation owing to increasing Bayes error and increasing distributional complexity can be clearly seen by considering optimal classification in the Gaussian model where both classes share the identity covariance matrix. In such a case, the Bayes decision boundary is a hyperplane equidistant between the means of the class-conditional densities. Moving the means closer at the same rate for both (the limit being Bayes error 0.5), the Bayes hyperplane remains fixed but the performance of cross-validation degrades. This corresponds to decreasing estimation performance while keeping the distributional complexity fixed at 1. In our proposed model, the Bayes error remains constant at 0, but the Bayes decision boundary increases in complexity. Clearly, the decreasing estimation precision is a consequence of very different factors in the two scenarios.

In the next step, the key components of the typical SRM-based biomarker validation work-flow were reviewed, modelled and analyzed. Based on the synthetic data the process was simulated and the effect of different parameter setting on the performance was studied. Ranking power and the TPR were used as two different metrics to assess the performance of the biomarker validation process as a function of the parameters of the model. The goal of this study was not determination of the exact value of each parameter for reaching a given performance value, but rather to investigate the effect of the different parameters, namely, sample purification, pep-

tide ionization efficiency, peptide specificity, modification noise, and sample size, on the overall performance of the SRM experiment utilized for biomarker validation.

The model presented here can not only be utilized to observe the effect of different instrument and experimental settings on biomarker validation by SRM, but also could be useful for experimental design, providing an insight on the working range of the important parameters of the SRM pipeline. It creates the required infrastructure for studying the inverse problem, where one can use the model to set the parameters of the entire experiment to reach the highest performance considering technical, experimental and financial constraints. Also the model has the advantage of being flexible to future possible extension in order to include more detailed modules of the SRM pipeline.

REFERENCES

[1] N.R. Kitteringham, R.E. Jenkins, C.S. Lane, V.L. Elliott, and B.K. Park. Multiple reaction monitoring for quantitative biomarker analysis in proteomics and metabolomics. *Journal of Chromatography B*, 877(13):1229–1239, 2009.

[2] G Hughes. On the mean accuracy of statistical pattern recognizers. *Information Theory, IEEE Transactions on*, 14(1):55–63, 1968.

[3] Chao Sima and Edward R Dougherty. The peaking phenomenon in the presence of feature-selection. *Pattern Recognition Letters*, 29(11):1667–1674, 2008.

[4] Anil Jain and Douglas Zongker. Feature selection: Evaluation, application, and small sample performance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(2):153–158, 1997.

[5] Patrenahalli M Narendra and Keinosuke Fukunaga. A branch and bound algorithm for feature subset selection. *Computers, IEEE Transactions on*, 100(9):917–922, 1977.

[6] Ari Frank, Dan Geiger, and Zohar Yakhini. A distance-based branch and bound feature selection algorithm. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 241–248. Morgan Kaufmann Publishers Inc., 2002.

[7] Songyot Nakariyakul and David P Casasent. Adaptive branch and bound algorithm for selecting optimal features. *Pattern Recognition Letters*, 28(12):1415–1427, 2007.

[8] BIN Yu and Baozong Yuan. A more efficient branch and bound algorithm for feature selection. *Pattern Recognition*, 26(6):883–889, 1993.

[9] Petr Somol, Pavel Pudil, and Josef Kittler. Fast branch & bound algorithms for optimal feature selection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(7):900–912, 2004.

[10] Marcelo Ris, Junior Barrera, and David C Martins Jr. U-curve: A branch-and-bound optimization algorithm for u-shaped cost functions on boolean lattices applied to the feature selection problem. *Pattern Recognition*, 43(3):557–568, 2010.

[11] L. Devroye, L. Gyorfi, and G. Lugosi. *A probabilistic theory of pattern recognition*, volume 31. Springer Verlag, Berlin, 1996.

[12] U.M. Braga-Neto and E.R. Dougherty. Is cross-validation valid for small-sample microarray classification? *Bioinformatics*, 20(3):374, 2004.

[13] U.M. Braga-Neto and E. Dougherty. Bolstered error estimation. *Pattern Recognition*, 37(6):1267–1281, 2004.

[14] S.N. Attoor and E.R. Dougherty. Classifier performance as a function of distributional complexity. *Pattern Recognition*, 37(8):1641–1651, 2004.

[15] E.R. Dougherty and U. Braga-Neto. Epistemology of computational biology: mathematical models and experimental prediction as the basis of their validity. *Journal of Biological Systems*, 14(1):65–90, 2006.

[16] C.A.B. Smith. Some examples of discrimination. *Annals of Human Genetics*, 13(1):272–282, 1946.

[17] V.N. Vapnik. *Statistical learning theory*. Wiley-Interscience, New York, 1998.

[18] R.O. Duda, P.E. Hart, D.G. Stork, et al. *Pattern classification*, volume 2. wiley New York, 2001.

[19] P.A. Lachenbruch and M.R. Mickey. Estimation of error rates in discriminant analysis. *Technometrics*, 10:1–11, 1968.

[20] B. Efron. The jackknife, the bootstrap and other resampling plans. In *SIAM Monograph No. 38, NSF-CBMS*, page 92, 1982.

[21] B. Efron. Bootstrap methods: another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979.

[22] B. Efron. Estimating the error rate of a prediction rule: improvement on cross-validation. *Journal of the American Statistical Association*, 78:316–331, 1983.

[23] I.H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, California, 2005.

[24] K.K. Murray, R.K. Boyd, M.N. Eberlin, G.J. Langley, L. Li, and Y. Naito. Standard definitions of terms relating to mass spectrometry. *International Union of Pure and Applied Chemistry Analytical Chemistry Division*, 2006.

[25] J. Hua, W.D. Tembe, and E.R. Dougherty. Performance of feature-selection methods in the classification of high-dimension data. *Pattern Recognition*, 42(3):409–424, 2009.

[26] R. Aebersold, M. Mann, et al. Mass spectrometry-based proteomics. *Nature*, 422(6928):198–207, 2003.

[27] C. Dass. *Fundamentals of contemporary mass spectrometry*, volume 16. John Wiley & Sons, Hoboken, New Jersey, 2007.

[28] E. de Hoffmann. Tandem mass spectrometry: a primer. *Journal of Mass Spectrometry*, 31(2):129–137, 1996.

[29] D. Zakett, RGA Flynn, and RG Cooks. Chlorine isotope effects in mass spectrometry by multiple reaction monitoring. *The Journal of Physical Chemistry*, 82(22):2359–2362, 1978.

[30] JD Baty and PR Robinson. Single and multiple ion recording techniques for the analysis of diphenylhydantoin and its major metabolite in plasma. *Biological Mass Spectrometry*, 4(1):36–41, 1977.

[31] S. Gallien, E. Duriez, and B. Domon. Selected reaction monitoring applied to proteomics. *Journal of Mass Spectrometry*, 46(3):298–312, 2011.

[32] E.W. Deutsch, H. Lam, and R. Aebersold. Peptideatlas: a resource for target selection for emerging targeted proteomics workflows. *EMBO Reports*, 9(5):429–434, 2008.

[33] V. Lange, J.A. Malmström, J. Didion, N.L. King, B.P. Johansson, J. Schäfer, J. Rameseder, C.H. Wong, E.W. Deutsch, M.Y. Brusniak, et al. Targeted quantitative analysis of streptococcus pyogenes virulence factors by multiple reaction monitoring. *Molecular & Cellular Proteomics*, 7(8):1489–1500, 2008.

[34] L. Malmström, J. Malmström, N. Selevsek, G. Rosenberger, and R. Aebersold. Automated workflow for large-scale selected reaction monitoring experiments. *Journal of Proteome Research*, 11, 2012.

[35] V. Lange, P. Picotti, B. Domon, and R. Aebersold. Selected reaction monitoring for quantitative proteomics: a tutorial. *Molecular Systems Biology*, 4(1), 2008.

[36] King Wai Lau, Sarah R Hart, Jennifer A Lynch, Stephen CC Wong, Simon J Hubbard, and Simon J Gaskell. Observations on the detection of b-and y-type ions in the collisionally activated decomposition spectra of protonated peptides. *Rapid Communications in Mass Spectrometry*, 23(10):1508–1514, 2009.

[37] Y. Taniguchi, P.J. Choi, G.W. Li, H. Chen, M. Babu, J. Hearn, A. Emili, and X.S. Xie. Quantifying e. coli proteome and transcriptome with single-molecule sensitivity in single cells. *Science*, 329(5991):533–538, 2010.

[38] Y. Sun, U. Braga-Neto, and E.R. Dougherty. A systematic model of the lc-ms proteomics pipeline. *BMC Genomics*, 13(Suppl. 6):S2, 2012.

[39] N.L. Anderson and N.G. Anderson. The human plasma proteome history, character, and diagnostic prospects. *Molecular & Cellular Proteomics*, 1(11):845–867, 2002.

[40] X. Xu and T.D. Veenstra. Analysis of biofluids for biomarker research. *PROTEOMICS-Clinical Applications*, 2(10-11):1403–1412, 2008.

[41] J.E. Bandow. Comparison of protein enrichment strategies for proteome analysis of plasma. *Proteomics*, 10(7):1416–1425, 2010.

[42] W. Timm, A. Scherbart, S. Böcker, O. Kohlbacher, and T.W. Nattkemper. Peak intensity prediction in maldi-tof mass spectrometry: a machine learning study to support quantitative proteomics. *BMC Bioinf.*, 9(1):443, 2008.

[43] N.B. Cech and C.G. Enke. Practical implications of some recent studies in electrospray ionization fundamentals. *Mass Spectrometry Reviews*, 20(6):362–387, 2002.

[44] P. Picotti, O. Rinner, R. Stallmach, F. Dautel, T. Farrah, B. Domon, H. Wenschuh, and R. Aebersold. High-throughput generation of selected reaction-monitoring assays for proteins and proteomes. *Nature Methods*, 7(1):43–46, 2009.

[45] J.M. Froelich and G.E. Reid. The origin and control of ex vivo oxidative peptide modifications prior to mass spectrometry analysis. *Proteomics*, 8(7):1334–1345, 2008.

[46] M. Anderle, S. Roy, H. Lin, C. Becker, and K. Joho. Quantifying reproducibility for differential proteomics: noise analysis for protein liquid chromatography-mass spectrometry of human serum. *Bioinformatics*, 20(18):3575–3582, 2004.

[47] C. Zhao, M.L. Bittner, R.S. Chapkin, and E.R. Dougherty. Characterization of the effectiveness of reporting lists of small feature sets relative to the accuracy of the prior biological knowledge. *Cancer Informatics*, 9:49, 2010.

[48] X. Ye, J. Blonder, and T.D. Veenstra. Targeted proteomics for validation of biomarkers in clinical samples. *Briefings in Functional Genomics & Proteomics*, 8(2):126–135, 2009.

[49] N. Glick. Additive estimators for probabilities of correct classification. *Pattern Recognition*, 10(3):211–222, 1978.

[50] A. Zollanvari, U. Braga-Neto, and E. Dougherty. Exact representation of the second-order moments for resubstitution and leave-one-out error estimation for

linear discriminant analysis in the univariate heteroskedastic gaussian model. *Pattern Recognition*, pages 908–917, 2011.