INTERACTIVE AND NONPARAMETRIC MODELING OF PREFERENCES ON AN ORDINAL SCALE USING SMALL DATA

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

LEVENT ERİŞKİN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
INDUSTRIAL ENGINEERING

DECEMBER 2015

Approval of the thesis:

INTERACTIVE AND NONPARAMETRIC MODELING OF PREFERENCES ON AN ORDINAL SCALE USING SMALL DATA

submitted by **LEVENT ERİŞKİN** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Industrial Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Murat Köksalan
Head of Department, **Industrial Engineering** _____

Prof. Dr. Gülser Köksal
Supervisor, **Industrial Engineering Dept., METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Cem İyigün
Industrial Engineering Dept., METU _____

Prof. Dr. Gülser Köksal
Industrial Engineering Dept., METU _____

Assoc. Prof. Dr. Banu Soylu
Industrial Engineering Dept., Erciyes University _____

Assist. Prof. Dr. Mustafa Gökçe Baydoğan
Industrial Engineering Dept., Boğaziçi University _____

Assist. Prof. Dr. Banu Lokman
Industrial Engineering Dept., METU _____

Date: 14.12.2015

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Levent ERİŞKİN

Signature :

ABSTRACT

INTERACTIVE AND NONPARAMETRIC MODELING OF PREFERENCES ON AN ORDINAL SCALE USING SMALL DATA

Erişkin, Levent
Ph.D., Department of Industrial Engineering
Supervisor: Prof.Dr. Gülser Köksal

December 2015, 170 pages

In this study, we consider learning preference structure of a Decision Maker (DM). Many preference modeling problems in a variety of fields such as marketing, quality control and economics, involve possibly interacting criteria, and an ordinal scale is used to express preference of objects. In these cases, typically underlying preference structure of the DM and distribution of criteria values are not known, and only a few data can be collected about the preferences of the DM.

For developing a preference model under such circumstances, we propose using nonparametric Statistical Learning approaches interactively. In particular, we employ Active Learning by asking a preference question to the DM at each step and try to reach a close approximation to the correct model in a small number of steps. Our experimental analysis proves that the proposed approach outperforms a

“naive” approach where subsequent questions are asked randomly. In the study, we also provide algorithmic recommendations for modeling different underlying value functions, if information is available about the form of the preference structure and/or distribution of criteria values.

This study can be regarded as a pioneering approach considering that Statistical Learning based approaches in the literature have been developed and tested based on a relatively large preference information and they do not interact with the DM in model developing process while Multi Criteria Decision Aid based approaches typically ignore interactions among the criteria, suffer from generalization ability, and have no concern about predicting equally good everywhere in the criteria domain.

Keywords: preference modeling, sorting, active learning, interactive approach, multi criteria decision aid

ÖZ

SIRALI ÖLÇEKTE, AZ VERİ KULLANARAK ETKİLEŞİMLİ VE PARAMETRİK OLMAYAN TERCİH MODELLEMESİ

Erişkin, Levent
Doktora, Endüstri Mühendisliği Bölümü
Tez Yöneticisi: Prof.Dr. Gülser Köksal

Aralık 2015, 170 sayfa

Bu çalışmada, Karar Verici (KV) tercih yapısının öğrenilmesi konusunu ele alıyoruz. Pazarlama, kalite kontrolü ve ekonomi gibi birçok farklı alandaki tercih modelleme problemleri, tercihlerin sıralı ölçekte ifade edildiği ve kriterlerin etkileşim içinde olduğu durumları içerirler. Bu gibi durumlarda genellikle KV tercih yapısı ile kriter değerlerinin dağılımı bilinmez ve KV tercihlerine yönelik az miktarda veri edinilebilir.

Bu tür problemlerde bir tercih modelinin geliştirilmesine yönelik olarak parametrik olmayan İstatistiksel Öğrenme tekniklerinin etkileşimli olarak kullanılmasını öneriyoruz. Özellikle, her adımda KV'ye bir tercih sorusu sormak ve az sayıda adımda en doğru modele ulaşabilmek için Aktif Öğrenme tekniklerini kullanıyoruz. Deneysel analizlerimiz, önerilen yaklaşımın müteakip soruları rassal olarak soran "sade" yaklaşıma göre daha başarılı olduğunu göstermektedir.

Çalışmamızda ayrıca, elimizde tercih yapısının formuna ve/veya kriter değerlerinin dağılımına ilişkin bilgi olması durumları için farklı tercih fonksiyonlarının modellenmesine yönelik olarak algoritmik tavsiyeler de sunuyoruz.

Çalışmamız; literatürde önerilen İstatistiksel Öğrenme tabanlı yaklaşımların büyük miktarda tercih verisi kullanılarak geliştirilmesi ve test edilmesi, model geliştime sürecinde KV ile etkileşime geçmemeleri; Çok Kriterli Karar Desteği yaklaşımlarının ise genellikle kriterler arası etkileşimi ihmal etmeleri, genelleme yeteneklerinin zayıf olması ve kriter bölgesinin her yerinde aynı tahmin başarısını elde etmeyi dikkate almamaları nedeniyle bu alanda öncü bir çalışma olarak değerlendirilebilir.

Anahtar kelimeler: tercih modellemesi, sıralı sınıflandırma, aktif öğrenme, etkileşimli yaklaşım, çok kriterli karar desteği.

To my children Ozan, Öykü and my beloved wife Serap

ACKNOWLEDGEMENTS

I would like to express my gratitude to Prof.Dr. Gülser Köksal for her support not only in this thesis study but also throughout my Ph.D. studies.

I also owe thanks to Assoc.Prof.Dr. Cem İyigün, Assist.Prof.Dr. Mustafa Gökçe Baydoğan and Assist.Prof.Dr. Banu Lokman for their valuable comments which helped me improve this thesis work.

At the end, I am deeply grateful to my wife Serap Şen Erişkin for her endless support and love. Without her, it would have been impossible to finish my Ph.D. studies. I would like to thank her and our children Öykü and Ozan for making my life a better one.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Many real life decision problems involve multiple criteria usually conflicting with each other. In the presence of multiple criteria, we may not talk about an optimal solution or best alternative since there often exists no dominant alternative outperforming others in terms of all criteria. Therefore, the Decision Maker (DM) needs to consider trading off the achievement of one criterion against another one. Thus, decision making under multiple criteria turns out to be a subjective task that depends on the preference structure of the DM.

When there are more than two criteria, the tradeoff issue gets more complex. In this respect, as the number of criteria to be considered increases, the decision to be made becomes more confusing for the DM. Multi Criteria Decision Aid (MCDA) offers several techniques to help confused DM make decisions in the presence of multiple criteria. MCDA basically deals with preference modeling, criteria aggregating and interactive problem solving. In all these applications, the main idea remains the same: explicitly or implicitly elicit preference structure of the DM in order to provide a decision support model that can be used in solving multiple criteria decision problems.

As part of MCDA, preference modeling aims at explicitly eliciting preferential system of the DM. Preference modeling is drawing a growing interest recently due to the fact that it became an imperative step in variety of areas. The alternatives considered in these decision problems range from projects to cars or candidate students to investment options. Especially if we are interested in making predictions about the DM's preferences with regard to multiple objectives, developed preference model provides a practical tool to achieve this. Making predictions for the preference of some alternative/solution is particularly important in marketing and manufacturing fields. In marketing, for instance, determining product features to maximize customer preferences requires estimation of the preference structure of potential customers. Additionally, as part of the marketing analysis, determining how much each feature contributes to overall preference (called part-worth) needs a robust preference modeling. In manufacturing, predicting which quality characteristic values result in an acceptable product, which ones cause rework or scrap is needed to design better product and processes. Restrictive experimental conditions and inadequate resources available for design of data collection experiments require careful and economical determination of the levels of product and process variables to be used in the design. If the aggregate response metric is the "quality" of a product, then the design points can be determined close to the quality characteristic levels that maximize the DM's preferences. Again, this process may require estimation of the DM's preference structure.

When criteria considered in the decision problem interact with (or depend on) each other, preference modeling task gets more challenging. In the simplest form, criterion set Y is preferentially independent of the remaining criterion set Z if the conditional preference structure in the $\mathbf{y}$ space given $\mathbf{z}'$ does not depend on $\mathbf{z}'$ (Keeney & Raiffa, 1993). More formally, this independence statement holds if and only if for some $\mathbf{z}'$,

$$(\mathbf{y}', \mathbf{z}') \succeq (\mathbf{y}'', \mathbf{z}') \Rightarrow (\mathbf{y}', \mathbf{z}) \succeq (\mathbf{y}'', \mathbf{z}) \quad \text{all } \mathbf{z}, \mathbf{y}', \mathbf{y}'' \tag{1}$$

Several different independence definitions have been made in the Multi-Attribute Utility Theory (MAUT) based on degree of dependency and if the consequences are certain or associated with probabilities. Refer to Dyer (2005) for a comprehensive review of these independence definitions. Without loss of generality, we will assume that there exist interacting criteria if preference relation in (1) does not hold for any subset of criteria for simplicity and use interaction and preferential dependency terms interchangeably.

Even though there is a general consent among researchers regarding the existence of interaction among criteria in real life decision problems, it is often ignored in applications. Marichal (2000) enumerates main reasons for ignorance as; lack of suitable tools to model them, absence of precise definitions for different types of interactions, complexity of some interactions and difficulty to detect one. Due to these reasons, most of the preference modeling strategies assume preferential independence among criteria, making modeling process relatively easygoing.

Nevertheless, interaction phenomenon is encountered quite commonly, even in simpler cases. Dolgun (2014) mentions several cases in the quality control field where criteria under consideration interact with each other. For instance, consider this example given by Marichal (2000): The problem involves evaluating students in statistics (St), probability (Pr), and algebra (Al). In this example, statistics and probability are assumed to be more important than algebra. Evaluations of four students are shown in Table 1. (Scale from 0 to 20):

Table 1. Evaluations of students in three subjects.

Student	**St**	**Pr**	**Al**
a	19	15	18
b	19	18	15
c	11	15	18
d	11	18	15

The DM is asked to rank students based on evaluations in three subjects. DM easily states that $a \succ c$ and $b \succ d$. On the other hand, DM realizes that other comparisons are not so evident since scores somewhat interlace. Hence, considering statistics and probability are substitutive, DM decides that a student being good at statistics is preferred to be better in algebra than probability. Additionally, if a student is not good at statistics, than it is better that (s)he is good at probability than algebra. These two preference statements reveal $a \succ b$ and $d \succ c$. Consequently, these preference statements propose that criteria expressed as evaluations in three subjects are not (preferentially) independent.

Most of the MCDA methodologies utilize value function approach and assume an underlying functional model. According to this approach, DM preference structure is compatible with the functional form adapted. For instance, UTA based methods assume an additive functional form that is believed to represent DM's preferential system. However, DM preference structure is usually unknown and adapting a functional form may lead to poor results. Additionally, even though proper functional (i.e. nonlinear) form is assumed for a preferential system having interactions among criteria, parametric functional models may fail to address complex interaction structures in high dimensions.

Many preference modeling problems in variety of fields such as marketing, quality prediction and economics, involve possibly interacting criteria and an ordinal scale is used to express preference of objects. In these cases, typically no information is available about the underlying preference structure, and only a few data can be collected about the preferences of the DM. For instance, preference information is usually obtained via on-line questionnaires in the marketing field. Before a respondent gets bored and leaves the questionnaire, only a limited number of questions can be posed. Hence, every single question is consumed thriftily. There are other cases where data collection process (or generating reference alternatives) is costly in terms of time or money. In quality engineering, for instance, we may need to produce expensive alternatives to generate a reference set, which will be useless after determining quality or preference of each product. Therefore, in order to conduct data collection or preference eliciting process affordably, we need to determine alternatives that will provide the most preference information.

Preference models where preferences are expressed in the ordinal scale have many real life applications. This kind of developed preference models are used for classification or sorting tasks. Some of these application areas can be summarized as follows (Zopounidis & Doumpos, 2002):

- **Pattern recognition:** Based on recognized attributes, subjects of interest are classified into predefined classes.
- **Human resources management:** Evaluating personnel based on their attributes such as skills, education, leadership etc. and promoting some of them or assigning to appropriate positions.
- **Marketing:** Classifying customer profiles and developing custom marketing policies for each group.
- **Economics:** Credit risk assessment, portfolio selection.
- **Education:** Selecting a subset of applicants for graduate program.
- **Medicine:** Diagnosis of diseases based on observed symptoms.

- **Quality management:** Sorting products into predefined quality groups based on considered attributes.
- **Evaluation of hotels:** Evaluating and assigning stars to hotels based on criteria of interest.

We can extend this list with many other applications. All these applications require preference models that are developed by obtaining preference information from the DM(s).

There are many studies proposed in the literature to deal with preference modeling problems in the ordinal scale, however, most of them require big amount of preference data for modeling and ignore interaction among criteria. Additionally, majority assume a known underlying preference structure, which is not a plausible assumption. Some others using small data suffer from poor generalization ability. Also most of them only aim to sort limited number of alternatives at hand and consider a subset of the alternatives for getting preference information. Hence, preference model developed based on the preference information obtained with respect to these alternatives is used to sort the rest. Consequently, there is no concern about predicting equally good everywhere in the criteria domain. These problematic issues have not been solved in the preference learning field.

In recent years there is a growing interest among Statistical Learning (SL) practitioners towards MCDA. In one perspective, both MCDA and SL methodologies aim to build robust models that will represent or explain the phenomenon of interest. In MCDA, phenomenon of interest is preferential system of the DM, while SL functions in a variety of domains depending on the type of data. Incorporating Machine Learning (MA) methodologies in it, SL is one of the major research areas in Artificial Intelligence (AI). AI is a popular field of study that comprise several major research areas, namely, machine learning/data mining, soft computing, evolutionary computation, knowledge engineering and

management, expert systems, symbolic reasoning, cognitive systems, etc. (Doumpos & Grigoroudis, 2013). Main focus of AI is to develop predictive decision systems and technologies that will model human brain. In this respect, finding the common ground in AI applications, SL researchers are more interested in MCDA field more than ever.

There are several applications in SL dealing with MCDA problems, mainly in cases where response (or preference) is expressed in categorical or ordinal scale. Doumpos and Zopounidis (2011) provide a comparative review regarding integration of these two fields, connections, similarities, differences and potential research areas. Emphasizing that similarities are obvious, they enumerate differences as follows (last three previously discussed by Waegeman et al., 2009):

- **Model interpretability:** It is important in MCDA that models developed be interpretable since MCDA not only aims at developing decision models but also integrating DM into modeling process so that DM perceives his/her preferential system. On the other hand, SL models usually focus on developing models of higher accuracy and present a “black-box” structure.
- **Data dimensionality:** SL applications usually require big amount of data, whereas MCDA methodologies assume that only a small reference set is available, in general.
- **Model validation:** Even though validation of the model developed is considered to some extent in MCDA by interacting with the DM, model validation is an important component of SL practices. Moreover, numerous techniques developed for validation process of the models.
- **The role of the DM:** In MCDA most of the time DM actively participates in model developing process, while SL assumes that only a training sample is available, hence, interacting with DM is not required.
- **Regularization:** SL puts special emphasis on generalization capability of the model developed, hence, considers trade-off between complexity and

performance of the model. In MCDA, however, regularization is not a big issue.

- **Data type:** In MCDA the data is characterized by criteria; qualitative or quantitative. SL, on the other hand, handles different types of data with more complex structures such as text, image or signal data.

From this discussion, it becomes evident that both fields have some strong capabilities with respect to developing predictive preference models. As Doumpos and Zopounidis (2011) imply, future research in integration of these two areas will evolve towards this bearing.

Considering all aforementioned shortcomings of the proposed approaches in MCDA, we utilize SL methodologies in preference modeling where preference is expressed in the ordinal scale and criteria interact with each other. Modeling strategy is based on obtaining holistic judgements from the DM regarding alternatives and adjusting subsequent questions based on the judgements gathered thus far, in an adaptive fashion. We start with a small reference set and employ nonparametric classifiers for model developing. Using nonparametric classifiers brings two advantages; firstly, we assume no functional form for the preferential system of the DM, hence, we do not suffer from erroneously adapting a wrong function. Secondly, nonparametric classifiers outperform their parametric counterparts in modeling complex data structures. In order to conduct modeling process in an adaptive way, we propose employing Active Learning (AL) techniques. In particular, we employ AL by asking a preference question to the DM at each step and try to reach a close approximation to the correct model in a small number of steps. AL is an application of semi-supervised Machine Learning (ML) where the learning algorithm iteratively queries “the Oracle” or user. The main rationale for using AL is that, usually we have abundant unlabeled data (those sinstances that do not have class information) at hand, whereas labeled data is scanty and labeling one is expensive. Thus, querying process is implemented so

that as much information as possible is obtained while as less unlabeled data as possible is queried, as mentioned in the marketing example. Consequently, preference modeling is structured as a learning process. Utilizing AL, we query the DM in an interactive way, thereby, DM is integrated into the model developing process. In this context, while utilizing strong features of SL in modeling complex structures, we also address the weak sides of SL criticized by Doumpos and Zopounidis (2011), in conjunction with preference modeling. As a consequence, this study can be regarded as a pioneering approach considering that SL based approaches in the literature have been developed and tested based on a relatively large preference information and do not interact with DM in model developing process while MCDA based approaches ignore interactions, suffer from generalization ability, and have no concern about predicting equally good everywhere in the criteria domain.

This thesis is organized as follows: In Chapter 2, we provide literature review and background. In Chapter 3 we present our proposed approach. In Chapter 4 experimental design and analysis performed in order to evaluate our proposed approach are explained in detail. Chapter 5 explains extension of the algorithm where we consider input distribution of the criteria. In Chapter 6 we present conclusion remarks and future work.

CHAPTER 2

LITERATURE REVIEW AND BACKGROUND

In decision making problems involving multiple criteria, the DM is presented a set of *A* potential actions that (s)he needs to consider. Potential actions might be comprised of a discrete set of alternatives where each alternative is described by a criterion set. On the other hand, there are cases where number of potential actions might be infinite. In this case, there exists a region in which all feasible alternatives lie and each point in this region corresponds to a potential action. In MCDA, the first problem type is defined as discrete decision making problem while problems of second type are called continuous decision making problems.

When the DM faces a discrete decision making problem, the analyst can provide a decision aid by utilizing four different types of analyses. In MCDA, these analysis types are referred to as decision making problematics and can be classified as follows (Roy, 1996; Doumpos & Zopounidis, 2002; Figure 1)

- **Choice:** Best or limited set of best alternatives are identified.
- **Ranking:** Alternatives are rank-ordered from the most preferred to the least preferred.
- **Classification/Sorting:** Alternatives are put into predefined groups based on degree of preference.

- **Description:** Alternatives are described based on their distinguishing features that are explained with criterion set.

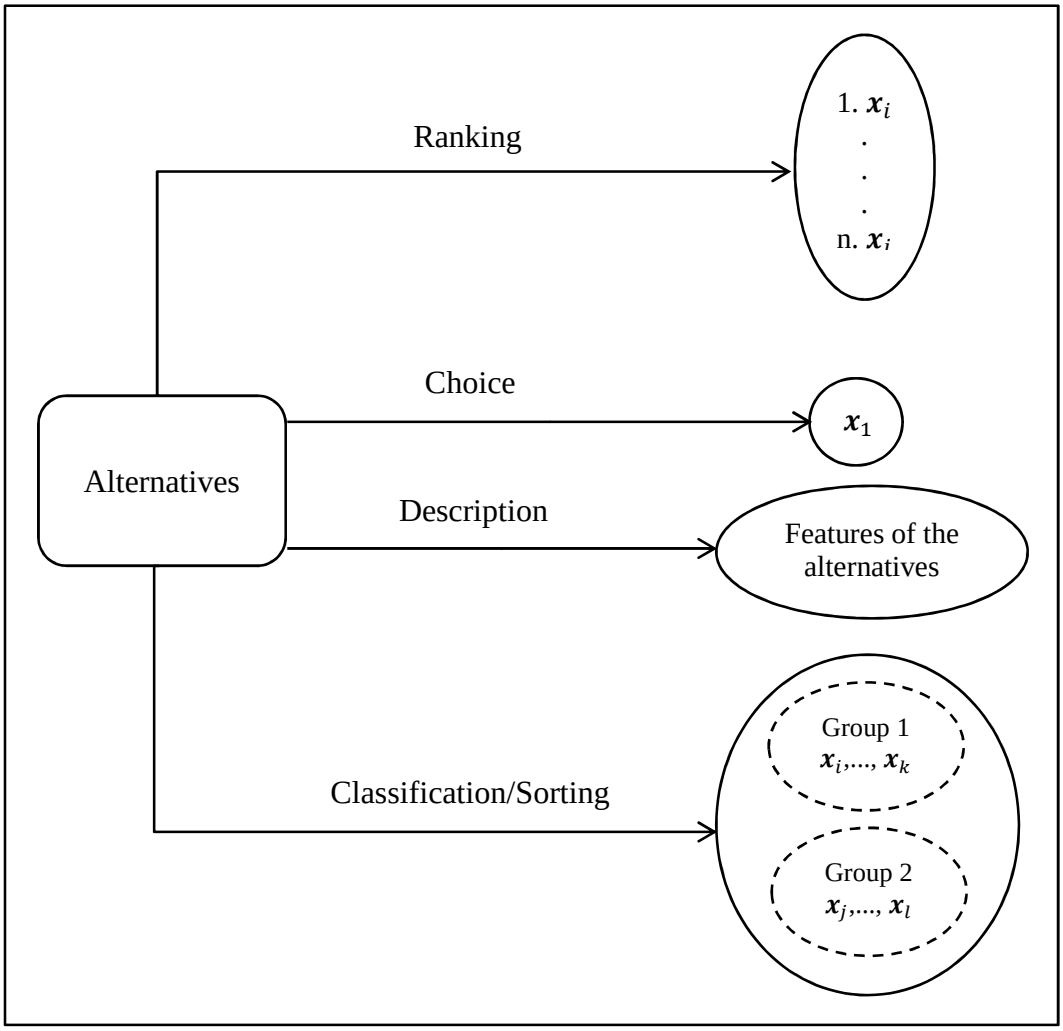


Figure 1. Decision making problematics (Source: Doumpos & Zopounidis (2011)).

In general, the first three problem types lead to a particular result. The first two problem types (choice and ranking) involve relative judgements, meaning, results

are expressed taking into account the other alternatives under consideration. If we add new alternatives to the set, the decision made by the DM might change. On the other hand, in the classification/sorting problem preferences made by the DM are absolute, based on the pre-defined groups. In manufacturing industry, for instance, a product undergoes a quality control process and labeled as “accepted”, “rejected” or “rework” based on some attributes under consideration. This judgement does not differ with respect to the other alternatives (or products) to be classified/sorted.

Classification/sorting problem is one of the most studied problems in variety of disciplines. Even though different disciplines describe this problem using different parlance, in general the main idea is to assign a pre-defined class label to the alternatives under consideration. There are different terms used for this problem type, however, the commonly accepted terms are as follows (Doumpos & Zopounidis, 2002) :

- Discrimination
- Classification
- Sorting

The first two terms are used by people studying in SL and AI fields, while sorting has been established by the MCDA practitioners (Moreover, sorting problems are defined as ordinal classification problems in the SL parlance) (Doumpos & Zopounidis, 2002). Even though all these three problems deal with assigning a class label to the alternatives under consideration, there is an ordering of classes in the sorting problem while in discrimination and classification problem, classes are defined nominally. This is the main difference between classification and sorting problems.

In the classification/sorting problem, each alternative $\boldsymbol{x}$ is represented with a set of criteria $\boldsymbol{c} = (c_1,\ldots,c_n)$. Therefore, alternatives turn out to be vectors

$\boldsymbol{x}_j = \left(c_{j1},...,c_{jn}\right)$, where c_{ji} corresponds to the score of alternative $\boldsymbol{x}_j$ achieved in criterion c_i. Consequently, the classification task can be defined as developing a model that maps each alternative into a class label $y \in \left(y_1,...,y_t\right)$, where index *t* corresponds to the number of pre-defined classes. In the sorting case, however, classes are ordered, hence, $y \in \left(y_1 \succ ... \succ y_t\right)$.

Even though the above definition of classification/sorting problem is common for various classification/sorting techniques, the model used to map alternatives into predefined groups may have different forms in MCDA. Most commonly used ones include (Doumpos & Zopounidis, 2011):

- **Value functions:** A value function ($V(\boldsymbol{x})$) representing degree of preference is defined such that, for two alternatives ***x*** and ***y***,

$$\begin{aligned} V(\boldsymbol{x}) > V(\boldsymbol{y}) &\Leftrightarrow \boldsymbol{x} \succ \boldsymbol{y} \\ V(\boldsymbol{x}) = V(\boldsymbol{y}) &\Leftrightarrow \boldsymbol{x} \sim \boldsymbol{y} \end{aligned} \tag{2}$$

  where $\succ$ and ~ correspond to preference and indifference relations.
- **Outranking relations:** An outranking relation *S* between two alternatives ***x*** and ***y*** is defined such that;

$$\boldsymbol{x}\, S\, \boldsymbol{y} \Leftrightarrow \boldsymbol{x} \text{ is at least as good as } \boldsymbol{y} \tag{3}$$

- **"If...then..." decision rules:** In this form, model consists of two parts; condition part that follows "if" statement and conclusion part that follows "then" statement.

Name of the function that is used to represent DM's preference structure differs based on the decision condition. Making this distinction first time in the literature,

Keeney and Raiffa (1993) distinguish preference functions based on the risk associated with the alternatives. In their parlance, preference representation functions under risk are referred to as utility functions while preference representation functions under certainty are referred to as value functions. In the case of risky choice, a probability distribution is associated with consequences, hence, there exists uncertainty with respect to each consequence. Conversely, consequences of alternatives are certain in the latter case. Assuming that all consequences of the alternatives are certain (decisions are made under certainty) we will refer to functions representing preference structure of the DM as value function thereafter.

Obtaining preference information from the DM is an important aspect of the preference modeling process. There are two main ways of getting preferential information from the DM: In direct way (or forward approach) DM specifies values for the parameters used in the preference model (i.e. weights, threshold values, etc.). In indirect way (or backward approach) DM is asked to make holistic judgements about reference alternatives. Based on these reference judgements, parameters of the preference model are elicited. Jacquet-Lagreze and Siskos (2001) explain the relationship between these two approaches with aggregation-disaggregation paradigms. In the aggregation paradigm criteria aggregation model is known a priori, hence, parameters of the model are estimated in the model development process. Conversely, in the disaggregation paradigm, the model is estimated from the holistic judgements made by the DM. In other words, disaggregation approach uses regression-like techniques to model DM preference structure by using a reference set of alternatives. Relationship between these two paradigms is illustrated in Figure 2 (Jacquet-Lagreze & Siskos, 2001).

Model development process utilizing disaggregation paradigm is referred to as Disaggregation Analysis (DA). The most important input of DA is the reference

set which contains representative alternatives associated with global preferences (or label information). Reference set may be comprised of (Siskos et al. ,2005):

- Fictitious but realistic representative examples,
- Past decisions of the DM,
- Subset of past decisions when the set is large.

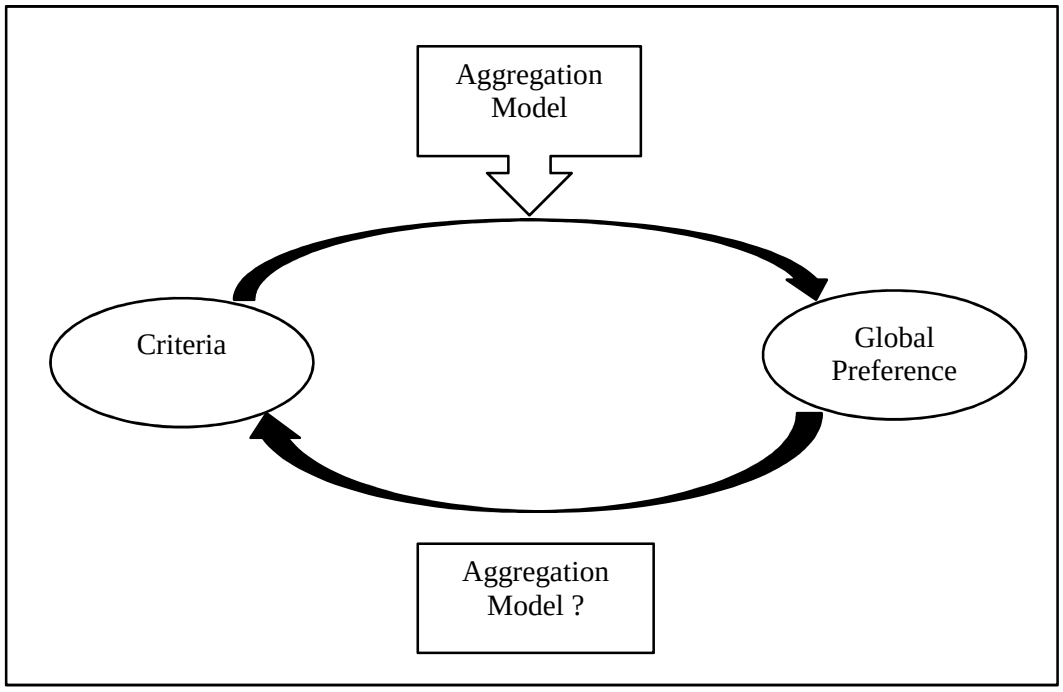


Figure 2. Aggregation and disaggregation paradigms (Source: Siskos et al. (2005)).

In DA, the main objective is to estimate the best set of parameters ϕ of the decision model that is believed to represent the preference structure of the DM as shown in Equation (4) (Doumpos & Zopounidis, 2011):

$$\hat{\phi}^* = \arg\min_{\hat{\phi}\in A} \left\| \hat{\phi} - \phi \right\|_p \tag{4}$$

where $\left\| \hat{\phi} - \phi \right\|_p$ corresponds to *p*-norm of the differences between the actual and the estimated parameters and *A* corresponds to the feasible set of values for the parameters. In order to solve problem (4), the reference set is used empirically. Let *D*(*X*) denote evaluations of the DM on a set *X*. Then, the optimization problem turns out to be:

$$\hat{\phi}^* = \arg\min_{\hat{\phi}\in A} L\left[D(X), \hat{D}(X, f_{\hat{\phi}}) \right] \tag{5}$$

where $f_{\hat{\phi}}$ is the preference model developed and *L*(.) is a function that measures the difference between preferences made by the DM ($D(X)$) and estimations made by the model ($\hat{D}(X, f_{\hat{\phi}})$).

Similar to the DA of MCDA, SL also learns from examples. For the classification/sorting setting where output is qualitative, an estimate $\hat{G}$ is used to predict outputs where each output takes values from set ζ. A loss function represented by a $K \times K$ matrix $\boldsymbol{L}$ is used to measure difference between estimates and observations, where K = card (ζ) (Hastie et al., 2009). According to Hastie et al. (2009), the Expected Prediction Error (EPE) is represented as:

$$EPE = E\left[L\left(G, \hat{G}(X)\right) \right] \tag{6}$$

where $X \in R^p$ denote a real valued random input vector. Taking expectation with respect to the joint distribution *P*(*G*, *X*) and conditioning,

$$EPE = E_x \sum_{k=1}^{K} L\left[\zeta_k, \hat{G}(X)\right] P(\zeta_k \mid X) \tag{7}$$

minimizing EPE pointwise,

$$\hat{G}(x) = \arg\min_{g \in \zeta} \sum_{k=1}^{K} L[\zeta_k, g] P(\zeta_k \mid X = x) \tag{8}$$

with the 0-1 loss function,

$$\hat{G}(x) = \arg\min_{g \in \zeta} \left[1 - P(g \mid X = x)\right] \tag{9}$$

As seen from the formal definitions of multi-criteria/multi-attribute sorting/classification problem in MCDA and SL domains, the main idea is to develop a model/classifier that learns from examples and maps each alternative into consequence space of outcomes. In the following sections we will provide techniques from these two fields that are developed to deal with classification/sorting problem. Additionally, we will mention Conjoint Analysis, a decomposition method from the marketing field, aiming to estimate preference structure of customer(s) (and parameters of the mathematical representation of the model) by means of questionnaires. Similar to the focus of this thesis and different than the techniques proposed in MCDA, Conjoint Analysis considers preference modeling as a learning process, thereby seeks for methodologies that implement preference modeling process in an evolutionary way.

2.1 Multicriteria Sorting Techniques in MCDA

One of the earlier examples of DA based sorting methods in MCDA is UTADIS (UTilities Additives DIScriminantes). Basic principles introduced by Devaud et al. (1980), UTADIS is a variant of UTA (UTilities Additives) method (Jaquet-

Lagreze and Siskos, 1982) that is developed for ranking problems. UTA method aims at inferring an additive utility function by using ordinal regression approach that ranks reference alternatives the same way as it was done by the DM. Utilizing an additive utility function, UTA assumes preferential independence among criteria. Likewise, UTADIS also utilizes an additive utility function as the criteria aggregation model and assumes preferential independence. Inferred additive utility function is used to sort reference alternatives where alternatives that achieve highest scores are labeled with C_1, and alternatives that achieve worst scores are labeled with C_k , where k represents number of groups and $C_1 \succ ... \succ C_k$. UTADIS procedure may generate multiple optimal solutions. This situation is usually encountered when references are perfectly separable. In this case, alternative utility functions that sort the reference examples the same way as it was done by the DM can be obtained. These kind of utility functions are called compatible utility functions (Figueira et al., 2009). In order to improve generalization ability of the model, post-optimality analysis is performed in an effort to explore existence of alternative optimal and near optimal solutions.

As a member of the family of ELECTRE methods, ELECTRE TRI method has been proposed by Yu (1992) for sorting problems. Like other ELECTRE methods, ELECTRE TRI utilizes outranking relations approach. The procedure is implemented in two stages. In the first stage, an outranking relation is developed in order to determine if an alternative outranks a profile, which is a fictitious alternative defined for separating classes. In the second stage, developed outranking relations are used to assign alternatives to classes. Being an outranking relation approach, ELECTRE TRI can model incomparability relations. The main drawback of ELECTRE TRI procedure is that it needs preferential parameters like criteria weights, veto thresholds or profiles. It is unrealistic to expect DM to provide these parameters directly and the process to elicit these parameters is usually troublesome.

Ulu and Köksalan (2001) have developed an interactive sorting procedure for a two-class case where classes corresponded to acceptable and unacceptable sets. They considered three different underlying utility functions, namely, linear, quasiconcave, and general monotone. Their procedure is effective when there are not many criteria and number of alternatives is large. In the algorithm, an alternative is assigned to one of the two classes based on previous preferences of the DM, dominance relationships among alternatives, and properties of assumed underlying utility function. In order to assign alternatives to classes, feasibility Linear Programs (LPs) are solved. If an alternative cannot be assigned to a class, then preference information for the alternative is claimed from the DM.

Utilizing the principles of rough set theory of Pawlak (1982), MCDA techniques based on rough sets theory have become popular among researchers. In their prominent paper, Greco et al. (2001) have outlined the principles of Dominance-based Rough Set Approach (DRSA) where a distinction between attribute and criterion was made. Different than an attribute, a criterion is defined on preference-ordered domain. In this respect, indiscernibility relation based on attributes is substituted by dominance relation. Coherent with these principles, Greco et al. (2002) have developed a rule based procedure for dealing with sorting problems. Their procedure makes use of disaggregation paradigm, thereby, they use reference examples for eliciting preference information from the DM. Making the distinction between attribute and criterion, they build “rough” approximations of decision classes defined by “indiscernibility” relations based on qualitative attributes, “similarity” relations based on quantitative attributes, and “dominance” relations based on criteria. They present the model with decision rules in the form of logical statements as “*if...then...*”. They state that models of this form have the advantage of interpretability.

Köksalan and Ulu (2003) have developed an interactive multicriteria sorting procedure in which DM was assumed to have a linear (additive) utility function.

The procedure is evolutionary in nature and based on the preference information obtained from the DM, dominance relations of the alternatives and additivity of the utility function, unlabeled alternatives are placed in pre-defined classes. At each iteration, either class information of an alternative is asked to the DM or an alternative is placed in a class by the algorithm based on the answers gathered thus far.

Utilizing disaggregation paradigm, Köksalan et al. (2009) have proposed a multicriteria sorting procedure based on outranking relations. Their work can be considered as an extension to the ELECTRl TRl method. ELECTRE TRl method requires DM to provide parameters like weights, thresholds and profiles that represent categories. Considering unrealistic nature of the method due to the difficulty of obtaining this information from the DM, they have proposed a new method that elicits profiles representing limits of the categories from the reference set that has been constructed in supervision of the DM.

Köksalan and Bilgin Özpeynirci (2009) have considered a method that assumed an additive underlying utility function representing preference structure of the DM. The proposed method, which is an extension to the UTADIS procedure in general, overcomes the misclassification drawback of UTADIS by interactively querying the DM. Rather than trying to estimate parameters of the additive utility function, they place alternatives into pre-defined classes based on additive utility function assumption and preference information obtained from the DM. ln their paper they have showed that proposed approach has outperformed UTADIS procedure.

As an another extension to the UTADIS method, Greco et al. (2010) have proposed UTADIS^{GMS}. Like its predecessor, UTADIS^{GMS} is an ordinal regression method, nevertheless, rather than considering only one or a subset of compatible utility functions produced by the model, the proposed procedure considers all compatible utility functions. By doing that, two kinds of assignments are made for

the alternatives: necessary and possible assignments. Necessary assignments are those that are proposed by all compatible value functions while assignments proposed by any compatible value function are called possible assignments.

Considering inadequacy of additive utility functions in handling interacting criteria, Angilella et al. (2009) have proposed using Choquet integrals, which have been popular as an aggregation operator in order to deal with interaction phenomenon, in the robust ordinal regression framework for sorting problems. Extending the idea of UTADIS, they utilize Choquet integral as an aggregator instead of an additive value function. Similar to the UTADIS^{GMS} procedure, robust ordinal regression framework is used to handle multiple compatible value functions, hence, necessary and possible assignments are elicited.

Considering the difficulty of eliciting too many parameters in the ELECTRE TRI method, Leroy et al. (2011) have proposed using a simplified version of ELECTRE TRI, where an alternative was assigned to a class or above if this alternative was as good as the "lower profile" of that class for a majority of criteria. They refer to their method as MR-Sort. In order to find profile and weight values, they utilize a learning procedure where a set of labeled reference assignments are used.

Soylu (2011) has proposed a multicriteria sorting method where preference structure of the DM was represented with a Tchebycheff utility function. The main reason for using Tchebycheff utility function is that a weighted Tchebycheff utility function can reach non-convex regions of the efficient frontier and may represent variety of preference structures by adjusting weights. The proposed methodology uses disaggregation paradigm, hence, a reference set of alternatives that are classified into predefined classes by the DM are used to label remaining alternatives by performing pairwise comparisons. Comparison of two alternatives is performed by calculating strength of alternative *i* over *j*, which is measured with

difference between weighted Tchebycheff distances of alternatives to the ideal point. The method utilizes the same idea as in UTADIS, therefore, final classification is made by comparing average strength of each alternative with computed threshold values of each class.

As an extension to UTADIS^{GMS} method, Greco et al. (2011) have proposed a methodology for selection of a representative value function which represented necessary and possible relations produced by UTADlS^{GMS} in an effort to provide an interpretable decision model to the DM. This is usually achieved by interacting with the DM, so that DM expresses targets that are to be attained by the representative value function. The produced value function has two possible uses: firstly, DM is presented an interpretable decision model which helps him/her understand his/her preference structure. Secondly, the produced model can be used to sort unseen (those instances that are not used in the training phase) future alternatives.

Buğdacı et al. (2013) have developed a probabilistic sorting approach where the probability of an alternative belonging to each class was calculated. Afterwards a proper assignment of alternatives into pre-defined classes is performed based on the calculated probability of misclassification. Given the assignments made thus far, membership probabilities of each alternative are recalculated and the procedure is carried out in an evolutionary manner. They assume an additive underlying utility function and also assume that utility related parameters and thresholds can be estimated by interacting with the DM. Whenever an alternative cannot be assigned to a class, correct class information is requested from the DM. lt is claimed that the proposed approach addresses two problematic issues usually encountered in multiple criteria sorting methodologies. Firstly, high misclassification rates of those procedures that aim to estimate parameters of the preference model. Secondly, excessive involvement of the DM in the interactive procedures.

In another study, Ulu and Köksalan (2014) have assumed a quasiconcave value function that represented DM's preferential structure for sorting alternatives to preference ordered classes. Emphasizing that their work is the first example utilizing quasiconcave underlying value functions in sorting problems, they use preference information obtained from the DM, dominance relations, and properties of quasiconcave value function for assigning unlabeled alternatives to classes. Whenever needed, DM is requested to place an alternative to a proper class and then as many alternatives as possible are assigned based on the dominance relations and mathematical properties of quasiconcave value function. Hence, they carry out the procedure in an interactive way.

In order to deal with alternative optimal solutions that lead to multiple compatible value functions in the ordinal regression approach, Çelik et al. (2015) have studied a new probabilistic distance based sorting procedure that utilized an approach similar to necessary-possible class method. Two kinds of threshold levels are computed that are used to separate each consecutive class pair: maximum and minimum. Based on these thresholds, pessimistic (widest) and optimistic (narrowest) ranges are determined which are used to calculate class-belonging probabilities.

Extending the idea of UTADIS, Corrente et al. (2015) have applied Multiple Criteria Hierarchy Process (MHCP) framework proposed by Corrente et al. (2012) to sorting problems. MHCP is developed to deal with multiple criteria in a hierarchical scheme, hence, large sets of criteria are partitioned into levels. By partitioning the criteria, MCDA process is simplified for the DM and the problem becomes more manageable. Similar to UTADIS, their technique assumes that DM's preference structure is compatible with additive functional form.

All these aforementioned studies are presented in Table 2 based on modeling approach and assumed underlying value function for comparison.

Table 2. Multicriteria sorting techniques proposed in the literature.

Authors	**Modeling Approach**	**Assumed Value Function**
Devaud et al. (1980)	Functional	Additive
Yu (1992)	Outranking Relations	-
Ulu and Köksalan (2001)	Functional	Additive Quasiconcave Gen.Monotone
Greco et al. (2002)	Rule Based	-
Köksalan and Ulu (2003)	Functional	Additive
Köksalan et al. (2009)	Outranking Relations	-
Köksalan and Özpeynirci (2009)	Functional	Additive
Angilella et al. (2009)	Functional	Choquet integral
Greco et al. (2010)	Functional	Additive
Soylu (2011)	Functional	Tchebycheff
Greco et al. (2011)	Functional	Additive
Leroy et al. (2011)	Outranking Relations	-
Buğdacı et al. (2013)	Functional	Additive
Ulu and Köksalan (2014)	Functional	Quasiconcave
Çelik et al. (2015)	Functional	L_p norm
Corrente et al. (2015)	Functional	Additive

As seen in Table 2, most of the techniques outlined in this section utilize a value function approach, hence, assume an underlying value function that is believed to represent preference structure of the DM. The main reason for assuming a functional form is that it makes modeling process easier, thus, by using mathematical properties of the assumed value function, a representative model is elicited. There are three outranking relations-based sorting techniques in the list while the only rule-based technique is proposed by Greco et al. (2002). Considering the years of the studies, we can infer that multicriteria sorting problem is drawing a growing interest recently. This is believed to stem from the fact that, multicriteria sorting problems are confronted more often in various fields like finance, risk evaluation, quality management, human resources management etc.

As stated previously, even though being an important phenomenon in real life MCDA problems, an interaction phenomenon is usually neglected or all the criteria under consideration are assumed to be preferentially independent. We see reflections of the aforesaid proposition in Table 2. Almost all of the proposed functional techniques assume functional forms that are not able to model interaction structures. Although quasiconcave functions do not assume preferential independence, their monotone structure prevent them model complex interaction structures. It is not clear if outranking relations approach can model interacting criteria and this is subject to experimentation, however, it is generally assumed that these techniques require preferential independence assumption among criteria (Corrente, 2012). A comparison of the proposed techniques in this context is presented in Table 3.

As clearly seen from Table 3, there are only three studies (Greco et al., 2002; Angilella et al., 2009; Ulu & Köksalan, 2014) that do not assume preferential independence. All techniques mentioned in this section assume that DM has a monotonic preference structure. Therefore, those three studies that do not assume preferential independence cannot model complex interactions that show nonmonotonic structures. As a consequence, we can assert that their ability to model interactions is limited. Interactive MCDA techniques are popular among sorting problems due to their ability to integrate DM to the model developing process and expedite learning process. Nevertheless, Table 3 reveals that interactive procedures usually ignore interaction phenomenon or assume it does not exist. Consequently, based on our literature review on MCDA sorting techniques, we can claim that there is a need for new procedures that elicit preference information from the DM interactively and is able to deal with complex interactions and nonmonotonic structures efficiently without making any functional assumptions, as indicated with gray zone in Table 3.

Table 3. Comparison of multicriteria sorting techniques.

Interactive	**Functional Assumption**	**Ability to Model Interactions**	
		No	**Yes (Limited)**
No	Yes	Devaud et al. (1980) Greco et al. (2010) Soylu et al. (2011) Greco et al. (2011) Çelik et al. (2015) Corrente et al. (2015)	
	No	Yu (1992) Leroy et al. (2011)	Greco et al. (2002) Angilella et al. (2009)
Yes	Yes	Ulu and Köksalan (2001) Köksalan and Ulu (2003) Köksalan and Özpeynirci (2009) Buğdacı et al. (2013)	Ulu and Köksalan (2014)
	No	Köksalan et al. (2009)	

In general, the main objective of the aforementioned techniques is to sort limited number of alternatives of the problem under consideration with maximum accuracy. They take a subset of the alternatives for getting preference information. Hence, the preference model that is developed based on the preference information obtained with respect to reference alternatives is used to sort the rest. Even though developed models represent preference structure of the DM, these techniques cannot be considered as preference modeling approaches. Additionally, they do not concern about predicting equally good everywhere in the criteria domain, which corresponds to generalization ability.

2.2 Nonparametric Classification and Ordinal Regression Techniques in Statistical Learning

Drawing a growing interest in a variety of fields recently, SL refers to tools for understanding and modeling complex structures embedded in data (James et al., 2013). Emerging as a subfield of statistics, it combines various disciplines such as computer science and operations research. Broadly speaking, SL techniques can be divided into two main parts: supervised and unsupervised learning. In supervised learning, the main aim is to infer a statistical model for prediction or estimation under the supervision of "outputs". Hence, a predictive model that explains the relationship between "inputs" and "outputs" is constructed. In unsupervised learning, on the other hand, only inputs are utilized to learn structures of the data. Classification problem falls into supervised learning part where outputs are qualitative or categorical. In this respect, classification models aim to build a predictive relationship between inputs, qualitative or quantitative, and qualitative output. After training a classification model, the model classifies or labels an (usually previously unseen) observation based on its input values.

Classification is one of the most studied supervised learning techniques in the literature, therefore, numerous procedures have been developed. In this section, we will provide a brief review of prominent nonparametric classification techniques proposed in the literature. Additionally, we will outline basics of Support Vector Machines (SVM) and Random Forest (RF), since we utilize these techniques in our thesis. As previously stated, we utilize nonparametric classification techniques in this study. In general, parametric models make assumptions regarding underlying statistical properties of the data of interest. Based on these statistical assumptions, parameters of the model are elicited. However, statistical properties of the data are hardly known, therefore, incorrect statistical assumptions lead to wrong inference. At this point, it would be proper to make it clear that nonparametric models use parameters to build models, as well. However, these

parameters are not parameters of predefined statistical model, rather, parameters that are determined by the observations of the data. Having this property, nonparametric models outperform their parametric counterparts in terms of learning from complex data structures.

Inspired by the human brain, artificial Neural Networks (NNs) have been introduced in the artificial intelligence field to deal with complex problems. Basically, a NN is a regression or classification model represented with a directed acyclic graph of neurons organized into layers. Generally speaking, a NN consist of a layer of input nodes (inputs), a layer of output (classes) nodes and intermediate (hidden) layer. Depending on the topology of the network, functions or models of different complexity can be represented with neural networks. Because of its somewhat closed-form, it is usually hard to provide an explanation or interpretation for outputs. Fitting NNs is quite an art. There are many issues to be considered to avoid overfitting. For detailed explanations of these issues, see Hastie et al. (2009).

First proposed by Breiman et al. (1984), Classification and Regression Trees (CART) is a nonparametric decision tree based statistical learning technique that can be applied to regression and classification problems based on type of the dependent variable, quantitative or qualitative. Based on the same principles, Quinlan (1993) introduced C4.5 algorithm. In decision tree learning, every node corresponds to an attribute while every branch represents a condition based on the node attribute. Each leaf corresponds to a class, hence, leaves label an observation satisfying the conditions of the branches on the path. Decision tree based classification algorithms become popular recently because of their following properties (Doumpos & Zopounidis, 2002):

- Handling both quantitative and qualitative attributes
- Dealing with missing values

- Interpretability of the models produced

Stone et al. (1997) developed an extension of Multivariate Adaptive Regression Splines (MARS) algorithm, namely PolyMARS, for classification problems. MARS is an adaptive procedure for regression and well-suited for modeling non-linearities and interactions among attributes (Friedman, 1991). MARS uses basis functions taking one of the three forms to build models: a constant, hinge functions and product of two or more hinge functions. Use of hinge functions provides high flexibility. Weber et al. (2012) proposed an extension to MARS where they apply it with Tikhonov regularization and conic quadratic programming. Our previous experience with MARS and PolyMARS has showed that, these techniques are highly sensitive to size of the training set. When the training size is small, they have a tendency to overfit data, which harms the generalization capability of the model.

2.2.1 Support Vector Machines

Support Vector Machines (SVMs) have become very popular recently due to their performance in variety of different applications such as text classification, face recognition, database marketing and bioinformatics (Campbell & Ying, 2011). Initially developed for binary classification, SVMs can be applied to multi-class classification, regression and clustering problems. The main idea of SVM is to generate a hyperplane or set of hyperplanes usually in high dimensional space that will separate data into classes as efficient as possible. If groups are perfectly separable by a linear hyperplane, then this classifier is called maximum margin classifier.

The main objective of a maximum margin classifier is to find a separating hyperplane that is farthest from the observations. Among all such perpendicular distances to the separating hyperplane, the one that is the smallest is called margin,

hence, maximum margin classifier seeks a solution where margin is maximized. Separating hyperplane can be formulated as $\boldsymbol{w}.\boldsymbol{x}+b=0$, where $\boldsymbol{x}$ are the points that are positioned on the hyperplane and $\boldsymbol{w}$ is the weight vector normal to the hyperplane. For the binary classification case, observations that satisfy $\boldsymbol{w}.\boldsymbol{x}+b\geq 0$ are classified as class 1 (y_i = *1*) , and observations that satisfy $\boldsymbol{w}.\boldsymbol{x}+b<0$ are classified as class -1 (y_i = *-1*). In this respect, closest points to the separating hyperplane hold $\boldsymbol{w}.\boldsymbol{x}+b=\pm 1$. These hyperplanes are called canonical hyperplanes. All these concepts are illustrated in Figure 3 (Schölkopf & Smola, 2002).

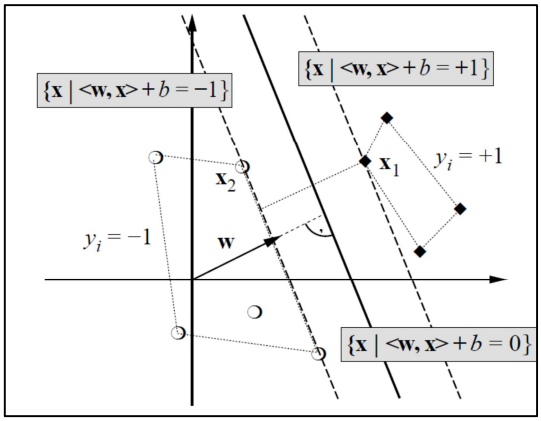


Figure 3. Margin and hyperplane concepts in SVM (Source: Schölkopf & Smola (2002)).

Assume we have two observations ($\boldsymbol{x_1}$ and $\boldsymbol{x_2}$) on the canonical hyperplanes lying on the opposite sides, then we can formulate $\boldsymbol{w}.(\boldsymbol{x}_1-\boldsymbol{x}_2)=2$. Normal vector to the separating hyperplane is $\boldsymbol{w}/\|\boldsymbol{w}\|_2$. Then, distance between two canonical hyperplanes can be written as $(\boldsymbol{x}_1-\boldsymbol{x}_2).\boldsymbol{w}/\|\boldsymbol{w}\|_2$ yielding to a margin $1/\|\boldsymbol{w}\|_2$. Hence, finding the maximum margin classifier problem can be formulated as (Campbell & Ying, 2011):

$$\begin{aligned} &\textit{Minimize} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 \\ &st. \\ &\quad y_i(\boldsymbol{w}.\boldsymbol{x}_i + b) \geq 1 \quad \forall \text{i} \end{aligned} \tag{10}$$

Generally classes are not perfectly separable, therefore there exists no separating hyperplane and maximum margin classifier. In this case, a generalization of the maximal margin classifier, namely support vector classifier, is used. Different than the maximum margin classifier, support vector classifiers allow some of the observations be on the wrong side of the margin, even wrong side of the hyperplane (James et al., 2013). Having this property, support vector classifiers are sometimes called as soft classifiers. Those vectors that lie in the margin or wrong side of the separating hyperplane are called support vectors and determine support vector classifier. In order to allow wrong located observations, an error norm and slack variable are utilized. Consequently, determining the support vector classifier turns out to be finding optimal solution to the following optimization problem:

$$\begin{aligned} &\textit{Minimize} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{m}\varepsilon_i \\ &st. \\ &\quad y_i(\boldsymbol{w}.\boldsymbol{x}_i + b) \geq 1-\varepsilon_i \quad \forall \text{i} \\ &\quad \varepsilon_i \geq 0 \qquad \forall \text{i} \end{aligned} \tag{11}$$

where ε is the slack variable and *C* is the tuning parameter that defines the trade-off between margin maximization and error minimization (Doumpos & Zopounidis, 2011). By using Lagrange multipliers, the primal problem in (10) can be rewritten as;

$$L_p = \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{m}\varepsilon_i - \sum_{i=1}^{m}\alpha_i\left[y_i(\boldsymbol{w}.\boldsymbol{x}_i + b) - 1 + \varepsilon_i\right] - \sum_{i=1}^{m}\mu_i\varepsilon_i \qquad (12)$$
$$\alpha_i \geq 0,\ \mu_i \geq 0 \quad \forall i$$

which is minimized with respect to $\boldsymbol{w}$, b and ε_i. Taking derivative of L_p with respect to these variables we get;

$$\boldsymbol{w} = \sum_{i=1}^{m}\alpha_i y_i \boldsymbol{x}_i \qquad (13)$$

$$0 = \sum_{i=1}^{m}\alpha_i y_i \qquad (14)$$

$$\alpha_i = C - \mu_i \qquad (15)$$

Substituting (13) - (15) into (12), we get Lagrangian (Wolfe) dual;

$$L_D = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{i=1}^{m}\alpha_i\alpha_j y_i y_j (x_i.x_j) \qquad (16)$$

which is maximized subject to $0 \leq \alpha_i \leq C$ and $\sum_{i=1}^{m}\alpha_i y_i = 0$.

Like maximum margin classifiers, support vector classifiers build linear boundaries. However, data may not be separable with linear boundaries as illustrated in Campbell & Ying (2011).

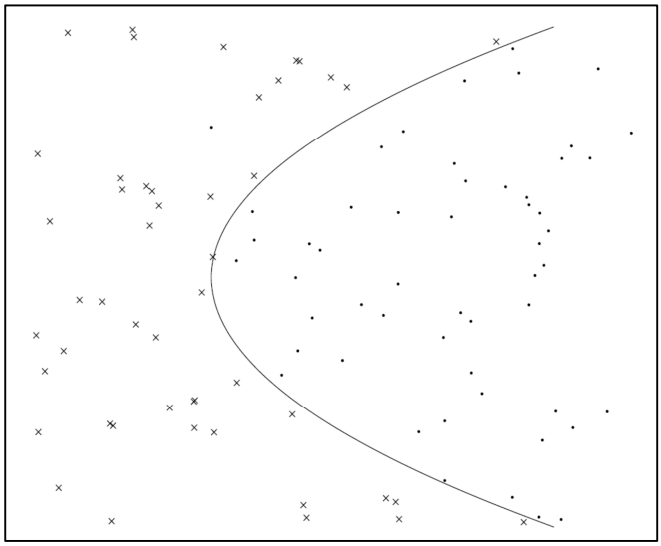


Figure 4. Linearly inseparable data (Source: Campbell & Ying (2011)).

In order to generalize the linearly separable case, a kernel trick is used to generate nonlinear boundaries by mapping data points to a higher dimension, called feature space. This is achieved by applying a transformation $x_i.x_j \rightarrow \Phi(x_i).\Phi(x_j)$ through the mapping function $\Phi(.)$. Mapping function is defined with a kernel, $K(x_i, x_j) = \Phi(x_i).\Phi(x_j)$. The main idea is that, with the transformed data in the feature space, linear boundaries can be developed that will efficiently separate classes. Most commonly used kernels are (Hastie et al., 2009);

- d^{th}-Degree polynomial: $K(x_i, x_j) = (1 + \langle x_i, x_j \rangle)^d$ (17)
- Radial basis: $K(x_i, x_j) = e^{(-\gamma \|x_i - x_j\|^2)}$ (18)
- Neural Network: $K(x_i, x_j) = \tanh(k_1 \langle x_i, x_j \rangle + k_2)$ (19)

As stated previously, SVMs are first developed for the binary classification case. Among many techniques to extend SVMs to multi-class case, two techniques prevail, namely, *one-versus-one* and *one-versus-all*. *One-versus-one* approach partitions the problem into $\binom{K}{2}$ SVM problems (K is the number of classes), where each problem considers two pair of classes. On the other hand, *one-versus-all* approach trains SVMs to separate K classes from the remaining K-1 classes. Then class assignments are done accordingly. For detailed information regarding extending SVMs to multi-class case, refer to Hastie et al. (2009).

2.2.2 Random Forests

Even though the idea of RFs is based on collective work of several researchers, it was first Breiman (2001) to introduce principles of RF. RFs are decision tree based ensemble methods for classification and regression. Thus, RFs utilize multiple decision trees that train on the data and prediction is made based on the predictions of individual trees.

Tree-based methods utilize a simple idea where the attribute domain is partitioned into rectangles and then a separate model is fit to each of them. When the dependent variable is nominal, a class label is assigned to each of these rectangles. Trees dealing with nominal dependent variables are called classification trees. In training the tree, recursive binary partitions are implemented, hence, only one independent variable is considered for partition at each step. For instance, assume that we consider variable X_1 for partition and we decide to split at $X_1 = t_1$. Then the attribute domain is partitioned into two sub-regions as $X_1 \leq t_1$ and $X_1 > t_1$ (Hastie et al., 2009). The idea of recursive partitioning is illustrated in Figure 5. At the training phase, observations fall into proper branches in the tree based on how they satisfy the conditions of splits. When they reach the terminal or leaf nodes, they are assigned the labels of these nodes as predictions. In tree-based methods

the training phase is implemented so that percentage of true predicted instances is maximized. On the other hand, a regularization is also utilized in order to improve generalization ability of the model.

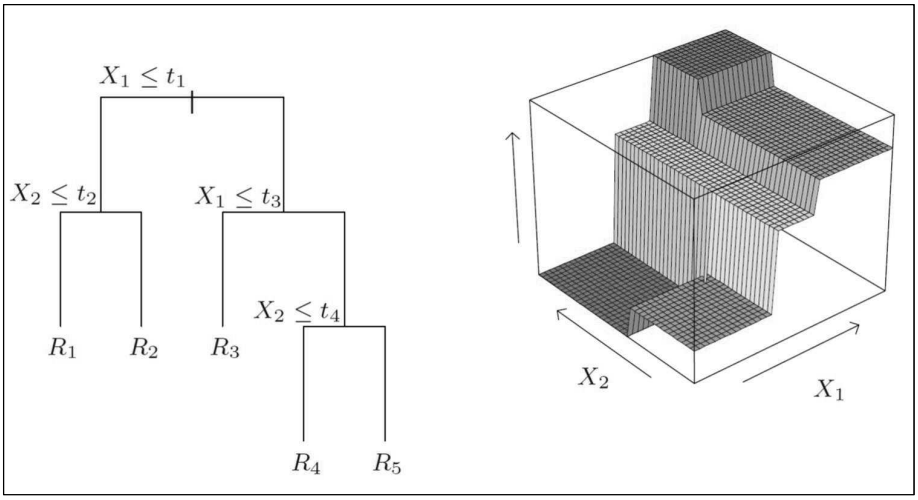


Figure 5. Recursive partitioning in tree-based methods (Source: Hastie et al. (2009)).

Decision trees have been very popular among machine learning community, however, when grown very deep, they tend to overfit the training data. Hence, they are low-bias and high variance procedures (Hastie et al., 2009). RFs use bagging (short for bootstrap aggregating) idea of Breiman (1996) in order to reduce the variance at the expense of a small increase in bias. Bagging aims at reducing the variance and increasing the prediction accuracy by taking many samples of same size from the original training set with replacement (bootstrapping), fitting separate decision trees (T_b) to each training set and averaging the predictions. In addition to bagging, RFs use a smart technique in order to decorrelate generated trees. When growing bagged decision trees, only a random subset of *m* predictor

variables out of all predictors (*p*) are considered for splitting each time we split. Usually size of the randomly selected random predictors is $\sqrt{p}$ (Hastie et al., 2009). The reason for this modification to bagged trees is to prevent strong predictors to dominate splitting phase, consequently resulting similar trees. This is the major difference between bagged trees and RF. In order to make prediction for an instance *x*, each tree makes its own prediction for the instance ($\hat{C}_b(x)$) and then majority vote ($\hat{C}_{RF}^{B}(x)$) is output as the prediction of the ensemble. Pseudo-code of the RF algorithm is presented in Figure 6 (Hastie et al., 2009).

1. **for** b = 1 to B **do**
2. draw a bootstrap sample Z of size N from the training data.
3. grow a RF tree T_b for the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size *nmin* is reached.
4. select m variables at random from the p variables
5. pick the best variable/split-point among the m.
6. split the node into two daughter nodes.
7. **end for**
8. output the ensemble of trees $\{T_b\}_1^B$
9. to make a prediction at a new point x:
10. let $\hat{C}_b(x)$ be the class prediction of the b^{th} RF tree. Then;

$$\hat{C}_{RF}^{B}(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$$

Figure 6. Pseudo-code of the RF algorithm.

Similar to decision trees, RFs are very capable of handling interactions among predictors. Additionally, RFs are very robust to overfitting (Hastie et al., 2009). This is due to its ensemble structure, i.e., averaging predictions of many grown bagged trees. This property becomes very important when size of the training data is small because classifiers usually suffer from overfitting in the presence of small data sets. Another valuable property of RFs is that, similar to other decision tree based learning methods, RFs handle qualitative predictors very easily, without creating dummy variables. On the other hand, RFs do not provide models that are as interpretable as those built with decision trees since they utilize multiple trees in the learning phase and predictions are made based on consensus.

All nonparametric classification techniques mentioned in this section offer strong classifiers. Nonetheless, it is not easy to determine which classification algorithm is the best. There are several studies regarding this issue, comparing performance of classifiers under different classifying tasks (Lim et al., 2000; Duan & Keerthi, 2005; Kotsiantis, 2007; Madjarov et al., 2012). However, as Kotsiantis (2007) and James et al. (2013) state, there is no best algorithm that outperforms others in all circumstances. For instance, if there exists a complex interaction structure among predictors, then a decision tree based approach like RF may outperform others. Conversely, if there is no interaction among predictors and classes can be separated with linear boundaries, another classifier (for instance SVM) may perform better than RF.

2.2.3 Ordinal Classification Problem

As stated previously, the main difference between classification and sorting (or ordinal classification) problems is that, in the first case classes are defined nominally while in the latter case there is an order among classes. All aforementioned SL techniques are developed to address nominal classification problems. Even though there are many applications that employ nominal

classification techniques for ordinal classification problems, special structure of the ordinal classification problems contain valuable information that can expedite the learning process.

In their paper, Hühn and Hüllermeier (2009) pose the question "**Is the ordinal class structure useful in classifier learning?**" and aim to determine if it is possible to exploit the order information of the problem. In order to investigate this empirically, they apply ordinal classification techniques to same ordinal problem sets in two different settings. In the first setting, learner is applied to the true ordinal problem. In the second setting learner is applied to the distorted problem where labels are given to groups in an arbitrary permutation. At the end of the experimentation they come up with two results:

- Ordinal classification techniques do exploit the ordinal structure of the data
- Complicated learners producing highly flexible decision boundaries benefit less than the learners producing less flexible boundaries.

As a consequence, prediction performance increases if learners that are able to exploit ordinal structure of the data are used for ordinal classification problems.

SL practitioners studied on ordinal classification problems in an aim to develop nonparametric approaches. For instance, Chu and Keerthi (2007) applied support vector approach to ordinal regression problems by optimizing multiple thresholds to define parallel discriminant hyperplanes for the ordinal scales. Pinto da Costa et al. (2008) have proposed a new approach where posterior probabilities of the predictions are enforced to follow a unimodal structure. They state that since there is an ordinal relationship among classes, the posterior probabilities should decrease monotonically on both sides of the class having the highest posterior probability. In order to achieve this, they impose unimodality in two ways. In the parametric approach, posterior probabilities are enforced to follow a given parametric distribution. In the nonparametric approach, unimodality is imposed by penalizing non-unimodal distributions using error measures. Waegeman and Boullart (2009)

proposed using an ensemble of SVMs to tackle with ordinal classification problems. They decompose problem into *r*-1 subproblems (where problem has *r* ordinal classes). Hence, *r*-1 binary SVMs are trained for each subproblem. They attach weights to each of the *n* training samples where weight of each sample differs for each binary classifier based on absolute difference between their rank (ordinal class) and predicted rank. All these aforementioned techniques require modification to the existing classification algorithms. To our knowledge, there exists no comprehensive study in the literature comparing these ordinal classification techniques with their nominal counterparts under different circumstances, such as data size, type or dimension, however, our experience with some of these ordinal classification algorithms showed that they did not perform well with small data. Their nominal classification counterparts outperform these techniques when they are trained on a small data set. Consequently, we think that there are still issues in ordinal classification area to be developed.

In order to tackle with ordinal classification problems by using standard classification algorithms, Frank and Hall (2001) proposed a simple approach. The main advantage of their approach is that, any base classification algorithm can be applied without making any modification. First of all, the problem having *r* ordinal classes is decomposed into *r*-1 subproblems. Hence, each of these subproblems turns out to be binary classification problems. Figure 7 illustrates transformation of a 4 class ordinal classification problem into 3 binary classification problems (Frank & Hall, 2001).

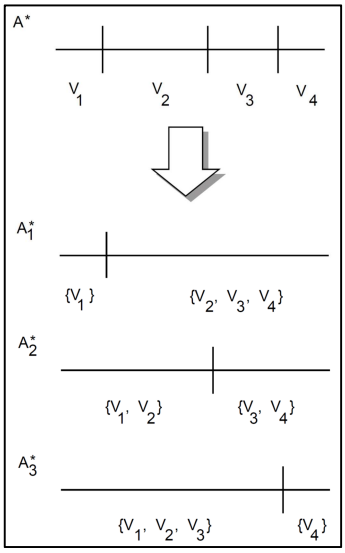


Figure 7. Transforming a 4-class ordinal classification problem into 3 binary classification problems (Source: Frank & Hall (2001)).

Each derived data set is trained with a base binary learner. Hence, we obtain *r*-1 binary learners. In order to make predictions on unseen instances, posterior probabilities *P*(.) are calculated by using *r*-1 binary learners as follows:

$$\begin{aligned} P(V_1) &= 1 - P(y > V_1) \\ P(V_i) &= P(y > V_{i-1}) - P(y > V_i) \qquad 1 < i < r \\ P(V_r) &= 1 - P(y > V_{r-1}) \end{aligned} \tag{20}$$

where V_i corresponds to the i^{th} ordered class. As a consequence, the class having the maximum posterior probability is assigned to instance. Empirical results show that the proposed technique outperforms standard classification algorithms when applied to ordinal classification problems. Another finding is that, as the number of classes increases, performance of the proposed approach increases with respect to standard classification algorithms.

2.3 Conjoint Analysis

Similar to the focus of this thesis and different than the applications proposed in MCDA, Conjoint Analysis considers preference modeling as a learning process, thereby seeks for methodologies that implement preference modeling process in an evolutionary way. In this respect, this thesis study is comparable to the approaches proposed in Conjoint Analysis (CA), rather than those proposed in MCDA. Additionally, our proposed approach addresses the problematic and unsolved issues of CA, which we will mention, shortly. This is the reason why we particularly go over CA in this section.

CA refers to set of decomposition methods that aim to estimate preference structure of the customers based on the joint effects of levels of two or more attributes (Rao, 2014). CA usually approaches preference structure modeling as a function estimation problem and uses reference evaluations of alternatives (or profiles as in CA parlance) as done in DA. By decomposing holistic preferences to partial contributions of product features (part-worths), or in other words how much each feature contributes to the overall preference, are quantitatively computed. Once part-worths are determined, not only preferences of the existing products but also preferences of future ones can be estimated.

There are mainly four types of CA methods (Rao, 2014). These are:

- The traditional method
- Choice-Based CA (CBCA)
- Adaptive CA (ACA)
- Self-explicated CA

Among aforementioned methods the first three are decomposition methods, which basically utilize disaggregation phenomenon of MCDA. Thus, these methods elicit part-worths based on holistic judgements made by the DM or customer. The last

one, on the other hand, is a compositional method. Utilizing aggregation phenomenon of MCDA, this method estimates preferences based on DM or customer statements regarding importance or scores of attributes and their levels.

In the traditional method, preferences are demanded from the customers in the ratio scale. In order to collect preference information, usually a design is constructed that includes all possible alternatives. These alternatives are called full profiles. Based on the assumed functional model and preference values obtained from the customer, regression-like techniques are used to build the preference model. The main drawback of this approach is that, in the presence of many attributes, a full profile (full factorial) design becomes very large. In order to overcome this issue, smaller designs (such as fractional factorial design) are utilized. Another disadvantage of this method is that, an underlying functional form representing preference structure of the customer is to be defined a priori.

In the CBCA, the preference information is obtained in terms of stated choices, rather than ratings of alternatives. Hence, the customer states a choice among 4 or 5 alternatives. The main advantage of this approach is its ability to mimic actual marketplace choices that people make (Rao, 2014). In CBCA, usually Multinomial Logit Model (MNL) is used to build a preference model. Similarly, CBCA suffers from large number of attributes.

In general, models estimated in traditional method and CBCA are parametric ones, therefore, as the number of attributes increase, the parameters to be estimated increase as well. Consequently, a full profile (and even a partial profile) design gets very large which is impractical for customer to answer. In order to deal with curse of dimensionality, ACA methods are developed. The main idea of ACA is to ask as less questions as possible while reducing uncertainty for the parameter estimates. In order to achieve this, methodology adaptively selects a new question based on previously obtained answers. Thus, ACA aims to get as much

information as possible while asking as less questions as possible. This approach is particularly beneficial for on-line questionnaires. On-line questionnaires are characterized by respondents that can leave the questionnaire at any time. Therefore, asking questions efficiently is of utmost importance in on-line environment.

There are many studies in the literature dealing with CA, particularly ACA. In polyhedral methods, for instance, a polyhedron that is formed by feasible values of part-worths is considered. Hence, questions are selected to reduce this polyhedron as fast as possible. Toubia et al. (2003) proposed a method that utilizes polyhedral approach where questions are selected to maximally reduce the volume of this polyhedron and minimize the length of its longest axis. Then, a technique called analytic central estimation is used to find values of the middle-most (values that are closest to the center of the polyhedron) parth-worths. Based on the similar idea, a probabilistic version of this study is proposed, as well (Toubia et al., 2007). Abernethy et al. (2008) proposed a robust ACA approach where response errors are considered in order to prevent future questions from being influenced by the previous erroneous answers. Cui and Curry (2005) used SVM as the base learner in CA and compared with other commonly used parametric methods. Evgeniou et al., (2005) proposed using SVM in developing a polyhedral method for preference modeling. Their approach is similar to that of Cui and Curry (2005) in that their model is based on formulating the problem of preference modeling as an optimization problem where an appropriate cost function is minimized without assuming a particular probabilistic model. The last two studies that utilize SVM are not adaptive methods.

As Rao (2014) states, ACA methods cannot model interactions among attributes, which is generally criticized. Traditional methods and CBCA handle interactions, however, proper underlying functional forms must be determined, which is usually unknown. Additionally, they strongly suffer from curse of dimensionality. This

issue is raised by Teichert and Shehu (2013), where they remark that model-free approaches should be developed in ACA in order to avoid model misspecifications and estimation biases.

2.4 Active Learning

Being a special case of semi-supervised learning, AL aims interactively querying the DM (or Oracle as in AL parlance) to learn about a phenomenon in an efficient way. “Passive” learning, on the other hand, learns from a training data that is collected in an unstructured way. Based on the training data obtained thus far, AL aims to minimize uncertainty regarding the data, hence, develops a “line of inquiry” in an evolutionary way (Settles, 2012). Consequently, main objective of AL is to obtain as much information as possible by querying as less questions as possible. In this respect, AL can be considered as ML equivalent of ACA and optimal experimental designs. However, AL is superior to ACA and optimal experimental designs due to the fact that AL can be utilized with nonparametric learners, hence, there is no need to assume a functional form for the phenomenon of interest. Having this property, it is a perfect choice for learning complex structures in an evolutionary and effective way.

The main motivation behind AL is that, there is usually cheap and abundant unlabeled data available, whereas, labeled data is scarce and obtaining is expensive. Expensiveness may denote human effort, time or money. Whatever the reason is, in the absence of plenty of labeled data, information embedded in the unlabeled data is exploited to boost supervised learning process.

AL strategies can be applied to both regression (continuous response) and classification problems. However, classification problems are widely studied in the literature. This is due to the fact that, classification problem draws more attention in the ML and AI fields.

In AL, there are mainly three different scenarios or ways in which the learner may ask queries (Settles, 2012):

- **Query synthesis.** In this scenario, any unlabeled data in the data set can be queried. The only requirement regarding this scenario is that the learner knows the feature (attribute) domain.
- **Stream-based selective sampling.** In this scenario, the learner samples an unlabeled instance from the input distribution, however, decides whether or not to query it afterwards. In this case, drawing an unlabeled instance from the input distribution is assumed to be free. Success of this scenario is closely related to form of the input distribution. If the input distribution is uniform, stream-based sampling is no different than the query synthesis. Conversely, if the input distribution is not uniform, then sampled instances will contain information about the structure of the input distribution.
- **Pool-based sampling.** In this scenario, candidate instances to be queried constitute a closed set, called the pool. All instances in the pool are evaluated based on a utility measure. The instance providing the most information regarding the measure is selected as the next query.

There are several AL strategies proposed in the literature. Most commonly used ones are (Settles, 2012):

- Uncertainty sampling
- Query by disagreement
- Query by committee
- Expected error reduction
- Variance reduction
- Density weighted methods
- Cluster based AL

Query By Disagreement (QBD) approach utilizes the idea that there might be several different learners that model the data the same way as supervised (in other words, predicting reference set accurately). This phenomenon is similar to the alternative optimal solutions (alternative utility functions) problem encountered in UTADIS and many other MCDA approaches where algorithm ends up with many different utility functions that sort the reference set the same way as it is done by the DM. QBD employs all these alternative learners for selecting a new question and queries the one for which the learner set disagrees most. The main disadvantage of this approach is that it considers all alternative (optimal) learners, which is computationally inefficient.

Expected Error Reduction (EER) aims for choosing a question that will provide the most gain in terms of reducing future error. The idea is that, if we are able to identify all the possible outcomes and corresponding probabilities, then we can calculate a weighted sum for expected future errors for each action (Settles, 2012). This approach requires probabilities for each DM answer to a question and the probabilities of making error on other instances once the answer of a particular question is known. In general, EER uses model's posterior distribution to estimate these unknown probabilities. This technique is criticized for being computationally expensive (Settles, 2012).

Similar to EER approach, Variance Reduction (VR) aims at querying an instance that will reduce the output variance over the unlabeled instances most. VR corresponds to optimal experimental design of statistics and employs Fisher information to calculate output variances. In order to employ Fisher information, a parametric model assumption is made and for a model having K parameters, $K \times K$ covariance matrix is computed. In general, this approach is used when parametric learners are utilized.

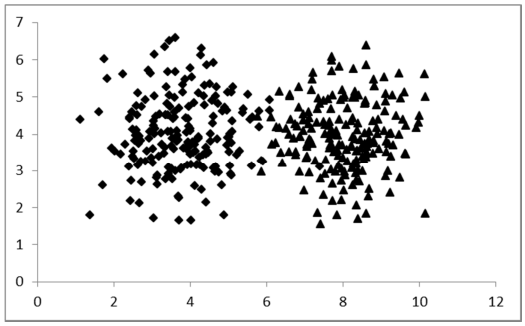
Density Weighted Methods (DWM) and Cluster based AL (CAL) exploit input distribution of unlabeled instances to expedite the learning process. DWM considers not only information content of a candidate question but also how much it represents other unlabeled instances. CAL, on the other hand, clusters the data and queries those instances that represent others in the same cluster most (i.e. cluster centroid). These approaches assume that we have a set of unlabeled instances that constitute an input distribution.

Refer to Settles (2012) for extensive definitions of these strategies. In the subsequent subsections, we will provide detailed information about Uncertainty Sampling (US) and Query By Committee (QBC), which we utilize in this study. We will explain in detail why we prefer these two techniques in Chapter 3.

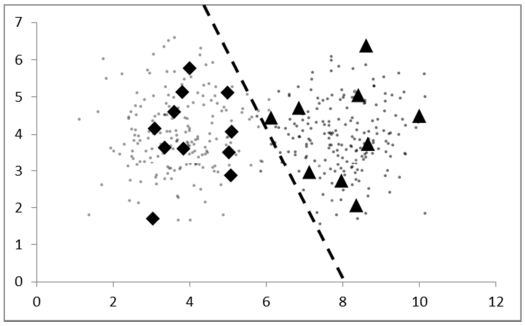
2.4.1 Uncertainty Sampling

Learners for the classification task generally determine decision boundaries which separate different classes. In general, learners are most confident about the class information of instances that are away from these decision boundaries, while least confident about labels of instances that are close to these decision boundaries. This is the main idea behind US; querying instances that are close to decision boundaries provides the most information gain. Hence, learner should concentrate on these kinds of instances.

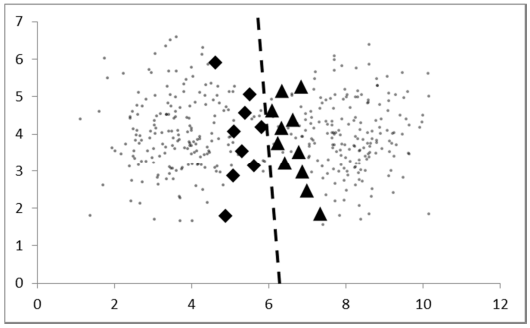
How US works is illustrated in Figure 8 (Settles, 2012). Assume a two-class data is scattered on the attribute domain as shown in Figure 8 (a). Figure 8 (b) shows label information of a reference set that is constructed randomly. Decision boundary determined by the learner is indicated with the dotted line. In Figure 8 (c), we see labels of another reference set of same size, constructed by using US. In this case, we see that learner provides a better decision boundary than the previous case since US concentrates its querying effort where uncertainty is most.



(a) A toy data set.



(b) Random sampling.



(c) Uncertainty sampling.

Figure 8. Example of US strategy (Adapted from Settles (2012)).

The most important component of this querying approach is to measure the uncertainty of candidate instances. As we recall, classification algorithms compute posterior (class membership) probabilities $P(\hat{y}_i \mid x)$ for each possible outcome (class) i in order to make predictions for an instance x. Then the class label having the biggest posterior probability is stated as the prediction. These measures utilize posterior probabilities to quantify uncertainty. There are mainly three measures of uncertainty used in the literature (Settles, 2012):

- **Least confident.** This measure considers minimum (least confident) of highest posterior probabilities of candidate instances:

$$\begin{aligned} x_{LC}^{*} &= \arg\min_{x} P_{\theta}(\hat{y} \mid x) \\ &= \arg\max_{x} 1 - P_{\theta}(\hat{y} \mid x) \end{aligned} \tag{21}$$

where $\hat{y} = \arg\max_{y} P_{\theta}(y \mid x)$ is the prediction with the highest posterior probability under the model θ. The rationale for this measure is that, among the predictions we make for a set of candidate instances, we are least confident about the instance whose most likely labeling is the least likely. Hence, querying this instance provides the most information gain. The main disadvantage of this measure is that it does not take into account posterior probabilities of other classes.

- **Margin.** This measure recommends querying the instance,

$$\begin{aligned} x_{M}^{*} &= \arg\min_{x} \left[P_{\theta}(\hat{y}_1 \mid x) - P_{\theta}(\hat{y}_2 \mid x) \right] \\ &= \arg\max_{x} \left[P_{\theta}(\hat{y}_2 \mid x) - P_{\theta}(\hat{y}_1 \mid x) \right] \end{aligned} \tag{22}$$

where $\hat{y}_1$ and $\hat{y}_2$ are the first and second highest posterior probabilities of the model. Margin measure considers that, if the margin between the first and second highest posterior probabilities is big, then the classifier is confident about its prediction. However, if the margin is small, then the classifier is not much confident about differentiating these two class labels. Hence, querying the instance having the minimum margin provides the most information gain.

- **Entropy.** This measure recommends querying the instance,

$$x_H^* = \arg\max_x H_\theta(Y \mid x) \\ = \arg\max_x -\sum_y P_\theta(y \mid x).\log P_\theta(y \mid x) \quad (23)$$

where *y* ranges over all possible labelings of *x*. Entropy (*H*(.)) corresponds to a variable's average information content (Settles, 2012). In ML practices, it is considered as an impurity measure. Thereby, this measure recommends querying an instance having the maximum information content.

2.4.2 Query By Committee

QBC approach constructs a committee of learners as in ensemble learning. As we have stated previously, there is no best algorithm that outperforms others in all circumstances. Additionally, overfitting phenomenon is encountered frequently, which we struggle to avoid with regularization. These facts are the main motivation behind ensemble learning, thus, ensemble learning aims to take advantage of more than one learner so that prediction performance increases. In ensemble learning, several learners are trained on the same reference data set. In the prediction phase, predictions are made based on the majority vote. QBC exploits this idea and measures disagreement of committee members with respect to candidate instances. To illustrate the idea, consider a binary classification case where data points ("-" for Class 1, "+" for Class 2) are scattered as shown in Figure 9. Assume that we partition the data as training and test sets and train 4 classifiers (rectangles) on the same training set. Also assume that these classifiers predict instances lying in the rectangles as Class 2 and outside as Class 1. In this case, there is a consensus among classifiers regarding the instances lying in the dark gray zone since all classifiers predict them as Class 2. Now consider the instance where we designate with a dark gray circle. Two of the classifiers predict this instance as Class 1, while other two predict as Class 2. Therefore, disagreement of classifiers regarding this instance is significant because two of the

classifiers disagree with other two. Hence, querying this instance will provide more information gain than those lying on the gray zone.

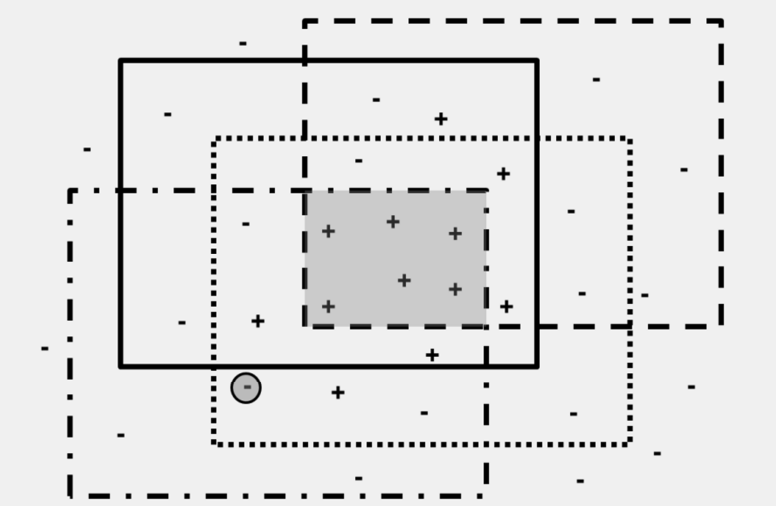


Figure 9. Example of QBC strategy.

There are different methods to generate members of the committee. One option is to use different type of classifiers, such as SVM, RF or NN. On the other hand, several other alternatives are proposed in the literature: varying case weights, data values, guidance parameters, variable subsets, or partitions of the input space (Seni & Elder, 2010). One straightforward technique for generating a committee, as done in RF learning, is using bagging (Breiman, 1996). Bagging aims at reducing the variance and increasing the prediction accuracy by taking many samples of same size from the original training set with replacement (bootstrapping), fitting separate but same type of learners to each training set and averaging the predictions. Querying strategy using bagging algorithm is called Query By Bagging (QBB) and first introduced by Abe and Mamitsuka (1998). One of the

advantages of QBB algorithm is that, it employs bootstrapping, therefore it is robust to small sample sizes, unlike boosting algorithm. Boosting is a ML approach where performance of the weak learners is boosted by repeatedly resampling on the training data (Freund & Schapire, 1997). Each time resampling is performed, data points are weighted in order to focus on those points where classifiers perform poorly. Boosting algorithm usually fails in the presence of small data sets because when trained on small data, classifiers usually predict the training data accurately, hence re-weighting phase cannot be performed. There are mainly two disagreement measures used in QBB (Settles, 2012):

- **Vote entropy.** This measure recommends querying the instance;

$$x_{VE}^{*} = \arg\max_{x} -\sum_{y} \frac{vote_{C}(y,x)}{|C|} . \log \frac{vote_{C}(y,x)}{|C|} \tag{24}$$

  where $vote_{C}(y,x) = \sum_{\theta \in C} 1_{\{h_{\theta}(x)=y\}}$ is the number of votes that the label y receives for x among the learners in committee C. This measure is committee counterpart of entropy measure. Instead of using posterior probabilities, vote entropy considers votes of every single committee member for labels.

- **Kullback-Leibler (KL) Divergence.** KL Divergence (or information gain) is the distance between two probability distributions (in general, true and target distributions) (Kullback & Leibler, 1951). In QBB context, KL Divergence measures disagreement as the average divergence of each committee member θ's prediction from that of the consensus C and recommends querying the instance:

$$x_{KL}^{*} = \arg\max_{x} \frac{1}{|C|} \sum_{\theta \in C} \sum_{y} P_{\theta}(y \mid x) . \log \frac{P_{\theta}(y \mid x)}{P_{C}(y \mid x)} \tag{25}$$

In other words, rather than considering mean posterior probabilities of the committee, KL Divergence measures how much each committee member diverge from the consensus. Even though mean posterior probabilities indicate that committee is confident about its prediction with regard to particular instance, KL Divergence measure might recommend querying it if committee members' posterior probabilities vary wildly from the consensus.

QBC can be used in the continuous case as well. In this case, variance of the predictions of the committee members can be used as the disagreement measure.

CHAPTER 3

PROPOSED APPROACH

3.1 Motivation

Aiming at explicitly eliciting preferential system of the DM, preference modeling is drawing a growing interest recently due to the fact that it became an imperative step in many areas. Particularly if we are interested in making predictions, we need robust preference models that represent preference structure of the DM. As we pointed out previously, making predictions for the preference of some alternative/solution is particularly important in marketing and manufacturing fields.

In many preference modeling problems where multiple interacting criteria exist, the DM needs to express his/her preferences in the ordinal scale. In these cases, typically no information is available about the underlying preference structure, and only a few data can be collected about the preferences of the DM. Additionally, data collection process (or generating reference alternatives) is usually costly in terms of time or money. In all these cases, preference modeling turns out to be a learning process where as much information as possible is required by asking as less preference questions as possible.

Preference modeling gets more complicated in the presence of interacting criteria. However, interaction phenomonea is encountered quite commonly, even in simpler cases. Even though there is a general consent among researchers regarding the existence of interaction among criteria in real life decision problems, it is often ignored in applications. Most of the preference modeling strategies assume preferential independence among criteria in order to make modeling process relatively easygoing.

Another problematic issue with most of the proposed MCDA approaches that model preference structure implicitly or explicitly is that, they assume an underlying functional form that is believed to conform to DM preference structure. Additionally, most of these functional forms are monotonous. Even though it is commonly accepted in the literature that rational human preferences present monotonically increasing or decreasing structure, there are examples of nonmonotonic preferences in real life. Regarding this issue, let us look at a nonmonotonic utility function example given by Keeney and Raiffa (1993). This example considers the blood sugar count of a patient. There is a “normal” or preferred blood sugar count that is desired. A sugar count below the normal figure is not preferred, however, a sugar count that is higher than the normal is more dangerous, hence the least preferred. Such a nonmonotonic utility function is represented in Figure 10 (Keeney & Raiffa, 1993).

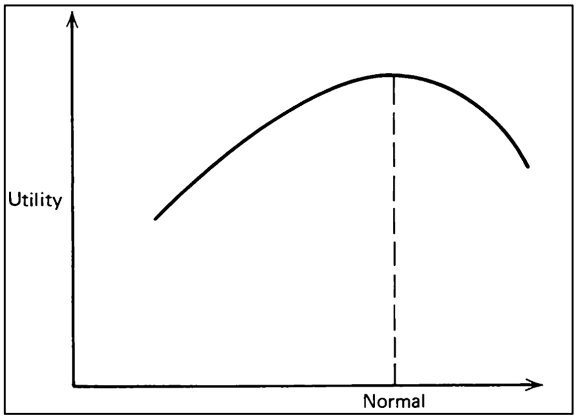


Figure 10. A nonmonotonic utility function (Source: Keeney & Raiffa (1993)).

Parametric approaches usually fail to model interactions, especially if a wrong functional form is assumed. A wrong functional form produces estimation bias and reduces the predictive performance of the model. In this respect, a straightforward approach might seem to be partitioning the criterion domain where preference shows somewhat monotonic behavior and then model preference corresponding to each of these domains with a monotonic function. However, it is not generally that simple to partition criteria domain where preference is monotonous, particularly in higher dimensions and in the presence of multi-way interactions. Consider a simple case, where we have two interacting criteria and preference is expressed in the ratio scale. Assume that preference surface is as illustrated in Figure 11. Additionally, contour plot of this preference surface is provided in Figure 12. As we see from Figure 11, the preference is nonmonotonic. Contour plot of the preference structure present a complex structure on the criteria domain which is hard to estimate.

These two figures reveal that even for the simplest case where preference depends on two interacting criteria, determining boundaries of the criteria domain where preference shows monotonic behavior is a hard task. Moreover, we probably

would have to ask many questions just to determine these boundaries. Restating, it becomes a harder task in the presence of more than two interacting criteria. This fact leads us to search for a better methodology in order to deal with nonmonotonocity and interacting criteria.

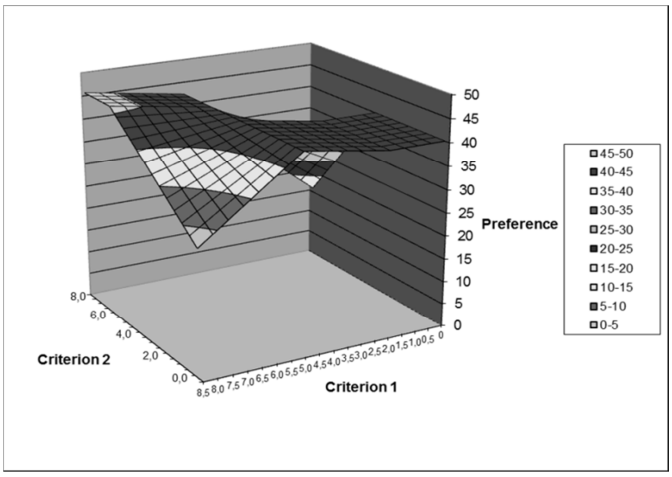


Figure 11. Preference surface of a two-criterion value function.

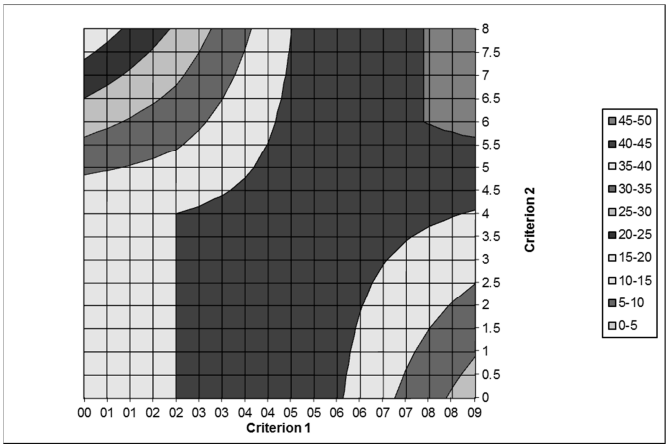


Figure 12. Contour plot of the preference surface of the two-criterion value function shown in Figure 11.

The main objective of preference modeling is to build predictive models that will be used to make predictions on unseen instances. Models with good generalization ability are expected to perform well on unseen instances. Not receiving enough interest in MCDA, generalization is one of the key issues in SL (particularly in AI) and CA. Apart from prediction, developed models can also be used for optimization. In a quality improvement problem, for instance, optimal process parameter levels are to be selected so that quality of a product, which is defined by the joint effects of all quality characteristics, is optimized. Quality characteristics usually interact with each other, therefore, models that are capable of dealing with interacting criteria may be required.

Restating, preference modeling is a learning process that should be handled evolutionary. Almost all of the MCDA approaches obtaining preference information in the ordinal scale aim at sorting limited number of alternatives of the problem under consideration with maximum accuracy. They take a subset of the alternatives or use a separate reference set that happens to be available for getting preference information. Additionally, instances in the initial reference set are not collected in a structured and evolutionary way so that subsequent learning process is expedited. Hence, the preference model that is developed based on the preference information obtained with respect to reference alternatives is used to sort the rest. In this respect, their generalization ability is limited. Even though developed models represent preference structure of the DM, these techniques cannot be considered as preference modeling approaches. Additionally, they cannot model nonmonotonic preference structures having complex interactions among criteria. ACA approaches, on the other hand, consider preference modeling as a learning process, where preference information is obtained in an evolutionary way and questions are structured in order to obtain as much information as possible by asking as less questions as possible. However, Rao (2014) states, ACA methods are model dependent and cannot model interactions among attributes. Additionally, they strongly suffer from curse of dimensionality.

Consequently, all these aforementioned discussions reveal that, there is a need for new preference modeling methodologies that approach the problem as a learning process, are able to model nonmonotonic structures having complex interactions successfully and have good predictive performance while training on small data.

3.2 Proposed Approach

In this section, we provide detailed information about our approach for modeling DM preference structure. In this study, we consider the case where DM expresses her/his preferences in the ordinal scale. For the DM, this task is a sorting or ordinal classification task after all, where DM assigns preference ordered class labels to alternatives of profiles, hence, preference structure of the DM is represented by the trained classifier. Using the classifier, one can make predictions on unseen instances. In order to illustrate the idea, consider the two-criteria nonmonotonic value function example presented in Section 3.1. Let us assume that preference of the DM conforms to an unknown continuous functional form as shown in Figure 13. For a sorting problem, there are some threshold values, which are explicitly unknown to the DM her(him)self, that separate preference values into classes, thus, creating an ordinal relationship among them. These hypothetical threshold values are presented in Figure 13 for a three class ordinal classification problem.

In this setting, the main objective of a trained classifier is to find projections of these threshold values on the criteria domain, which are implied by the DM via answers to the alternatives presented. Without prior knowledge about underlying functional form of the preference and threshold values separating classes, classifier constructs boundaries on the criteria domain that separate domain into classes as shown in Figure 14. At the end of the training phase, trained classifier can be used as a predictive model and an optimization aide since it designates regions on the criteria domain where preference is most.

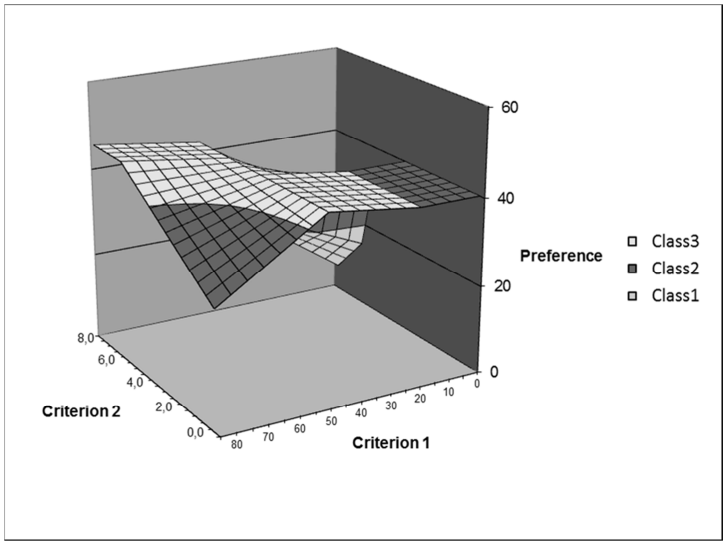


Figure 13. Preference surface of a two-criterion value function for a three-class ordinal classification problem.

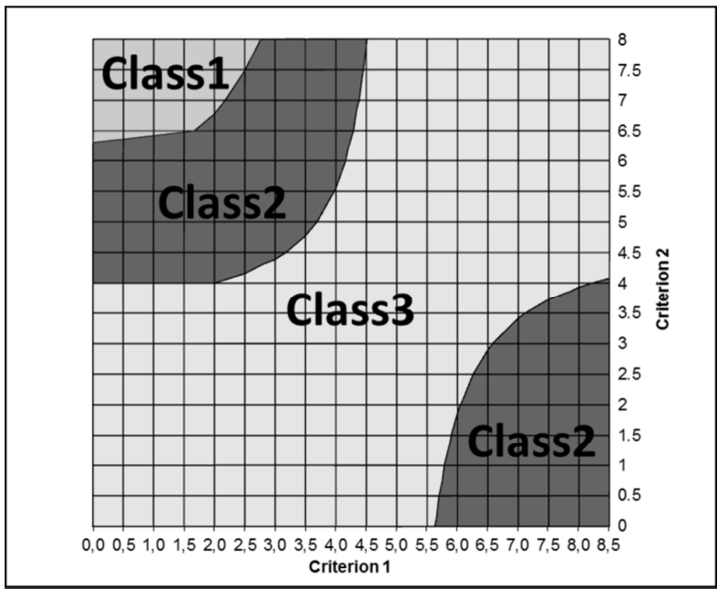


Figure 14. Criterion domain of the value function of Figure 13 separated into preference ordered classes by the trained classifier.

In our approach we assume that we do not have any previous knowledge about preference structure of the DM. Moreover, we have no information about how criteria values of the potential alternatives are distributed (input distribution). AL techniques, on the other hand, are developed to query DM on a data set that is present. In other words, we have abundant unlabeled instances and small number of labeled instances obtained from some source. All candidate data points that can be queried constitute a closed data set. Hence, we can talk about an input distribution where data comes from. In this respect, “Density Weighted Methods” and “Cluster Based AL” techniques aim to exploit the structure of the whole data set. This is the reason why AL practitioners put special emphasis on how much a candidate data point represents remaining unlabeled data points in the data set. As a consequence, the data set that we need to predict or classify is present (unlabeled instances) and we do not want to spend our query resources with those that do not represent other points in the whole data set (outliers). In our case where we model DM preferences, however, we cannot talk about an input distribution since we do not have an unlabeled data set. This is particularly valid in the marketing field. In marketing, the main objective is to model customer preferences with respect to presented products. We tailor attributes of the alternative products in order to estimate customer preferences. Therefore, we aim to model preference on the whole attribute domain (design domain) where we desire to perform equally good everywhere. This requirement makes our problem a more challenging one with respect to classical AL practices. On the other hand, we provide an extension to our algorithm for special cases where we can exploit information provided by the unlabeled data set.

We employ SVM and RF as the base learners in the algorithm. From variety of learners, we prefer SVM and RF because they are among the strongest nonparametric classifiers proposed in the literature. Additionally, our experience showed that most of the nonparametric classifiers perform bad or fail with small data sets. Conversely, SVM and RF model small data sets very successfully. RF is

very robust to overfitting (Hastie et al., 2009). This is due to its ensemble structure, i.e., averaging predictions of many grown bagged trees. This property becomes very important when size of the training data is small because classifiers usually suffer from overfitting in the presence of small data sets. Another valuable property of RF is that, similar to other decision tree based learning methods, RF handles qualitative predictors very easily, without creating dummy variables.

SVM and RF are nominal classifiers, meaning, they are developed to handle classes that do not have order relation. However, DM preferences are ordinal in nature, hence, there exists additional information to be exploited, as stated by Hühn and Hüllermeier (2009). Even though there are several learners proposed in the literature for ordinal classification, to our knowledge, there exists no comprehensive study in the literature comparing these ordinal classification techniques with their nominal counterparts under different circumstances, such as data size, type or dimension. However, our experience with some of these ordinal classification algorithms showed that they did not perform well with small data. Their nominal classification counterparts outperform these techniques when they are trained on a small data set. Due to these reasons, we apply Frank and Hall (2001) approach to SVM and RF for multiclass (more than two classes) cases in the study. The main advantage of Frank and Hall (2001) approach is that, it can be used with any standard classification algorithm without modifying it. Additionally, our experiments prove that when used with Frank and Hall (2001) approach, SVM and RF outperform their standard nominal counterparts if the training data have ordinal classes.

Initially, we do not have any reference alternative having class information at hand. First of all, we ask DM how many ordinal classes (s)he wants to consider or determine the number of classes based on the problem under consideration. In order to generate an initial reference set to start with, we perform a stratified random sampling where we ask DM to show random examples from each class.

Size of the reference set depends on willingness of the DM and characteristic of the process of interest (i.e., easiness of providing examples, number of classes and attributes). However, we generally demand a small reference set since we aim to ask small number of questions to the DM and thereby consume our questions thriftily. After constituting our reference set, we start running our algorithm. Our algorithm asks one question at a time and tailors the new question based on the answers obtained thus far (that is the current reference set comprising of initial reference set and alternatives asked to that point). In order to determine next question to be asked, we utilize AL techniques and implement Pool-Based US and Pool-Based QBC with different uncertainty measures, namely, Least Confident, Margin and Entropy. In applying QBC, we use QBB approach proposed by Abe and Mamitsuka (1998).

Among 3 different querying scenarios we prefer using pool-based sampling. Stream-based selective sampling requires a set of unlabeled instances that constitute the input distribution. As stated previously, we cannot talk about an input distribution in preference modeling. In query synthesis, on the other hand, any unlabeled data point from the attribute domain can be queried. This technique is effective if we know additional information about the attribute domain, so that asking questions from some particular region boosts the learning process. Thereby, this scenario is not applicable to our problem. Consequently, pool-based sampling is the best alternative because we would like to perform equally good everywhere on the attribute domain in terms of prediction ability. By using pool-based sampling, we draw random samples from the overall domain in order to form candidate questions (alternative profiles) pool. Among these profiles, we query the one that will provide the most information gain.

In Chapter 2, we mentioned several different querying strategies used in AL. Among these, DWM and CAL strategies exploit information provided by the input distribution of unlabeled instances. As we emphasized previously, we do not have

an input distribution under consideration in our problem setting. VR strategy employs Fisher information in order to compute output variances. This approach is applicable when we use parametric classifiers since Fisher information needs design matrix expanded to the form of the parametric model that we assume. QBD and EER approaches are criticized for being computationally inefficient (Settles, 2012). On the other hand, US and QBB strategies do not require parametric models or input distribution while they are computationally efficient. Additionally, QBB is robust to small sample sizes because it employs bootstrapping. Due to these reasons, we prefer US and QBB querying strategies in our study.

The most important component of the proposed approach is to measure the uncertainty of candidate instances. In Chapter 2, we explained main uncertainty measures proposed in the literature. In this study, we developed and experimented several uncertainty measures which would also exploit ordinal structure of the preference classes. However, developed measures performed worse than those proposed in the literature. We will provide more information about developed measures in the subsequent section. Among those measures proposed for the QBB strategy, Vote Entropy utilizes a hard voting approach where the number of votes given by the committee members is counted in order to compute uncertainty of each candidate instance. This voting scheme loses information offered by the posterior probabilities of predictions made by each committee member. Kullback-Leibler Divergence measure, on the other hand, gives instable results due to log(.) term in the formulation, because when committee member's posterior probability ($P_\theta(y \mid x)$) is less than the consensus posterior probability ($P_C(y \mid x)$), this term yields to a negative value. However, a positive divergence from the consensus posterior probability of same amount (when $P_\theta(y \mid x)$ is greater than $P_C(y \mid x)$), which yields to a positive value, should have the same uncertainty value. Due to these reasons, we utilized a soft voting scheme for prediction aggregation in QBB and used the same uncertainty measures (Least Confident, Margin and Entropy) for both US and QBB querying strategies.

As we emphasized in the preceding paragraphs, we propose and experiment different classifiers, querying strategies and uncertainty measures for preference modeling. However, determining which classifier, querying strategy and uncertainty measure to use for which circumstances is the most important ingredient of this thesis study. For this purpose, we will provide algorithmic recommendations for the end user in Chapter 4. Below are the pseudo codes and commentary of our algorithms.

1. ***R* is the initial reference set of size *n*, *m* is the number of queries, *L* is the learner, *p* is the pool size**
2. **for** i=1, 2, …m **do**
3. train L using R
4. generate random set of instances from the attribute domain of size p for the *pool*
5. predict labels of instances in the *pool* using L
6. calculate uncertainty measure for each instance in the *pool*
7. select instance having the maximum uncertainty (w^*)
8. Ask label (class) y of w^* to the DM
9. $R \leftarrow R \cup \langle w^*, y \rangle$
10. $n \leftarrow n + 1$
11. **end for**

Figure 15. Pseudo-code of the Pool-Based Uncertainty Sampling Algorithm.

1. ***R* is the initial reference set of size *n*, *m* is the number of queries, *L* is the learner, *p* is the pool size, *C* is the committee of learners**
2. **for** *i*=1, 2, …*m* **do**
3. generate random set of instances from the attribute domain of size *p* for the *pool*
4. **for** j=1, 2, …|*C*| **do**
5. draw a bootstrap sample of size *n* from the reference set (R_j)
6. train L_j using R_j
7. predict labels of instances in the *pool* using L_j
8. **end for**
9. aggregate predictions made by the committee ($L_1, \ldots, L_{|C|}$)
10. calculate uncertainty measure for each instance in the *pool*
11. select instance having the maximum uncertainty (w^*)
12. Ask label (class) *y* of w^* to the DM
13. $R \leftarrow R \cup \langle w^*, y \rangle$
14. $n \leftarrow n + 1$
15. train *L* using *R*
16. **end for**

Figure 16. Pseudo-code of the Pool-Based Query By Bagging Algorithm.

Generating the initial reference set. Generating the initial reference set constitutes the first stage of the preference learning. Based on the preference information obtained from this set, line of inquiry is initiated. This set is generated by stratified random sampling, meaning, labels of random instances from each preference class is asked to the DM. Alternatively, DM can be encouraged to provide such exemplary instances from each preference class. Size of the initial reference set depends on the number of classes and attributes as well as willingness of the DM to provide such examples. Since we perform stratified

random sampling, number of exemplary instances must be greater than the number of preference classes. Additionally, as the number of attributes increases, size of the reference set must increase as well. Another limiting factor is the ability of the learner to train on small data sets. Most of the nonparametric learners fail to train on small sets. SVM and RF are very capable with this regard, however, our experiments show that when the number of training instances is less than 10, the probability that these learners fail increases. Hence, we can assert that a reference set of minimum size 10 is preferable.

Training the learner. We use Frank and Hall (2001) approach when training our base learners SVM and RF for the multiclass cases while we employ them directly in the two-class cases. First of all, the training (reference) set having *r* ordinal classes is decomposed into *r*-1 binary subproblems in multi-class cases. Hence, each of these subproblems turns out to be binary classification problems as illustrated in Figure 7. Each derived data set is trained with a base binary learner. Hence, we obtain *r*-1 binary learners. In order to make predictions on unseen instances, posterior probabilities *P*(.) are calculated by using *r*-1 binary learners as shown in (20). As a consequence, the class label having the maximum posterior probability is assigned to the instance.

Generating the pool. Pool corresponds to the set of candidate instances from which the next question to be asked to the DM is selected at each iteration. Since we aim at modeling preference on the whole attribute domain (design domain) while desiring to perform equally good everywhere, we generate candidate instances randomly. Hence, assuming that we have *n* criteria under consideration, criteria values of each candidate instance in the pool are determined such that:

$$x_i \sim U(x_i^{\min},\ x_i^{\max}) \quad \text{for} \quad i = 1,\ldots,n \tag{26}$$

where x_i is the random value attained in criterion i, $x_i^{\min}$ and $x_i^{\max}$ are the minimum and the maximum criterion values attainable in criterion i respectively, and $U(.)$ is the uniform distribution. In order to ensure diversity, a new pool is generated at each iteration. Even though a large pool is desirable so that we increase the probability to find an instance that will provide the most information gain, it is computationally costly. We did not perform an experiment as to measure the effect of pool size on performance of the algorithm, however, we generated a pool of size 500 in our experimental studies.

Making predictions and measuring uncertainties of the instances in the pool. After training the learner with the reference set (including instances queried thus far), the learner makes predictions for each instance in the pool based on the criteria values of the instances. Each prediction is quantified with posterior probabilities provided by the learner. These posterior probabilities are used to measure uncertainty of each instance as explained in Chapter 2. We consider Least Confident, Margin and Entropy as the uncertainty measures.

Generating the committee in QBB. In employing QBB, we need a committee of learners. This is achieved by drawing bootstrap samples from the same reference set $|C|$ times (size of the committee). Bootstrapping corresponds to random sampling with replacement of same size as the original sample. The main idea behind bootstrapping is that, when we cannot make parametric assumptions about underlying distribution that generated the random sample, the sample can be treated as a pseudo-population (Martinez & Martinez, 2002). In QBB, same type of learner (i.e. SVM or RF) is trained to the generated bootstrap samples, hence, we come up with $|C|$ separate models. Size of the committee has impact on the performance of the predictions made by the committee in return for a computational cost, however, committees of 5 to 15 learners are quite common in the literature and have been shown to perform well (Settles, 2012). Thereby, at each iteration we generate a committee of size 10.

Aggregating predictions made by the committee in QBB. At each iteration, each committee member makes predictions for the candidate instances in the pool. These predictions are aggregated in order to measure information content of each candidate instance. We use a soft voting scheme for QBB strategy which accounts for each committee member's (θ) confidence ($P_\theta(y \mid x)$) in predictions made such that:

$$P_C(y \mid x) = \frac{1}{|C|} \sum_{\theta \in C} P_\theta(y \mid x) \tag{27}$$

where $P_C(y \mid x)$ corresponds to the consensus probability that *y* is the correct label, and *C* is the committee (Settles, 2012). Consequently, uncertainty of each candidate instance is computed based on the consensus posterior probabilities $P_C(y \mid x)$.

In this study, we employ SL techniques in preference learning in the ordinal scale aiming to train a predictive model. Hence, preference structure is represented by the trained model. There are several advantages of SL techniques in this respect. Nonparametric SL techniques are model-free approaches, making no functional assumptions. Additionally, they are very capable of modeling complex structures, i.e. interactions among variables and nonmonotonic structures. Conversely, they require big amount of data in general and their training process is not interactive. Recognizing these shortfalls, we use SL techniques in the presence of small data set and drive the learning process in an evolutionary way while integrating DM into it. In this context, while utilizing strong features of SL in modeling complex structures, we also address the weak sides of SL criticized by Doumpos and Zopounidis (2011), in conjunction with preference modeling in the ordinal scale.

As we pointed out previously, our approach is a novel one because different than the MCDA sorting approaches proposed in the literature that aim at sorting limited

number of alternatives of the problem under consideration with maximum accuracy, we consider preference modeling as a learning process. To our knowledge, this is the first study in the MCDA literature that approaches preference modeling as an evolutionary learning process. With this study, we address the criticized issues of MCDA regarding underlying preference function and monotonicity assumptions, and inefficiency to model interacting criteria while showing applicability of SL techniques to MCDA preference modeling in an interactive manner. In this respect, rather than MCDA sorting techniques proposed in the literature, our work is comparable to AI and ACA methodologies. AI applications usually work with big amount of data. Moreover, they offer black-box methodologies where DM or respondent has no role in the training process. ACA, on the other hand, starts with a small reference set and leads the line of inquiry in an evolutionary way. However, as Rao (2014) remarks, they are incapable of handling interactions among criteria and complex structures.

As a consequence, this study can be regarded as a pioneering approach considering that SL based approaches in the literature have been developed and tested based on a relatively large preference information and do not interact with DM in model developing process, while MCDA based approaches ignore interactions, suffer from generalization ability, and have no concern about predicting equally good everywhere on the criteria domain. Even though our work is not comparable with MCDA sorting methodologies, our trained model offers a sorting tool, as well. In this respect, our proposed approach also addresses the need for model-free sorting methodologies that are capable of modeling interactions while implemented interactively with the DM, as emphasized in Table 4.

Table 4. Classification of the proposed approach in MCDA sorting.

Interactive	Functional Assumption	Ability to Model Interactions	
		No	**Yes (Limited)**
No	Yes	Devaud et al. (1980) Greco et al. (2010) Soylu et al. (2011) Greco et al. (2011) Çelik et al. (2015) Corrente et al. (2015)	
	No	Yu (1992) Leroy et al. (2011)	Greco et al. (2002) Angilella et al. (2009)
Yes	Yes	Ulu and Köksalan (2001) Köksalan and Ulu (2003) Köksalan and Özpeynirci (2009) Buğdacı et al. (2013)	Ulu and Köksalan (2014)
	No	Köksalan et al. (2009)	**Erişkin (2015)**

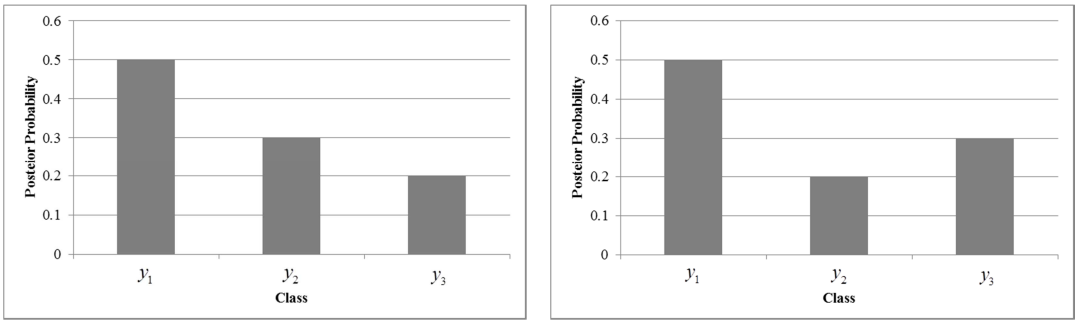
3.3 Uncertainty Measures Experimented

Uncertainty measures proposed in the AL literature have been developed for nominal classification problem, hence, they did not consider additional information embedded in the ordinal structure of the data. In the ordinal classification problem, however, there is an ordinal relationship among classes. Therefore, posterior probabilities should decrease monotonically on both sides of the class having the highest posterior probability in order to provide a sound prediction. Consider an ordinal classification problem where we have three ordered classes such that $y_1 \succ y_2 \succ y_3$. Assume that our trained model makes

predictions for two profiles (x^i, $i = 1, 2$) and generates posterior probabilities as follows:

$$P(\hat{y}_1 \mid x^1) = 0.5,\ P(\hat{y}_2 \mid x^1) = 0.3,\ P(\hat{y}_3 \mid x^1) = 0.2$$
$$P(\hat{y}_1 \mid x^2) = 0.5,\ P(\hat{y}_2 \mid x^2) = 0.2,\ P(\hat{y}_3 \mid x^2) = 0.3 \tag{28}$$

It is clear that, both of these profiles have same uncertainty values in terms of Least Confident, Margin and Entropy measures. Therefore, these uncertainty measures conclude that these two profiles have the same information content. However, consider Figure 17 where we plot posterior probabilities predicted by the trained model. Our trained model predicts both of these profiles as class y_1. Second most likely class predicted for profile 1 is y_2, while it is y_3 for profile 2. Figure 17 shows that posterior probabilities predicted for profile 1 conform to ordinality, while those of profile 2 do not. Since our problem is an ordinal classification problem, querying label of profile 2 is expected to provide more information gain than profile 1.



(a) Profile 1 (b) Profile 2

Figure 17. Bar charts of posterior probabilities predicted by the trained model.

In our study, we developed and experimented new uncertainty measures for the multi-class cases exploiting the idea explained above. In this section, we provide information about these measures. Consequently, we aimed to exploit ordinal structure of the preferences in selecting the most informative profile or in calculating consensus output distribution in QBB.

Measure 1. In this measure, we define a triangular penalty function for punishing divergence from ordinality. Each posterior probability is multiplied with penalty value depending on the most likely class (class having the maximum posterior probability). Each profile in the pool is given penalty accordingly. Uncertainty value is weighted with penalty value of the profile as formulated in (29).

$$\begin{aligned} x^* &= \arg\max_x \left(\varphi(x).Pn(x)\right) \\ Pn(x) &= \sum_y P(y|x).\delta(y \mid y^*) \end{aligned} \tag{29}$$

$\delta(y \mid y^*)$ represents penalty value for class y for a given most likely class y^*, $\varphi(x)$ represents uncertainty value for profile *x* and *Pn*(*x*) corresponds to penalty for profile *x*. Hence, a profile not complying with ordinality has more chance to be queried. Triangular penalty function is illustrated in Figure 18.

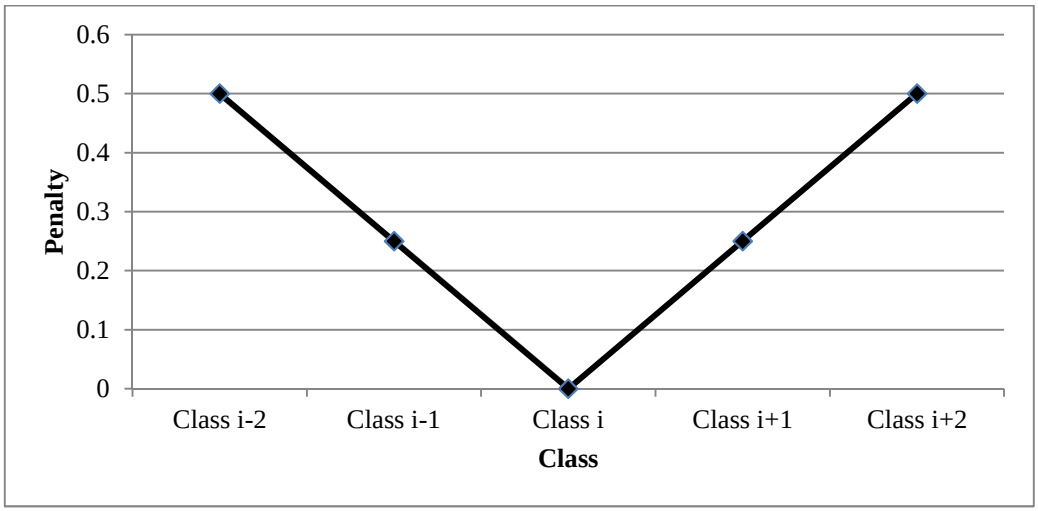


Figure 18. Penalty function for divergence from ordinality (Assuming Class *i* has the maximum posterior probability).

Measure 2. In order to take into account how much first and second most likely classes are away from each other, the Margin uncertainty measure is modified as follows:

$$x^* = \arg\min_x \frac{\left(P\left(y^*|x\right) - P\left(y'|x\right)\right)}{\left(j-i\right)^2} \tag{30}$$

Here, $P\left(y^*|x\right)$ is the maximum posterior probability and $P\left(y'|x\right)$ is the second maximum posterior probability predicted for a profile *x*. *j* is the numeric rank for the most likely class and *i* is the numeric rank for the second most likely class. Therefore, if first and second most likely classes are not adjacent for a given profile *x*, the probability that this profile is selected as the next query increases.

Measure 3. In order to take into account how much output distribution is in compliance with ordinality, we developed a measure as follows:

$$x^* = \arg\min_{x} \sum_{i \in Y / j} \frac{\left(P\left(y^*|x\right) - P\left(y_i|x\right)\right)}{\left(j-i\right)^2} \tag{31}$$

where Y is the rank set of classes. Here, $P\left(y^*|x\right)$ is the maximum posterior probability and $P(y_i \mid x)$ is the posterior probability predicted for class i. j is the numeric rank for the most likely class.

Measure 4. In QBB, contribution of each committee member to consensus output distribution is weighted with degree of compliance with ordinality. Hence, a committee member predicting posterior probabilities in compliance with ordinality contributes to the consensus output distribution more as shown in (32):

$$P_c\left(y|x\right) = \frac{1}{|C|} \sum_{\theta \epsilon C} \left(P_\theta\left(y|x\right) . \frac{1}{Pn_C\left(x\right)} \right) \tag{32}$$

Here, $P_C\left(y|x\right)$ is the consensus probability that y is the correct class according to the committee and $P_\theta\left(y|x\right)$ is the probability that y is the correct class according to committee member θ. $Pn_C\left(x\right)$ is calculated as in (29).

Measure 5. In QBB, contribution of each committee member to consensus output distribution is weighted with how much the committee member is confident about its prediction. Hence, a committee member being more confident about its prediction contributes to the consensus output distribution more as shown in (33).

$$P_c(y|x) = \frac{1}{|C|}\sum_{\theta \epsilon C}\left(P_\theta(y|x).P_\theta\left(y^*|x\right)\right) \quad (33)$$

Here, $P_\theta\left(y^*|x\right)$ is the maximum posterior probability predicted by the committee member θ. C is the committee set.

Measure 6. Each measure experimented (Least Confident, Margin and Entropy measures) measure different aspects of the output distribution. In order to make use of each of these measures simultaneously, we aggregated them into one measure in different combinations. Below are the aggregated measures experimented. Each measure is normalized before aggregating.

$$x^* = \arg\min_x\left(\varphi(x).\frac{1}{\Phi(x)}\right) \quad (34)$$

$$x^* = \arg\min_x\left(\beta(x).\frac{1}{\Phi(x)}\right) \quad (35)$$

$$x^* = \arg\min_x\left(\beta(x).\varphi(x)\right) \quad (36)$$

$$x^* = \arg\min_x\left(\beta(x).\varphi(x).\frac{1}{\Phi(x)}\right) \quad (37)$$

Here, $\varphi(x)$ corresponds to Least Confident measure, $\beta(x)$ corresponds to Margin measure and $\Phi(x)$ corresponds to Entropy measure.

None of the measures explained above achieved improvement over the ones proposed in the AL literature. This is due to the fact that, classifiers we use in the study are nominal classifiers and we make them exploit ordinal structure of the data with Frank and Hall (2001) approach. The main problem with this approach is that generated posterior probabilities do not add up to one. Uncertainty measures consider posterior probabilities for measuring uncertainty, therefore, they suffer from this property of Frank and Hall (2001). We expect and believe that, aforementioned measures would work successfully with learners developed particularly for ordinal classification. As a consequence, we used Least Confident, Margin and Entropy uncertainty measures in this study.

CHAPTER 4

ANALYSIS OF THE PROPOSED APPROACH

In this chapter, we analyze the proposed approach. First of all, we define an experimental design that considers various factors which are believed to have impact on the performance of the proposed algorithms. In order to determine which factors are statistically significant, we perform Analysis of Variance (ANOVA). Secondly, performance measure means for all test combinations are computed in order to designate which querying algorithm is the best for different circumstances. Thirdly, we look at learning curves of the proposed algorithms and compare them with the naive approach, where subsequent queries are performed randomly. Lastly, we provide algorithmic recommendations for modeling different underlying value functions in case we have information about the form of the preference structure of the DM.

4.1 Experimental Setup

The proposed approach does not assume an underlying functional form that represents preference structure of the DM and aims to model interactions among criteria as well as nonomonotonous preferential behavior. On the other hand, we also expect it to perform satisfactorily over various forms of preference structures including those assuming preference independence. Moreover, we expect a

preference learning algorithm in the ordinal classification setting to be robust to different circumstances such as number of classes, number of attributes or reference set at hand. Therefore, in order to evaluate and compare performances of the proposed algorithms, we design an experiment where algorithms are implemented and evaluated under different conditions. Detailed information regarding the experiment is provided below.

4.1.1 Design Factors

Experimental design investigates the performance of the algorithms on the basis of the factors shown in Table 5, which are believed to have impact on the performance.

Table 5. Experimental factors and their levels.

LEGEND	FACTOR	LEVELS
F1	Query Algorithm	1. Uncertainty Sampling 2. Query By Bagging
F2	Uncertainty Measure	1. Least Confident 2. Margin 3. Entropy/Vote Entropy
F3	Underlying Value Function	1. Linear 2. Multiplicative 3. Tchebycheff 4. Complex Nonmonotonic
F4	Number of classes	1. Two 2. Five
F5	Number of attributes	1. Three 2. Six
F6	Size of the reference set	1. 10 2. 30
F7	Number of queries	1. 30 2. 100
F8	Classifier	1. RF 2. SVM

We have two algorithms under consideration for factor F1. On the other hand, using 3 different uncertainty measures (for factor F2) with these algorithms, we can test 6 different querying algorithms in this experiment.

Underlying value functions listed in Table 5 for factor F3 are commonly used in MCDA practices and believed to be representing most of the DM preference structures. A complex and nonmonotonic preference structure showing switches in preference (high order interactions among attributes) and having irregular preference surface is represented with hinge functions.

Factor F4 corresponds to number of classes while Factor F5 represents the number of attributes. Even though most of the classification schemes consider only two-class (binary) classification, multiclass classification should be considered, as well. As the number of classes and attributes increase, complexity of the classification problem increases. Hence, we would like to see algorithms' performances as we increase the complexity of the problem.

Factor F6 is the size of the reference set. Small reference set contains little information about the preference structure of the DM. However, it is desirable that querying algorithm exploits information embedded in the reference set as much as possible so that next querying instance providing maximum information gain can be determined. One of the most important features of our approach is that we start with a small reference set. Hence, we expect candidate algorithms to perform well with small sized reference sets. As we mentioned in Chapter 3, our experiments show that when the number of training instances is less than 10, the probability that these learners fail increases. Therefore, we choose 10 and 30 as the two levels of factor F6 in the experiment.

Another factor (F7) we consider is the number of queries. With this factor, we would like to see if there exists a difference in the performance of algorithms with respect to the number of questions asked.

As the last factor (F8), we include two types of classifiers into our experiment. As we have pointed out previously, there is no "best" classification algorithm that outperforms others in all circumstances. Bearing that in mind, we desire to observe performance of different classifiers with the proposed algorithms. For that purpose, we consider two well-known classifiers in the SL literature, namely RF and SVM. From variety of learners, we prefer SVM and RF because they are among the strongest nonparametric classifiers proposed in the literature. Additionally, our experience showed that most of the nonparametric classifiers perform bad or fail with small data sets, while SVM and RF model small data sets very successfully.

4.1.2 Evaluation Measures

We consider three different evaluation measures (Accuracy, Balanced Class Accuracy and Kappa) in the experiment to measure performance of the approaches. We consider two more measures (Mean Absolute Error Ordinal, Mean Squared Error Ordinal) for 5 class cases. Each measure considers a different aspect of the algorithms. Evaluation measures used for comparing querying algorithms are summarized below.

- **Accuracy (Acc):** Accuracy can be defined as the number of correct predictions across all classes, L, divided by the number of observations (N, number of test cases) (Ferri et al., 2009). Acc can be formulated as follows:

$$Acc = \frac{\sum_{i=1}^{L} \#\ of\ examples\ of\ class\ i\ predicted\ right}{N} \tag{38}$$

- **Balanced Class Accuracy (BCA):** Balanced class accuracy is the average of accuracies calculated for all classes separately. This measure takes into account the number of instances from each class in the test data (in the case of class imbalances) (Su & Hsiao, 2009).

$$BCA = \frac{\sum_{i=1}^{L} \left(\frac{\#\ of\ examples\ of\ class\ i\ predicted\ right}{\#\ of\ examples\ in\ class\ i} \right)}{N} \tag{39}$$

- **Kappa coefficient (Cohen's Kappa):** The kappa coefficient is a measure of association used to describe and to test the degree of agreement in classification (Kotz et al., 2006). This coefficient is generally more robust than simple Acc since it takes into account the probability of chance agreement. Kappa coefficient can be computed for multiclass classification as follows (Kotz et al., 2006):

$$K = \frac{p_o - p_c}{1 - p_c}$$

$$p_0 = \sum_{ij} p_{ij} \tag{40}$$

$$p_c = \sum_{ij} p_{i.} p_{.j}$$

where p_o is the observed proportion of agreement, p_c is the proportion of agreement expected by chance, p_{ij} is the proportion of the N items

classified into category *i* by the first observer (true classes) and into *j* by the other (predicted classes), with

$$p_{i.} = \sum_{j} p_{ij}, \qquad p_{.j} = \sum_{i} p_{ij} \tag{41}$$

- **Mean Absolute Error Ordinal (MAEO):** MAEO measures both how much predicted classes are "incorrect" with respect to true classes, and how "inconsistent" the learner is with respect to relative order of the classes. MAEO can be formulated as follows (Cardoso & Sousa, 2011):

$$MAEO = \frac{1}{N} \sum_{r=1}^{L} \sum_{c=1}^{L} n_{r,c} \left| r - c \right| \tag{42}$$

where *r* corresponds to numeric rank of the true class and *c* represents numeric rank of the predicted class. $n_{r,c}$ represents the number of points from the *r*th class predicted as being from the *c*th class.

- **Mean Squared Error Ordinal (MSEO):** This measure is similar to MAEO. The only difference is that MSEO punishes inconsistency more than MAEO. MSEO can be formulated as follows (Cardoso & Sousa, 2011):

$$MSEO = \frac{1}{N} \sum_{r=1}^{L} \sum_{c=1}^{L} n_{r,c} \left(r - c \right)^2 \tag{43}$$

4.1.3 Underlying Value Functions

Value functions representing DM's underlying preference structure are summarized below. Besides, we give threshold values that are used to discretize value space so that preference ordered classes are generated. In the experiment, preference values of the alternatives are calculated based on the value functions given below. Then, they are classified according to these threshold values. For clarification, we are not assuming any functional form in the training phase whatsoever. These functions mimic DM when the DM answers algorithm's questions. Moreover, test cases are generated according to tested underlying value function.

- **Linear Value Function**
  - Three-attribute case

$$v(x_1, x_2, x_3) = 0.1x_1 + 0.4x_2 + 0.5x_3 \qquad 0 \le x_i \le 1 \quad \forall i \tag{44}$$

Table 6. Class threshold values for 2 class-3 attribute linear value function.

Threshold Values	Class
$v(.) < 0.5$	Class 1
$v(.) \ge 0.5$	Class 2

Table 7. Class threshold values for 5 class-3 attribute linear value function.

Threshold Values	Class
$v(.) < 0.35$	Class 1
$0.35 \le v(.) < 0.45$	Class 2
$0.45 \le v(.) < 0.55$	Class 3
$0.55 \le v(.) < 0.65$	Class 4
$v(.) \ge 0.65$	Class 5

- Six-attribute case

$$v(x_1,\ldots,x_6) = 0.1x_1 + 0.04x_2 + 0.06x_3 + 0.3x_4 + 0.2x_5 + 0.3x_6$$
$$0 \le x_i \le 1 \quad \forall i \tag{45}$$

Table 8. Class threshold values for 2 class-6 attribute linear value function.

Threshold Values	Class
$v(.) < 0.5$	Class 1
$v(.) \ge 0.5$	Class 2

Table 9. Class threshold values for 5 class-6 attribute linear value function.

Threshold Values	Class
$v(.) < 0.4$	Class 1
$0.4 \le v(.) < 0.45$	Class 2
$0.45 \le v(.) < 0.52$	Class 3
$0.52 \le v(.) < 0.57$	Class 4
$v(.) \ge 0.57$	Class 5

- **Multiplicative Value Function**

  - Three-attribute case

$$\begin{aligned} v(x_1, x_2, x_3) = {} & 0.5x_1 + 0.7x_2 + 0.2x_3 + 0.28x_1x_2 + \\ & 0.08x_1x_3 + 0.112x_2x_3 + 0.056x_1x_2x_3 \\ & 0 \le x_i \le 1 \quad \forall i \end{aligned} \tag{46}$$

Table 10. Class threshold values for 2 class-3 attribute multiplicative value function.

Threshold Values	Class
$v(.) < 0.8$	Class 1
$v(.) \ge 0.8$	Class 2

Table 11. Class threshold values for 5 class-3 attribute multiplicative value function.

Threshold Values	Class
$v(.) < 0.55$	Class 1
$0.55 \le v(.) < 0.65$	Class 2
$0.65 \le v(.) < 0.85$	Class 3
$0.85 \le v(.) < 0.95$	Class 4
$v(.) \ge 0.95$	Class 5

- Six-attribute case (3 and higher order interaction terms are omitted)

$$\begin{aligned} v(x_1,\ldots,x_6) = {} & 0.5x_1 + 0.7x_2 + 0.2x_3 + 0.1x_4 + 0.6x_5 + 0.3x_6 + 0.28x_1x_2 + \\ & 0.08x_1x_3 + 0.04x_1x_4 + 0.24x_1x_5 + 0.12x_1x_6 + 0.112x_2x_3 + \\ & 0.056x_2x_4 + 0.336x_2x_5 + 0.168x_2x_6 + 0.016x_3x_4 + \\ & 0.96x_3x_5 + 0.048x_3x_6 + 0.048x_4x_5 + 0.024x_4x_6 + 0.144x_5x_6 \\ & 0 \le x_i \le 1 \quad \forall i \end{aligned} \tag{47}$$

Table 12. Class threshold values for 2 class-6 attribute multiplicative value function.

Threshold Values	Class
$v(.) < 1.8$	Class 1
$v(.) \ge 1.8$	Class 2

Table 13. Class threshold values for 5 class-6 attribute multiplicative value function.

Threshold Values	Class
$v(.) < 1.2$	Class 1
$1.2 \le v(.) < 1.5$	Class 2
$1.5 \le v(.) < 1.8$	Class 3
$1.8 \le v(.) < 2.5$	Class 4
$v(.) \ge 2.5$	Class 5

- **Weighted Tchebycheff Value Function**

  - Three-attribute case

$$v(x_1, x_2, x_3) = max\left(0.3(1.0 - x_1), 0.4(1.0 - x_2), 0.3(1.0 - x_3)\right) \quad (48)$$
$$0 \le x_i \le 1 \quad \forall i$$

Table 14. Class threshold values for 2 class-3 attribute weighted Tchebycheff value function.

Threshold Values	Class
$v(.) \ge 0.25$	Class 1
$v(.) < 0.25$	Class 2

Table 15. Class threshold values for 5 class-3 attribute weighted Tchebycheff value function.

Threshold Values	Class
$v(.) \geq 0.3$	Class 1
$0.25 \leq v(.) < 0.3$	Class 2
$0.17 \leq v(.) < 0.25$	Class 3
$0.11 \leq v(.) < 0.17$	Class 4
$v(.) < 0.11$	Class 5

- Six-attribute case

$$v(x_1,\ldots,x_6) = max\begin{pmatrix} 0.1(1.0-x_1), 0.06(1.0-x_2), 0.04(1.0-x_3), \\ 0.4(1.0-x_4),\ 0.3(1.0-x_5), 0.1(1.0-x_6) \end{pmatrix} \qquad (49)$$
$$0 \leq x_i \leq 1 \quad \forall i$$

Table 16. Class threshold values for 2 class-6 attribute weighted Tchebycheff value function.

Threshold Values	Class
$v(.) \geq 0.25$	Class 1
$v(.) < 0.25$	Class 2

Table 17. Class threshold values for 5 class-6 attribute weighted Tchebycheff value function.

Threshold Values	Class
$v(.) \geq 0.3$	Class 1
$0.25 \leq v(.) < 0.3$	Class 2
$0.2 \leq v(.) < 0.25$	Class 3
$0.15 \leq v(.) < 0.2$	Class 4
$v(.) < 0.15$	Class 5

- **Complex Nonmonotonic Value Function**

  - Three-attribute case

$$\begin{aligned} v(x_1, x_2, x_3) = {} & 0.8 + 0.2h(0.5 - x_1) - 0.5h(x_2 - 0.2) + \\ & 0.3h(x_3 - 0.7) - 0.6h(0.6 - x_2)h(x_3 - 0.4) + \\ & 0.3h(0.7 - x_1)h(0.4 - x_2)h(0.2 - x_3) \\ & 0 \leq x_i \leq 1 \quad \forall i \end{aligned} \tag{50}$$

Table 18. Class threshold values for 2 class-3 attribute complex nonmonotonic value function.

Threshold Values	Class
$v(.) < 0.7$	Class 1
$v(.) \geq 0.7$	Class 2

Table 19. Class threshold values for 5 class-3 attribute complex nonmonotonic value function.

Threshold Values	**Class**
$v(.) < 0.51$	Class 1
$0.51 \le v(.) < 0.6$	Class 2
$0.6 \le v(.) < 0.7$	Class 3
$0.7 \le v(.) < 0.8$	Class 4
$v(.) \ge 0.8$	Class 5

- Six-attribute case

$$\begin{aligned} v(x_1,\ldots,x_6) = {} & 0.8 + 0.2h(0.5 - x_1) - 0.5h(x_2 - 0.2) + \\ & 0.3h(x_3 - 0.7) - 0.25h(x_5 - 0.4) + \\ & 0.18h(x_6 - 0.64) - 0.6h(0.6 - x_2)h(x_3 - 0.4) + \\ & 0.8h(0.7 - x_1)h(0.4 - x_2)h(0.2 - x_3) - \\ & 0.35h(x_4 - 0.46)h(0.8 - x_6) \\ & 0 \le x_i \le 1 \quad \forall i \end{aligned} \tag{51}$$

Table 20. Class threshold values for 2 class-6 attribute complex nonmonotonic value function.

Threshold Values	**Class**
$v(.) < 0.6$	Class 1
$v(.) \geq 0.6$	Class 2

Table 21. Class threshold values for 5 class-6 attribute complex nonmonotonic value function.

Threshold Values	**Class**
$v(.) < 0.5$	Class 1
$0.5 \leq v(.) < 0.6$	Class 2
$0.6 \leq v(.) < 0.65$	Class 3
$0.65 \leq v(.) < 0.75$	Class 4
$v(.) \geq 0.75$	Class 5

In the complex nonmonotonic value function, $h(.)$ corresponds to hinge function that takes the form $max(0, x-c)$ or $max(0, c-x)$ where c is a constant. This constant is called a knot. Hinge functions create piecewise linear structures and capable of representing complex functional forms. The main difference from other piecewise linear functions is that they can be multiplied together to form nonlinear functions.

4.2 Analysis of the Experimental Results

In order to analyze the experiment, we define a general full factorial design. With our factorial setting, this design has 768 combinations. Each combination is replicated 20 times. Hence, we run the experiment 15,360 (768 $\times$ 20) times in order to obtain required performance measures. Every time a run is started, a new reference set is generated randomly. Assuming that we have *n* criteria under consideration, criteria values of each candidate instance in the reference set are determined such that:

$$x_i \sim U(x_i^{\min}, x_i^{\max}) \quad \text{for} \quad i = 1, ..., n \tag{52}$$

where x_i is the random value attained in criterion *i*, $x_i^{\min}$ and $x_i^{\max}$ are the minimum and the maximum criterion values attainable in criterion *i* respectively, and $U(.)$ is the uniform distribution. Afterwards, class labels of these instances are determined based on the underlying value function and class threshold values considered for that particular test combination. Hence, underlying value function mimics DM in answering questions. In order to include instances from each preference class in the reference set, we ensure that there are at least $\left\lfloor \frac{|reference\ set|}{number\ of\ classes} \right\rfloor$ examples from each class. This is also a requirement for classifiers in order not to fail in training at the initial phases of the algorithm since we start with small sized reference sets.

At each iteration of the algorithm (each time a classifier is trained), test set (validation set) of size 1,000 and pool set of size 500 are randomly generated in accordance with the procedure explained in Chapter 3. Test set is used to measure performance of the classifier at each iteration. In other words, classifier trained with the current reference set (instances in the initial reference set and questions

asked thus far) makes predictions for the instances in the test set and performance of the classifier is measured in terms of the evaluation measured considered.

Algorithms are implemented in the R v3.1.2 statistical programming language (R Core Team, 2014). We implemented SVM with the R package e1071 (Meyer et al., 2014) and RF with the R package caret (Max et al., 2015). Results of the experiment are analyzed using Minitab 16 (Minitab 16 Statistical Software, 2010).

While performing the ANOVA, we have checked adequacy of the ANOVA models in terms of;

- Normality (of the residuals)
- Homoscedasticity (constant variance)
- Auto-correlation (independence of residuals)

In our preliminary analysis, we have observed that residuals suffered from nonnormality and nonconstant variance. Therefore, we have decided to apply transformations to the measures under consideration. Since Acc, BCA and Kappa are proportions, we have applied $arcsin\sqrt{perfMetric}$ transformation to these measures. This transformation stabilizes the variance and makes distribution of the residuals close to normal (Montgomery, 2009). Box-Cox transformations have been applied to MAEO and MSEO, for the same reason. After transformations, we have verified that models generated met assumptions of the ANOVA.

Table 22 shows eight-way ANOVA results for the Acc measure. All main effects and interaction effects presented in this table are significant. Other than 17 effects shown in the table, there are 45 other interaction effects that are statistically significant (at significance level 0.01), as well. In order to determine how much each effect contributes to the total variance, ω^2 statistic is computed (Tabachnick & Fidell, 2007), as done in (Doumpos & Zopounidis, 2002, p.135). Each effect

presented in this table explains at least 0,1% of the total variance except for effects **Query Algorithm** and **Uncertainty Measure**. Being main effects, they are included in the table.

As stated previously, all main effects are significant. This result shows that all factors included in the design have significant impact on the average classification accuracy. Factors **Number of Classes** and **Number of Attributes** are the most influential effects. Along with **Underlying Value Function**, these factors determine complexity of the classification task. **Number of Classes*Number of Attributes** interaction is the 7th influential effect and **Underlying Value Function*Number of Classes** interaction is the 9th influential effect in the experiment. These interactions reveal that effects of **Underlying Value Function** and **Number of Attributes** vary depending on the number of classes we have in the problem. **Number of Queries** is the 5th influential effect. This is to be expected since as we continue asking questions, we obtain more information about the preference structure of the DM and consequently classifier performance increases. **Underlying Value Function* Classifier** interaction is the most influential interaction effect in the table. Additionally, **Classifier** is the 6th influential effect. This reveals that performances of the algorithms differ depending on the classifier. Moreover, we can conclude that underlying value function has impact on the classifier performance. Since **Querying Algorithm** and **Uncertainty Measure** are statistically significant, we can state that there is statistically significant difference among different querying algorithms (and uncertainty measures).

Table 22. ANOVA results for the Acc measure.

Factor	*df*	*Seq SS*	*Adj SS*	*Adj MS*	*F*	*p*	ω^2
F4	1	619.010	619.010	619.010	175740.280	0.000	28.17%
F5	1	185.865	185.865	185.865	52768.100	0.000	10.53%
F3*F8	3	289.144	289.144	96.381	27363.220	0.000	5.75%
F3	3	220.579	220.579	73.526	20874.480	0.000	4.45%
F7	1	49.822	49.822	49.822	14144.710	0.000	3.06%
F8	1	20.497	20.497	20.497	5819.300	0.000	1.28%
F4*F5	1	17.710	17.710	17.710	5027.870	0.000	1.11%
F5*F8	1	15.582	15.582	15.582	4423.700	0.000	0.98%
F3*F4	3	35.058	35.058	11.686	3317.740	0.000	0.73%
F3*F4*F8	3	18.975	18.975	6.325	1795.720	0.000	0.40%
F4*F8	1	5.132	5.132	5.132	1457.080	0.000	0.32%
F6	1	4.558	4.558	4.558	1294.110	0.000	0.29%
F3*F5	3	10.220	10.220	3.407	967.210	0.000	0.22%
F7*F8	1	2.728	2.728	2.728	774.500	0.000	0.17%
F3*F5*F8	3	7.073	7.073	2.358	669.380	0.000	0.15%
F2	2	1.089	1.089	0.544	154.550	0.000	0.03%
F1	1	0.134	0.134	0.134	38.060	0.000	0.01%

Table 23 shows eight-way ANOVA results for the BCA measure. This table presents similar results to those of Acc measure. All main and interaction effects shown in Table 23 are statistically significant, therefore, all factors included in the design have impact on BCA. Other than 18 effects shown in this table, there are 45 other interaction effects that are statistically significant (at significance level 0.01), as well. Each effect presented in this table explains at least 0.1 % of the total variance except for effects F1 and F2.

Number of Classes and **Number of Attributes** are the most influential effects, as in the ANOVA table for the Acc. These two factors explain 38.93 % of the total variance. The most influential interaction effect is the **Underlying Value Function*Classifier** interaction. This reveals that underlying value function of the

DM has impact on the classifier performance according to BCA measure. This interaction effect explains 5.36% of the total variance.

Table 23. ANOVA results for the BCA measure.

Factor	*df*	*Seq SS*	*Adj SS*	*Adj MS*	*F*	*p*	ω^2
F4	1	720.797	720.797	720.797	198684.580	0.000	29.35%
F5	1	183.739	183.739	183.739	50646.820	0.000	9.58%
F3*F8	3	294.910	294.910	98.303	27096.910	0.000	5.36%
F3	3	249.192	249.192	83.064	22896.230	0.000	4.57%
F7	1	52.549	52.549	52.549	14484.810	0.000	2.94%
F8	1	19.288	19.288	19.288	5316.610	0.000	1.10%
F4*F5	1	16.814	16.814	16.814	4634.570	0.000	0.96%
F5*F8	1	16.216	16.216	16.216	4469.950	0.000	0.93%
F3*F4	3	45.783	45.783	15.261	4206.620	0.000	0.87%
F3*F4*F8	3	21.143	21.143	7.048	1942.650	0.000	0.40%
F4*F8	1	6.251	6.251	6.251	1722.940	0.000	0.36%
F6	1	4.566	4.566	4.566	1258.610	0.000	0.26%
F3*F5	3	11.163	11.163	3.721	1025.660	0.000	0.21%
F7*F8	1	2.888	2.888	2.888	796.120	0.000	0.17%
F3*F5*F8	3	7.518	7.518	2.506	690.770	0.000	0.14%
F4*F7	1	1.787	1.787	1.787	492.610	0.000	0.10%
F2	2	0.615	0.615	0.308	84.760	0.000	0.02%
F1	1	0.137	0.137	0.137	37.790	0.000	0.01%

Table 24 shows eight-way ANOVA results for the Kappa measure. This table presents similar results to those of Acc and BCA measures. All main effects shown in Table 24 are significant, thereby, all factors included in the design have impact on Kappa. Other than 18 effects shown in this table, there are 45 other interaction effects that are statistically significant, as well. Each effect presented in this table explains at least 0.1 % of the total variance except for effects **Query Algorithm** and **Uncertainty Measure**. As in the ANOVA table for BCA, main effects

Underlying Value Function, **Number of Classes**, **Number of Attributes**, **Number of Queries**, **Classifier** and interaction effect **Underlying Value Function*Classifier** are the most influential effects and explain 51.57% of the total variance.

Table 24. ANOVA results for the Kappa measure.

Factor	*df*	*Seq SS*	*Adj SS*	*Adj MS*	*F*	*p*	ω^2
F4	1	489.779	489.779	489.779	76763.570	0.000	20.01%
F5	1	292.571	292.571	292.571	45855.060	0.000	13.00%
F3*F8	3	450.692	450.692	150.231	23545.830	0.000	7.12%
F3	3	342.160	342.160	114.053	17875.720	0.000	5.50%
F7	1	82.168	82.168	82.168	12878.240	0.000	4.03%
F8	1	38.144	38.144	38.144	5978.290	0.000	1.91%
F5*F8	1	22.641	22.641	22.641	3548.510	0.000	1.14%
F4*F8	1	13.555	13.555	13.555	2124.460	0.000	0.69%
F4*F5	1	12.041	12.041	12.041	1887.130	0.000	0.61%
F3*F4	3	32.941	32.941	10.980	1720.980	0.000	0.56%
F6	1	7.578	7.578	7.578	1187.660	0.000	0.39%
F3*F5	3	18.099	18.099	6.033	945.570	0.000	0.31%
F7*F8	1	4.714	4.714	4.714	738.910	0.000	0.24%
F3*F5*F8	3	11.191	11.191	3.730	584.670	0.000	0.19%
F3*F4*F8	3	10.068	10.068	3.356	525.970	0.000	0.17%
F1*F4*F5	1	2.537	2.537	2.537	397.630	0.000	0.13%
F2	2	1.519	1.519	0.759	119.020	0.000	0.04%
F1	1	0.095	0.095	0.095	14.900e	0.000	0.00%

To sum up, three ANOVA tables presented reveal similar results. Similarity among results provided by different measures stems from the fact that we start with a small initial reference set that is formed with stratified random sampling. Therefore, initial set is somewhat balanced. As querying process continues, balance among classes rarely changes. In general, same factors explain variance in

all three performance measures. All main effects are statistically significant and have impact on the performance measures under consideration. Factors **Query Algorithm** and **Uncertainty Measure** are statistically significant. Hence, we can state that there is a statistically significant difference among different querying algorithms (and uncertainty measures) when these three measures are considered. Factors **Underlying Value Function**, **Number of Classes**, **Number of Attributes**, **Number of Queries** and **Classifier** are the most influential main effects among all factors. Factors **Underlying Value Function**, **Number of Classes** and **Number of Attributes** designate the complexity of the classification task. Factor **Classifier** is significant, as well, indicating that there is significant difference between performances of SVM and RF under different experimental conditions. Most influential interaction effect in those three ANOVA tables is the **Underlying Value Function*Classifier** effect. As stated previously, this interaction reveals that underlying value function has impact on the classifier performance. Three way interaction **Underlying Value Function*Number of Classes*Classifier** is the most influential three-way interaction effect for Acc and BCA. Significance of this interaction asserts that underlying value function, number of classes in the problem and type of classifier jointly have a significant impact on the classification performance. In other words, if we have a clue about form of the underlying value function of the DM, we can improve prediction performance by choosing the proper classifier.

Apart from these three performance measures, we also consider MAEO and MSEO for 5-class cases. As stated previously, these measures measure not only accuracy of a learner but also how "inconsistent" the learner is with respect to relative order of the classes. In this context, Table 25 shows seven-way ANOVA results for the MAEO performance measure. Since these measures are considered only for multiclass cases, factor **Number of Classes** is not included in the analysis.

All main effects and interaction effects presented in Table 25 are significant, therefore, all factors included in the design have impact on MAEO. Other than 14 effects shown in this table, there are 22 other interaction effects that are statistically significant (at significance level 0.01), as well. Except for **Query Algorithm**, each effect presented in this table explains at least 0,1% of the total variance. Being a main effect, **Query Algorithm** is included in this table, too.

Table 25. ANOVA results for the MAEO measure.

Factor	*df*	*Seq SS*	*Adj SS*	*Adj MS*	*F*	*p*	ω^2
F5	1	92.997	92.997	92.997	69847.190	0.000	18.90%
F3*F8	3	124.776	124.776	41.592	31238.660	0.000	9.44%
F3	3	121.676	121.676	40.559	30462.510	0.000	9.23%
F7	1	22.629	22.629	22.629	16996.090	0.000	5.37%
F5*F8	1	5.970	5.970	5.970	4484.010	0.000	1.47%
F6	1	1.943	1.943	1.943	1459.030	0.000	0.48%
F3*F5	3	5.650	5.650	1.883	1414.470	0.000	0.47%
F3*F5*F8	3	4.568	4.568	1.523	1143.700	0.000	0.38%
F7*F8	1	0.772	0.772	0.772	579.860	0.000	0.19%
F8	1	0.716	0.716	0.716	537.610	0.000	0.18%
F2*F8	2	1.194	1.194	0.597	448.370	0.000	0.15%
F3*F7	3	1.209	1.209	0.403	302.620	0.000	0.10%
F2	2	0.804	0.804	0.402	301.850	0.000	0.10%
F1	1	0.264	0.264	0.264	198.070	0.000	0.07%

Factors **Underlying Value Function** and **Number of Attributes** are the most influential effects. These factors designate complexity of the classification task. **Number of Queries** is the 3rd most influential factor since as we continue asking questions, we obtain more information about the preference structure of the DM and consequently classifier performance increases. These three factors explain 33.49 % of the total variance. The most influential interaction effect is the

Underlying Value Function*Classifier interaction. This interaction reveals that underlying value function has impact on the classifier performance. Factors **Query Algorithm** and **Uncertainty Measure** are statistically significant and among the most influential effects. This shows that, type of query algorithm and uncertainty measure used have impact on the performance of the learner in terms of MAEO.

Table 26 shows seven-way ANOVA results for MSEO. Similar to our findings regarding MAEO, all main effects and interaction effects presented in Table 26 are significant. Other than 16 effects shown in this table, there are 17 other interaction effects that are statistically significant (at significance level 0.01), as well. Except for **Query Algorithm**, each effect presented in this table explains at least 0,1% of the total variance. Being a main effect, **Query Algorithm** is included in this table, too.

Table 26. ANOVA results for the MSEO measure.

Factor	*df*	*Seq SS*	*Adj SS*	*Adj MS*	*F*	*p*	ω^2
F5	1	358.837	358.837	358.837	18877.670	0.000	13.75%
F3*F8	3	733.834	733.834	244.611	12868.490	0.000	9.80%
F3	3	698.160	698.160	232.720	12242.930	0.000	9.37%
F7	1	122.558	122.558	122.558	6447.510	0.000	5.16%
F5*F8	1	27.192	27.192	27.192	1430.500	0.000	1.19%
F6	1	16.465	16.465	16.465	866.210	0.000	0.73%
F2	2	21.083	21.083	10.541	554.560	0.000	0.47%
F3*F7*F8	3	25.357	25.357	8.452	444.670	0.000	0.37%
F3*F7	3	24.721	24.721	8.240	433.510	0.000	0.36%
F8	1	6.895	6.895	6.895	362.730	0.000	0.31%
F3*F5	3	17.537	17.537	5.846	307.530	0.000	0.26%
F3*F5*F8	3	13.406	13.406	4.469	235.090	0.000	0.20%
F2*F8	2	7.874	7.874	3.937	207.120	0.000	0.17%
F7*F8	1	3.757	3.757	3.757	197.660	0.000	0.17%
F6*F7	1	3.184	3.184	3.184	167.500	0.000	0.14%
F1	1	1.718	1.718	1.718	90.370	0.000	0.08%

Similar to our findings regarding MAEO, factors **Underlying Value Function**, **Number of Attributes** and **Number of Queries** are the most influential effects. These three factors explain 28.28 % of the total variance. Factors **Query Algorithm** and **Uncertainty Measure** are significant, hence, type of query algorithm and uncertainty measure used have impact on the performance of the learner in terms of MSEO.

As the next step of our analysis, we examine the performance of query algorithms across all factors. We also include results of another querying strategy, the naive approach, where the next question is asked randomly. This naive strategy provides us a benchmark to assess performance of our querying algorithms. Tables A.1-A.10 in Appendix A show mean performance measure values of 20 replications across all design factors. Algorithm having the best performance measure value for the given test combination is indicated with a bold font. In all of the 64 different experimental conditions, at least one of the proposed approaches outperforms the random approach in terms of the performance measures.

In order to observe how querying algorithms perform with different reference sizes and number of queries, we compute mean performance measures for different reference and query sizes which are presented in Tables B.1-B10 in Appendix B. Analysis results verify our prior conclusion asserting that there exist statistically significant differences between performances of the algorithms that start with different reference sizes. Performance figures improve (Acc, BCA and Kappa increase while MAEO and MSEO decrease) as size of the reference set increases. Similar result holds for the number of questions, as well. Acc, BCA and Kappa increase (and MAEO and MSEO decrease) as we ask more questions, as expected.

In order to designate which querying algorithm is the best for different underlying value functions, performance measure means for all test combinations are computed. According to results presented in the following tables, the US algorithm

outperforms the QBB algorithm in all cases. Therefore, we can conclude that, the US algorithm is the best choice for modeling preference of the DM in the ordinal classification setting, regardless of the functional form of the preference structure. According to the following tables, SVM outperforms RF in modeling preference structures compatible with linear and multiplicative value functions. Conversely, RF performs better when the underlying value function is Tchebycheff or Complex Nonmonotonic. This result makes sense because tree based learners are very capable of modeling nonlinear or complex relationships while they may fail if the relationship between independent and dependent variables is close to linear. As James et al. (2013) remark, a model that is capable of modeling linear relationships is likely to outperform tree-based methods in the linear case. The linear value function is completely linear while the multiplicative value function is close to linear. This is the reason why SVM, which is also successful in modeling linear relationships, performs better.

Table 27. Acc means of the algorithms for different underlying value functions.

Underlying Value Function	**Uncertainty Sampling**						**Query By Bagging**					
	LC		**M**		**E**		**LC**		**M**		**E**	
	RF	**SVM**	**RF**	**SVM**	**RF**	**SVM**	**RF**	**SVM**	**RF**	**SVM**	**RF**	**SVM**
Linear	0.7265	**0.8101**	0.7586	0.8089	0.7161	0.8051	0.7170	0.8009	0.7546	0.8004	0.7095	0.7979
Multiplicative	0.7239	**0.8279**	0.7545	0.8261	0.7160	0.8276	0.7206	0.8042	0.7503	0.8059	0.7124	0.8090
Tchebycheff	**0.9920**	0.7862	0.9907	0.7848	0.9895	0.7850	0.9889	0.7942	0.9879	0.7939	0.9855	0.7907
Complex Nonmonotonic	0.7826	0.7820	**0.8094**	0.7807	0.7782	0.7811	0.7771	0.7932	0.8069	0.7934	0.7740	0.7967
Average	0.8062	0.8016	**0.8283**	0.8001	0.8000	0.7997	0.8009	0.7981	0.8249	0.7984	0.7954	0.7986

Table 28. BCA means of the algorithms for different underlying value functions.

Underlying Value Function	Uncertainty Sampling						Query By Bagging					
	LC		M		E		LC		M		E	
	RF	SVM	RF	SVM	RF	SVM	RF	SVM	RF	SVM	RF	SVM
Linear	0.7168	**0.7983**	0.7427	0.7973	0.7062	0.7937	0.7091	0.7899	0.7389	0.7883	0.7019	0.7864
Multiplicative	0.6964	0.8116	0.7239	0.8044	0.6887	**0.8174**	0.6912	0.7893	0.7192	0.7847	0.6830	0.7915
Tchebycheff	**0.9920**	0.7841	0.9910	0.7791	0.9882	0.7824	0.9875	0.7916	0.9865	0.7881	0.9833	0.7880
Complex Nonmonotonic	0.7661	0.7698	**0.7950**	0.7648	0.7642	0.7701	0.7610	0.7825	0.7913	0.7803	0.7585	0.7870
Average	0.7928	0.7910	**0.8131**	0.7864	0.7868	0.7909	0.7872	0.7883	0.8090	0.7854	0.7817	0.7882

Table 29. Kappa means of the algorithms for different underlying value functions.

Underlying Value Function	Uncertainty Sampling						Query By Bagging					
	LC		M		E		LC		M		E	
	RF	SVM	RF	SVM	RF	SVM	RF	SVM	RF	SVM	RF	SVM
Linear	0.6222	**0.7323**	0.6621	0.7307	0.6084	0.7254	0.6111	0.7148	0.6559	0.7146	0.6013	0.7120
Multiplicative	0.6080	0.7440	0.6455	0.7411	0.5983	**0.7442**	0.6015	0.7131	0.6394	0.7164	0.5911	0.7194
Tchebycheff	**0.9897**	0.6838	0.9881	0.6826	0.9866	0.6809	0.9857	0.7017	0.9845	0.7028	0.9814	0.6965
Complex Nonmonotonic	0.6932	0.6734	**0.7299**	0.6716	0.6885	0.6720	0.6849	0.6995	0.7250	0.7012	0.6810	0.7051
Average	0.7283	0.7084	**0.7564**	0.7065	0.7204	0.7056	0.7208	0.7073	0.7512	0.7087	0.7137	0.7082

Table 30. MAEO means of the algorithms for different underlying value functions.

Underlying Value Function	Uncertainty Sampling						Query By Bagging					
	LC		M		E		LC		M		E	
	RF	SVM	RF	SVM	RF	SVM	RF	SVM	RF	SVM	RF	SVM
Linear	0.5880	0.3555	0.4879	**0.3489**	0.6243	0.3672	0.6036	0.3581	0.4923	0.3534	0.6262	0.3741
Multiplicative	0.5414	**0.2907**	0.4650	0.2976	0.5682	0.2924	0.5462	0.3253	0.4716	0.3284	0.5697	0.3168
Tchebycheff	**0.0160**	0.3300	0.0186	0.3391	0.0210	0.3299	0.0216	0.3390	0.0239	0.3498	0.0287	0.3489
Complex Nonmonotonic	0.3966	**0.3438**	0.3476	0.3533	0.4100	0.3465	0.4030	0.3550	0.3483	0.3581	0.4102	0.3515
Average	0.3855	0.3300	**0.3298**	0.3347	0.4059	0.3340	0.3936	0.3443	0.3340	0.3474	0.4087	0.3479

Table 31. MSEO means of the algorithms for different underlying value functions.

Underlying Value Function	Uncertainty Sampling						Query By Bagging					
	LC		M		E		LC		M		E	
	RF	SVM	RF	SVM	RF	SVM	RF	SVM	RF	SVM	RF	SVM
Linear	58.8373	26.6222	29.2925	**17.2059**	63.6564	28.5184	57.0389	26.2365	29.2080	17.4359	63.4934	30.1443
Multiplicative	42.4039	17.6993	25.9324	**14.7246**	47.5618	16.8677	44.8173	18.6413	26.6115	16.1575	47.3459	17.4296
Tchebycheff	**0.4271**	21.4122	0.5939	17.4520	0.5389	20.5120	0.5530	21.9037	0.5332	18.8721	0.9373	23.9323
Complex Nonmonotonic	27.7527	21.6627	**17.1170**	17.4464	28.9136	21.3312	29.2664	23.6311	17.3112	19.0478	30.1247	21.7183
Average	32.3552	21.8491	18.2339	**16.7072**	35.1677	21.8074	32.9189	22.6032	18.4160	17.8783	35.4753	23.3061

These tables guide us regarding which algorithm and uncertainty measure to use if we have information about functional form of the preference structure. We have already designated that US is the best choice under all circumstances. When the underlying value function is linear, Acc, BCA and Kappa measures recommend Least Confident uncertainty measure while MAEO and MSEO propose the Margin measure. In the Multiplicative case, Least Confident is the best choice according to Acc and MAEO. All measures recommend using Least Confident when the underlying value function is Tchebycheff. When we are faced with a preference structure showing complex structure having interacting criteria, majority of the performance measures propose the Margin measure.

As stated previously, we usually have no idea about functional form of the preference structure of the DM unless we have previous information or performed further diagnostic analysis. In this respect, in order to provide a global evaluation regarding performances of all querying algorithms under various classification conditions, overall means of 64 combinations are shown in Table 32.

Table 32. Overall performance measure means of querying algorithms.

Performance Measure	Uncertainty Sampling						Query By Bagging					
	LC		M		E		LC		M		E	
	RF	SVM	RF	SVM	RF	SVM	RF	SVM	RF	SVM	RF	SVM
Acc	0.8062	0.8016	**0.8283**	0.8001	0.8000	0.7997	0.8009	0.7981	0.8249	0.7984	0.7954	0.7986
BCA	0.7928	0.7910	**0.8131**	0.7864	0.7868	0.7909	0.7872	0.7883	0.8090	0.7854	0.7817	0.7882
Kappa	0.7283	0.7084	**0.7564**	0.7065	0.7204	0.7056	0.7208	0.7073	0.7512	0.7087	0.7137	0.7082
MAEO	0.3855	0.3300	**0.3298**	0.3347	0.4059	0.3340	0.3936	0.3443	0.3340	0.3474	0.4087	0.3479
MSEO	32.3552	21.8491	18.2339	**16.7072**	35.1677	21.8074	32.9189	22.6032	18.4160	17.8783	35.4753	23.3061

In harmony with our previous findings, US algorithm outperforms QBB in terms of all performance measures. Regarding the choice of an uncertainty measure, Margin prevails. Consequently, if we have no clue about functional form of the preference structure of the DM, which is the case we are usually faced with, the US algorithm with the Margin uncertainty measure would be the best choice. Table 32 recommends using RF as the base classifier in this case. However, as we have pointed out previously, if we have evidence that preference structure is somewhat linear, it would be better to utilize SVM.

Since our primary motivation for using nonparametric classification techniques with AL is to model complex preference structures where criteria interact with each other and show somewhat nonmonotonic behavior, we will take a closer look at performances of querying algorithms when underlying preference structure is complex nonmonotonic.

In ML practices, performance of a learner/algorithm is usually displayed or compared with another learner/algorithm by using learning curves. Concisely, a learning curve shows cross-validation/training error or accuracy as a function of the number of training instances. In our research we aim using querying resources

parsimoniously, hence we query new instances one by one interactively. In this respect, we use learning curves to illustrate performances of querying algorithms when the underlying value function has the complex nonmonotonic forms shown in Subsection 4.1.3. Figure 19-Figure 30 show learning curves of Acc and MSEO for different reference set sizes and number of attributes/classes. In these figures, learning curves for the best RF and SVM based algorithms as well as random approaches where RF and SVM used as base learners are presented.

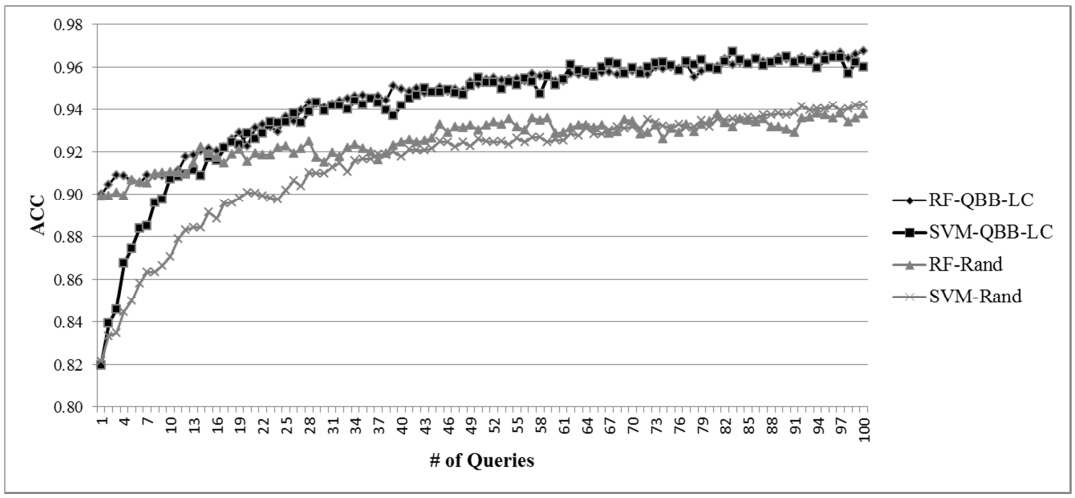


Figure 19. Learning curve of Acc (Reference set size 10. Value function has 3 attributes and 2 classes.)

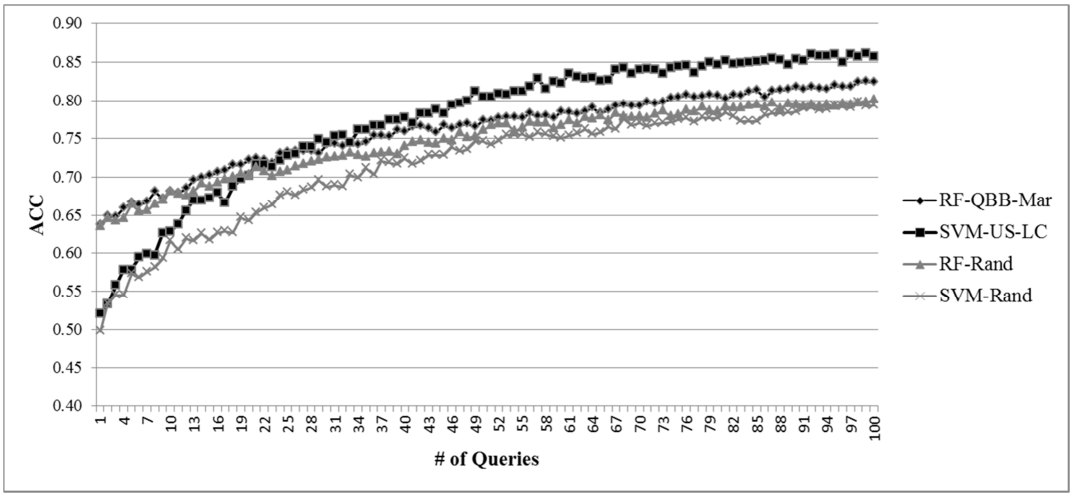


Figure 20. Learning curve of Acc (Reference set size 10. Value function has 3 attributes and 5 classes.)

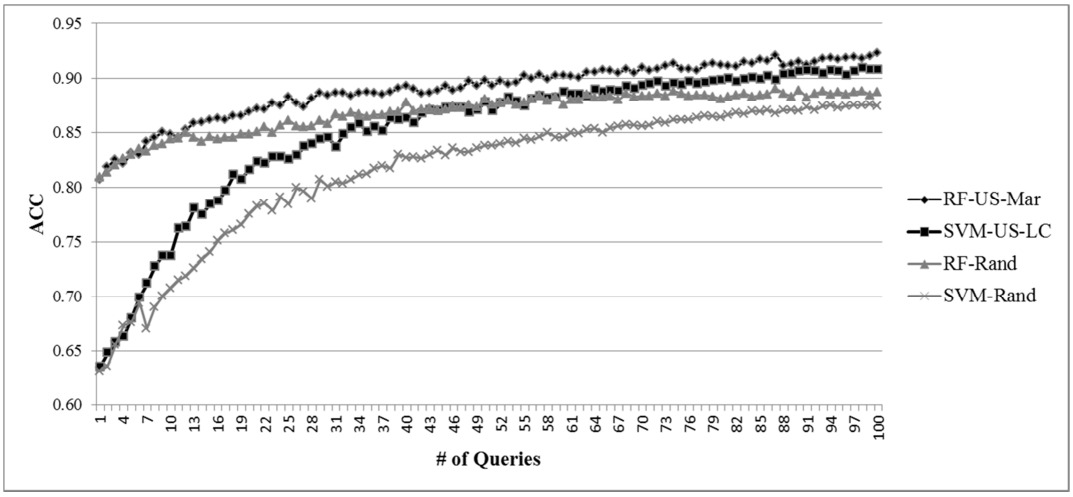


Figure 21. Learning curve of Acc (Reference set size 10. Value function has 6 attributes and 2 classes.)

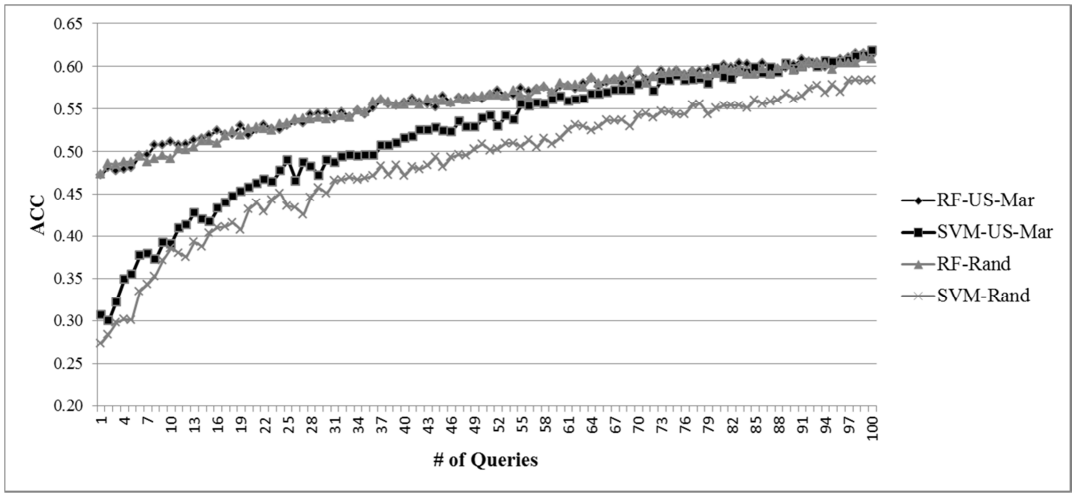


Figure 22. Learning curve of Acc (Reference set size 10. Value function has 6 attributes and 5 classes.)

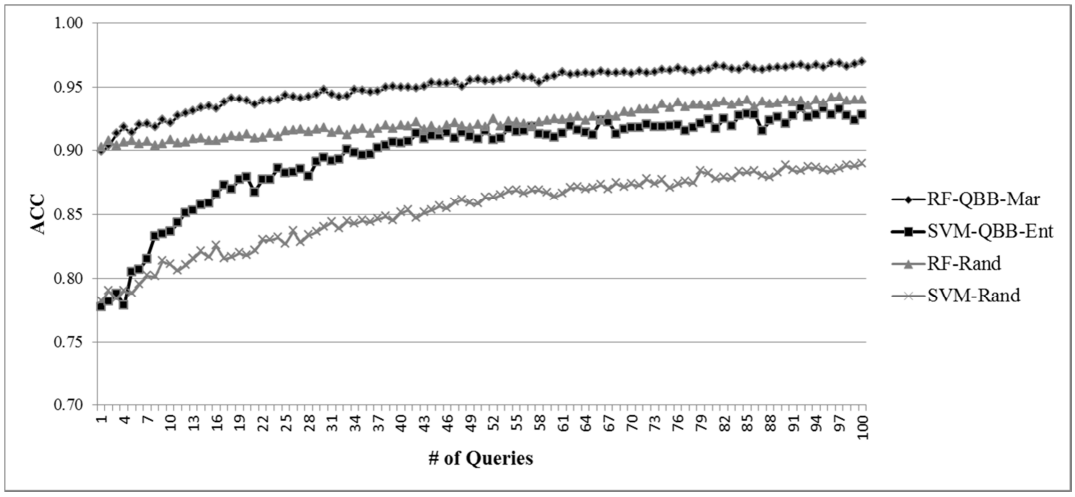


Figure 23. Learning curve of Acc (Reference set size 30. Value function has 3 attributes and 2 classes.)

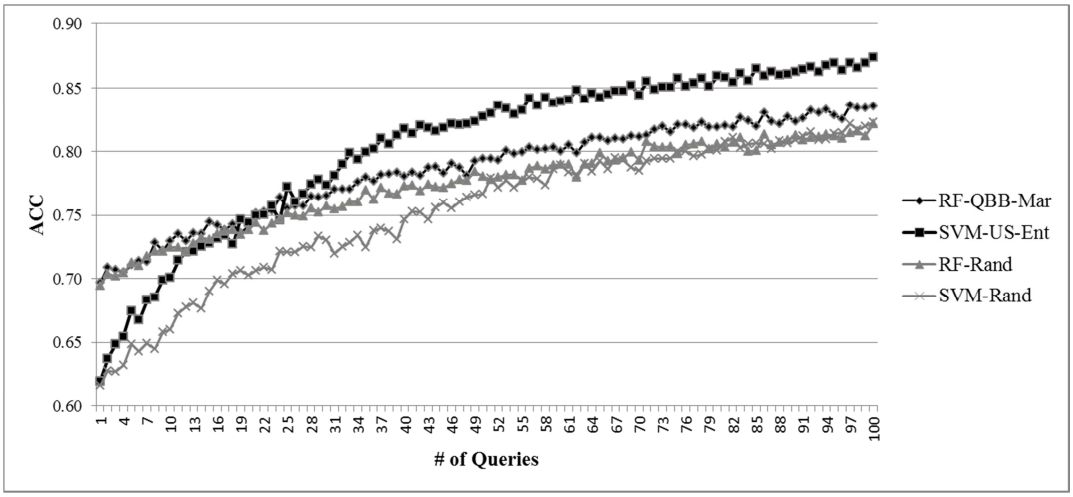


Figure 24. Learning curve of Acc (Reference set size 30. Value function has 3 attributes and 5 classes.)

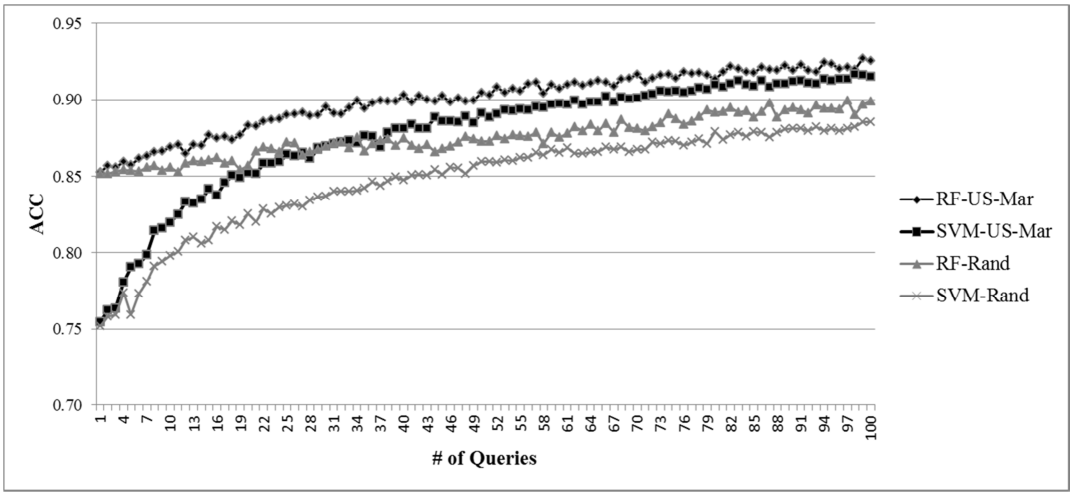


Figure 25. Learning curve of Acc (Reference set size 30. Value function has 6 attributes and 2 classes.)

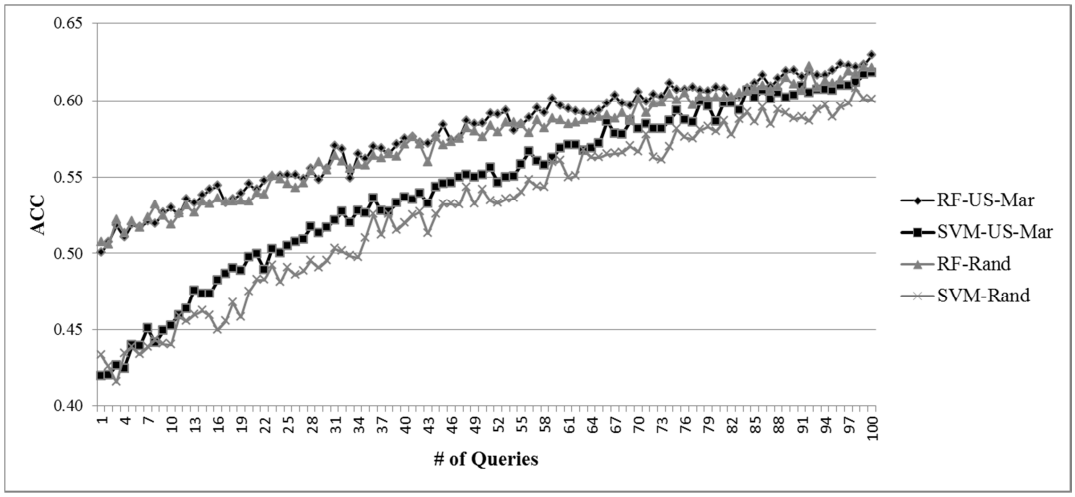


Figure 26. Learning curve of Acc (Reference set size 30. Value function has 6 attributes and 5 classes.)

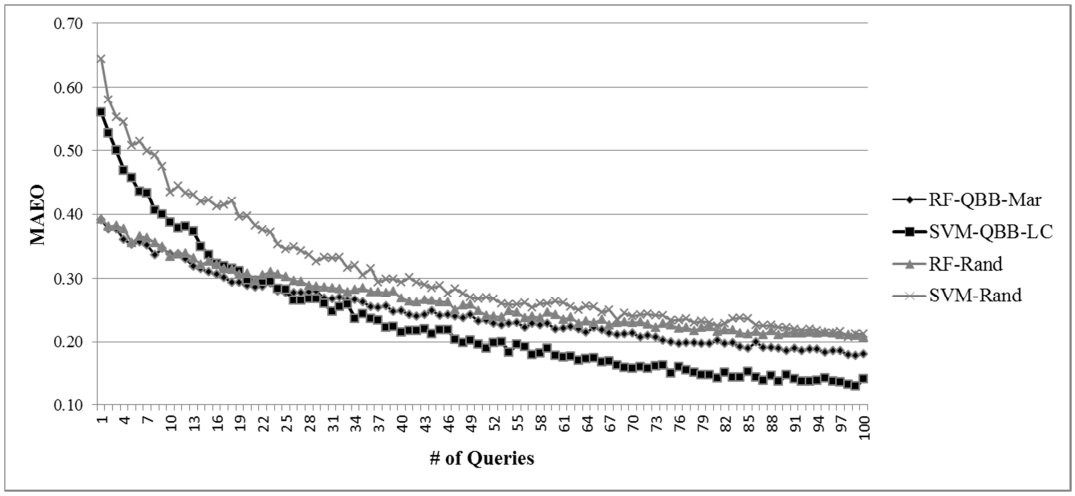


Figure 27. Learning curve of MAEO (Reference set size 10. Value function has 3 attributes and 5 classes.)

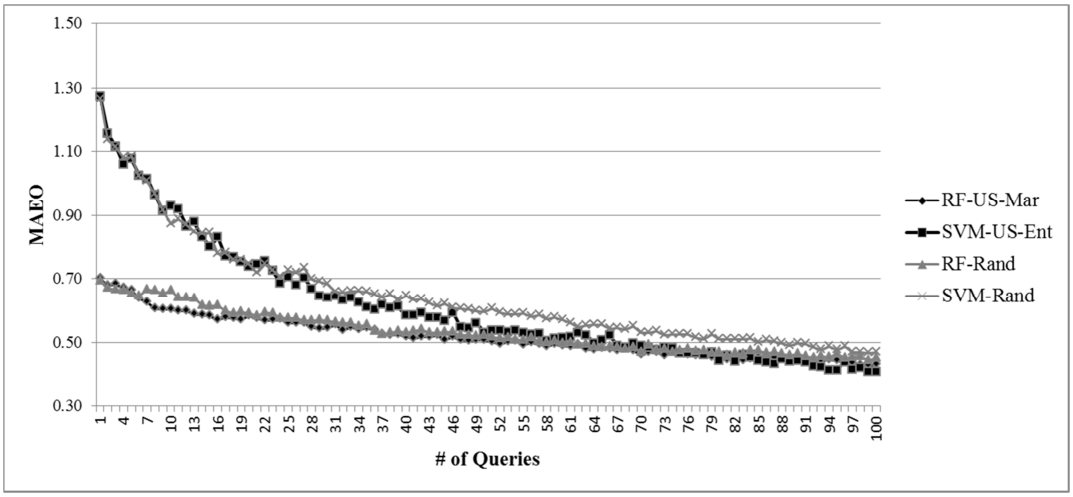


Figure 28. Learning curve of MAEO (Reference set size 10. Value function has 6 attributes and 5 classes.)

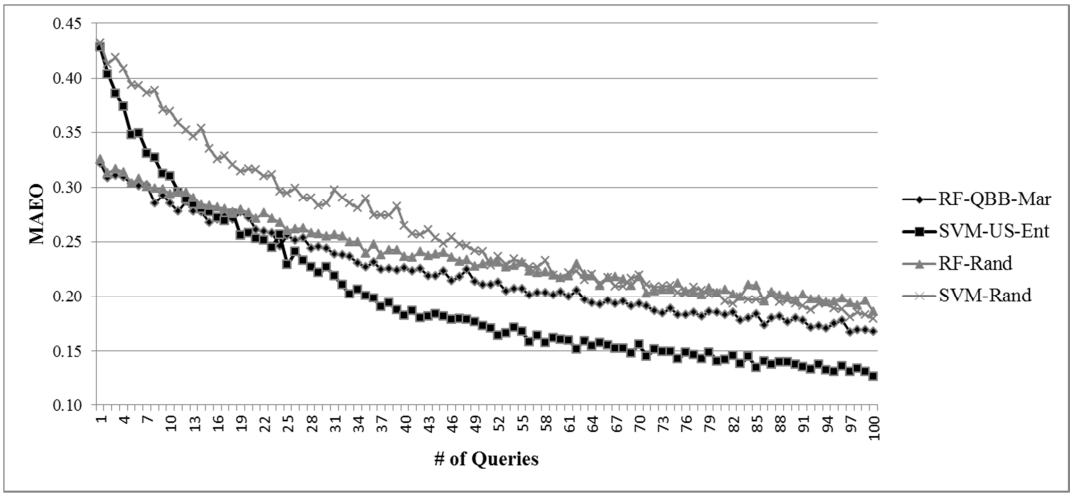


Figure 29. Learning curve of MAEO (Reference set size 30. Value function has 3 attributes and 5 classes.)

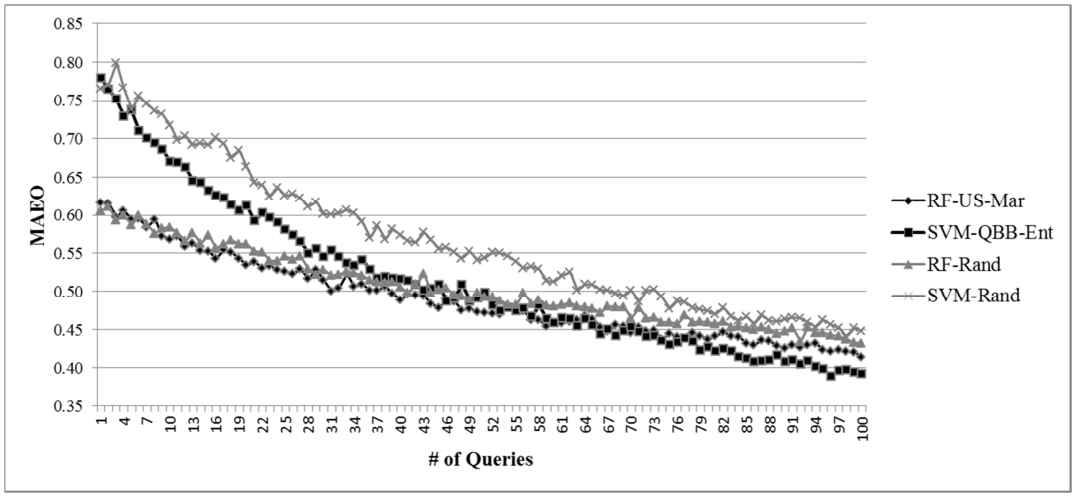


Figure 30. Learning curve of MAEO (Reference set size 60. Value function has 6 attributes and 5 classes.)

Learning curves show that when the classification task is relatively easy (small number of attributes and classes), proposed algorithms outperform the naive (random) approach by far. On the other hand, when the classification task gets more complex, proposed algorithms still outperform the random approach, however, the difference is not as significant as in the previous case. The reason for this result is believed to stem from the fact that, random approach has one advantage over the other querying strategies; the random approach explores regions in the attribute domain, which may not be visited by the proposed algorithms, particularly when the reference set size is small. While the proposed algorithms concentrate over some particular region in the attribute domain where uncertainty is the most, random approach might query an instance which might not provide an instant improvement, however, providing significant information gain for the whole querying process.

Another finding brought to our attention by learning curves is that, as size of the reference set increases, difference in performance between proposed querying

algorithms and random approach increases. This is to be expected, because proposed algorithms exploit information embedded in the reference set to guide interactive querying process. Hence, the more we have information, the better we explore the attribute domain.

Comparing performances of the classifiers, we realize that RF outperforms SVM at the initial phase of the querying process. This result can be verified with both Acc and MAEO learning curves. As querying process proceeds, SVM catches up with RF and in some cases performs better at the end of the process. This result is due to the ensemble characteristic of the RF classifier. As we have pointed out in Chapter 2, RFs are very robust to overfitting (Hastie et al., 2009), because they employ multiple decision trees generated with bagging. This ensemble characteristic improves generalization ability of the model. Furthermore, this property becomes very important when size of the training data is small because classifiers usually suffer from overfitting in the presence of small data sets. Consequently, it seems to be a better strategy to start with RF as the base learner and then switch to SVM (after 30-40 questions, according to learning curves). If we are certain that we are allowed to ask limited number of questions, using RF throughout the process would be the best course of action.

In order to verify if the best algorithms in terms of Acc and MAEO are statistically better than the random approach when the underlying value function is complex nonmonotonic, we perform pairwise t-tests for the mean values of Acc and MAEO measures achieved throughout the querying process. Following hypotheses are tested in the analysis at 95% confidence level.

$$H_o : d_i = 0$$
$$H_a : d_i > 0 \qquad (53)$$

where $d_i = Acc_i^{best} - Acc_i^{random}$ and $d_i = MAEO_i^{random} - MAEO_i^{best}$ for i=1,...,*numberOfQueries*. Test results are presented in Appendix C.

Pairwise t-tests propose that we have enough evidence to reject H_0, claiming there is no difference in the mean Acc and MAEO values of the best algorithms and the random approach. Therefore, we conclude that differences between the best algorithms and the random approach in terms of Acc and MAEO performance measures are statistically significant.

To sum up, in this chapter proposed algorithms are tested against various cases. Factors that are believed to have impact on the performance of the algorithms are included in the experimental design. In the experiment, underlying value functions that are believed to represent most of the preference structures are considered. Preference structure representing nonmonotonic behavior and having high order interactions among attributes is represented with a function that is built with hinge functions.

ANOVA reveal that all factors included in the experiment are statistically significant with respect to measures under consideration, meaning that all factors have impact on the performance of the querying algorithms. Furthermore, most of their two and three way interactions are significant, as well.

Regarding classification performances of the query algorithms, US algorithm outperforms QBB and the random approach in terms of all measures across different experimental conditions. We also conclude that, if we have no clue about functional form of the preference structure of the DM, which is the case we are

usually faced with, employing the US algorithm with the Margin uncertainty measure would be the best course of action.

Lastly, we have analyzed performance of the algorithms when the underlying value function is complex nonmonotonic. In this case, most of the performance measures propose using the US algorithm with the Margin uncertainty measure. Regarding the classifier, we conclude that it is a better strategy to start with RF as the base learner and then switch to SVM after asking a decent number of questions.

In order to verify that the best algorithms determined outperform the random approach, we have performed pairwise t-tests to mean values of Acc and MAEO achieved throughout the querying process. These tests confirm that the differences in means between the best algorithms and the random approach are statistically significant in all cases.

CHAPTER 5

EXTENSION OF THE ALGORITHM TO CONSIDER THE INPUT DISTRIBUTION

In this thesis study, we have focused on the case where an unlabeled data does not exist, as discussed in Section 3.2. Therefore, we have not made use of input distribution of such data. Our aim is to estimate a function on a particular domain (design domain) where we desire to perform equally good everywhere. This requirement makes our problem a more challenging one with respect to typical AL practices. This situation is particularly valid in CA applications. In CA, preference of the customers (or potential customers) is elicited by presenting alternative profiles that are generated according to a strategy. In traditional CA, these alternatives are usually generated via a statistical experimental design while in ACA subsequent profiles (questions) are tailored in an interactive query session in order to maximize information gain. Possible objectives of these applications are to estimate market share of an existing or a new product, estimating how much each feature of the profile contributes to overall preference (part-worths), classifying customer profiles and developing custom marketing policies for each group. As process definition of the CA implies, features (or attributes) of the product or part-worths will be determined based on the preference structure of the customers.

A similar application can be found in the manufacturing field. For example, quality of a part or a product might need to be predicted for some given process parameter levels for the purpose of improving quality of products. This is similar to aforementioned profile development problem in CA. In both of these applications and many others such as student selection to graduate programs, credit risk assessment, classifying hotels or diagnosing diseases, we are not interested in how often a particular attribute value or joint values of attributes in other existing unlabeled examples (or alternatives) out there happen to occur.

In some other cases, the joint distribution of attribute levels may change from one setting of parameters that produce alternatives to another setting. Let us consider an example from the manufacturing field. A manufactured boot is classified as “Good”, “Rework” or “Scrap” based on the quality characteristics under consideration. A Good boot is ready for market sale while a boot classified as Rework needs to undergo a corrective operation in order to be qualified as Good for the market. There is no hope for a Scrap boot and it goes to garbage. As the problem definition implies, this is an ordinal classification problem. A quality control technician classifies all manufactured boots accordingly. Assume that two of the quality characteristics under consideration are thickness and flexibility of boot sole and also assume that we measure these two quality characteristics quantitatively. Suppose that measured quality characteristics of manufactured boots follow a normal distribution at each characteristic domain, based on current process parameter levels. If we change levels of the process parameters, then quality characteristics may follow different distributions or at least the same distributions with different parameters. If we keep the current parameter levels, we may observe that thickness and flexibility have negative correlation since a thick sole is usually not flexible, consequently, resulting in a joint multivariate normal distribution in two dimensional attribute domain. This is the input distribution of two quality characteristics and an example of such distribution is shown in Figure

31 for illustration. This input distribution can be estimated by analyzing previously manufactured and unclassified pool of boots.

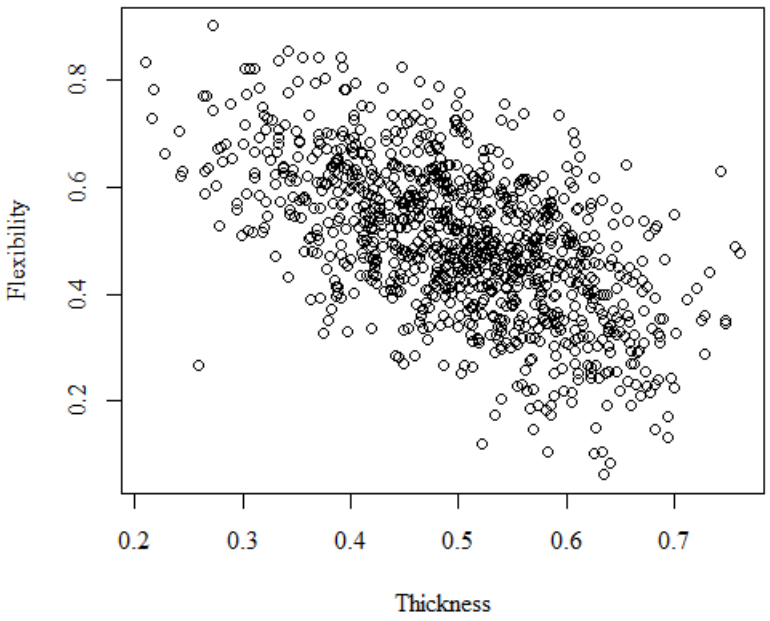


Figure 31. Input distribution of two quality characteristics.

In the case of a constant and known input distribution, does this information help in training a classifier that will be used as a model in predicting resulting classes such as qualities of manufactured boots? Attribute values of most of the manufactured boots to be classified will be lying in the dense region shown in Figure 31. If our ultimate goal is to train a predictive model that will classify instances with high accuracy, concentrating on this dense region in the training phase will boost the learning process. Recognizing this special property of some problems, AL practitioners developed “Density Weighted Methods” and “Cluster Based AL” techniques in order to exploit information embedded in the input

distribution. In this context, density weighted methods consider not only information content of a candidate question but also how much it represents other instances in the unlabeled pool. Hence, these methods measure information content and representativeness of candidate instances and aggregate these two quantities to determine the instance that will boost the learning process most. Representativeness is usually measured with a similarity measure which computes average similarity of a candidate instance to all instances in the unlabeled pool. Consequently, assuming that we have a set of unlabeled instances *U*, an aggregate measure as shown below is utilized to select the next question to be queried (Settles, 2012):

$$x^* = \underset{x}{\operatorname{argmax}} \Phi_A(x) \left(\frac{1}{|U|} \sum_{x' \in U} \operatorname{sim}(x, x') \right)^{\beta} \tag{54}$$

where $\Phi_A(x)$ corresponds to base uncertainty measure under sampling strategy *A*, and the rest in the parenthesis measures representativeness of the candidate instance. Parameter β determines weight of the representativeness in the aggregate measure. Similarity (sim(.)) can be measured with different distance measures such as Euclidean distance or Cosine similarity (Settles, 2012).

In order to observe efficiency of the aforementioned extension, we have applied it to our most demanding experimental case, where the underlying value function is 6-attribute complex nonmonotonic, instances are to be classified into 5 classes and initial reference set size is 10. SVM is employed as the base learner and Euclidean distance is used to measure similarity. Being the most successful querying strategy, US is utilized. In order to mimic the input distribution, we have randomly generated an unlabeled set of size 1,000 from a multivariate normal distribution, where $\mathbf{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with parameters;

$$\boldsymbol{\mu} = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix}, \quad \boldsymbol{\Sigma} = \begin{bmatrix} 0.0100 & 0.0050 & 0.0000 & 0.0000 & 0.0000 & 0.0023 \\ 0.0050 & 0.0200 & 0.0011 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0011 & 0.0050 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0200 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0100 & 0.0000 \\ 0.0023 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0200 \end{bmatrix}$$

In line with the problem definition, the test set of size 1,000 is also generated from the same multivariate normal distribution. Pseudo-code of the algorithm that considers the input distribution is shown in Figure 32.

1.	***R* is the initial reference set of size *n*, *m* is the number of queries, *L* is the learner, *p* is the pool size, *U* is the unlabeled set of instances**
2.	**for** i=1, 2, …m **do**
3.	train L using R
4.	generate random set of instances from the attribute domain of size p for the *pool*
5.	predict labels of instances in the *pool* using L
6.	calculate uncertainty measure ($\Phi(x)$) for each instance in the *pool*
7.	calculate average similarity ($\Omega(x)$) of each instance in the pool to the instances in set U
8.	calculate information content of each instance in the pool ($\Phi(x) . \Omega(x)^{\beta}$)
9.	select instance having the maximum information content (w^*)
10.	Ask label (class) y of w^* to the DM
11.	$R \leftarrow R \cup \langle w^*, y \rangle$
12.	$n \leftarrow n + 1$
13.	**end for**

Figure 32. Pseudo-code of the algorithm that considers the input distribution.

Performance measures of the experiment and comparison with the previous results where input distribution is not considered is presented in Table 33. Table 33 shows that considering input distribution helps classifier train more efficiently and boosts the learning process. In all measures, extension of the algorithm outperforms the original algorithm (that does not consider input distribution) and random approach.

Table 33. Experimental results of the extension and comparison with previous results.

Performance Measure	**Considering Input Distribution**			**Not Considering Input Distribution**			**Random**
	LC	**M**	**E**	**LC**	**M**	**E**	
Acc	**0.7583**	0.6828	0.7519	0.6081	0.6185	0.6172	0.5840
BCA	0.5754	**0.6656**	0.5770	0.5840	0.5947	0.5985	0.5626
Kappa	**0.6562**	0.5619	0.6481	0.5003	0.5146	0.5137	0.4709
MAEO	**0.2433**	0.3203	0.2530	0.4143	0.4176	0.4089	0.4690
MSEO	**10.6424**	26.1933	11.5735	26.7399	19.0762	23.4776	24.0141

Figure 33 and Figure 34 show learning curves of the extension of the algorithm for Acc and MAEO performance measures. These figures reveal that learning curves of the algorithm considering input distribution is significantly steeper than the random approach. Furthermore, difference between the algorithm and the random approach is statistically significant even at early phases of the learning process.

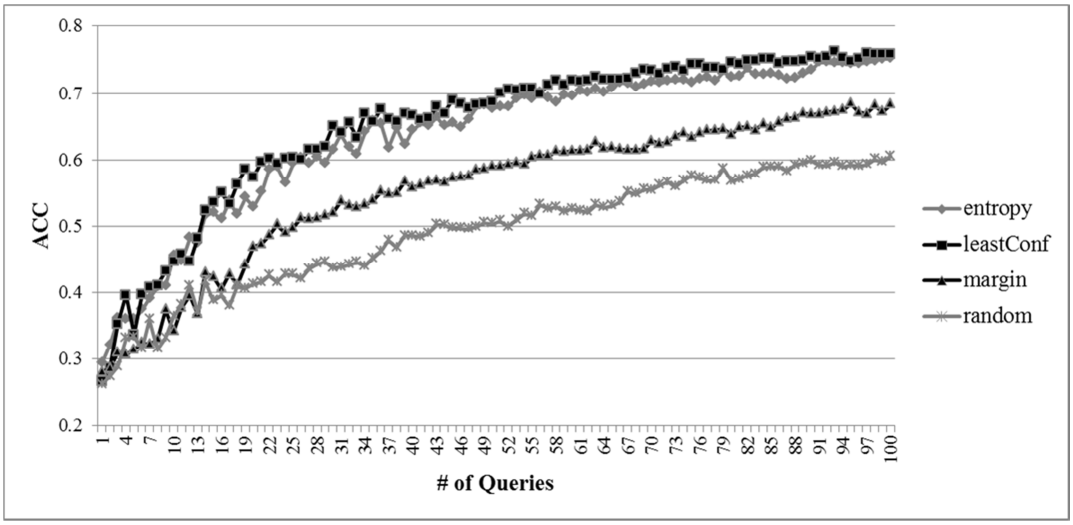


Figure 33. Learning curve of Acc (considering the input distribution).

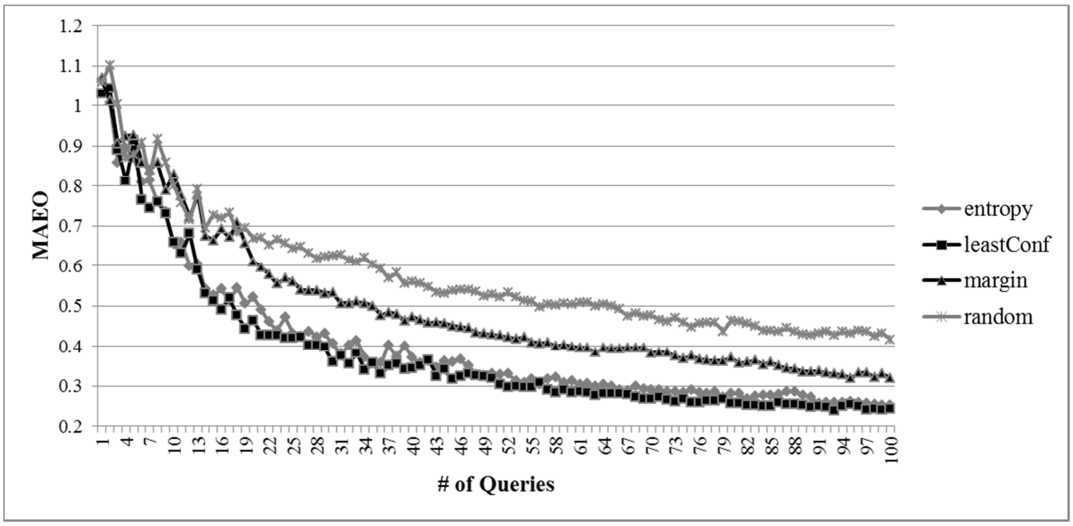


Figure 34. Learning curve of MAEO (considering the input distribution).

To sum up, considering input distribution boosts learning process of the algorithm because line of inquiry is driven so that the algorithm focuses on the dense regions

of the input space, hence, producing a model having a better predictive ability. However, this extension is not applicable to all preference learning applications, rather, depends on the problem of interest. If nature of the problem enables us to exploit input distribution as in the boot manufacturing example, then the algorithm given in Figure 32 helps us generate superior predictive models.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

In this study, we consider learning preference structure of a DM in an ordinal classification setting especially when the criteria interact, no information is available about underlying value function and distribution of the criteria value, and prior or new data about preferences are difficult to obtain. Interacting criteria make the preference learning problem more challenging particularly when the number of criteria under consideration is big, due to curse of dimensionality. Even though there is a general consent among researchers regarding the existence of interaction among criteria in real life decision problems, it is often ignored in applications. Accordingly, most of the preference modeling strategies assume preferential independence among criteria, making modeling process relatively easygoing. Nevertheless, interaction phenomonea is encountered quite commonly, even in simpler cases.

Most of the MCDA methodologies utilizing value functions assume an underlying functional model. According to this assumption, DM preference structure is compatible with an adapted functional form, which is generally monotonous. However, a wrong functional form produces estimation bias and reduces the predictive performance of the model. Additionally, even though a proper functional (i.e. nonlinear) form is assumed for a preferential system having

interactions among criteria, parametric models may fail to address complex interaction structures in high dimensions.

In order to deal with these problematic issues in preference modeling, we propose using nonparametric SL techniques interactively and consider preference modeling for sorting/ordinal classification decision problem. SL offers several nonparametric model-free methodologies that are particularly superior in terms of generalization ability and modeling complex data structures. Finding a vast application area in AI for human behavioral modeling, SL techniques draw a growing interest among MCDA researchers, as well. However, there are several issues criticized about usage of SL techniques in MCDA applications. The most important ones of these critics are their need for big amount of data and lack of ability to work interactively (Doumpos & Zopounidis, 2011). In this study, we address these criticized issues while exploiting strong features of SL for learning DM preference structure in sorting/ordinal classification setting.

Our modeling strategy is based on obtaining holistic judgements from the DM regarding alternatives and adjusting subsequent questions based on the judgements gathered thus far, in an adaptive fashion. We start with a small reference set and employ nonparametric classifiers for model developing. In order to conduct modeling process in an adaptive way, we propose employing AL techniques. Thus, querying process is structured so that as much information as possible is obtained while as less number of questions as possible is queried. Utilizing AL, we ask the DM in an interactive way, thereby, the DM is integrated into the model development process. Consequently, preference structure of the DM is represented with the trained classifier.

In order to evaluate the proposed approach, we have conducted an experimental analysis. In the experiment we have tested the approach against several sorting setups, i.e. problems having different number of classes, attributes, questions and

reference set sizes. Additionally, we have employed four different underlying value functions that were believed to represent majority of preference structures in order to mimic the DM. Our experimental analysis proves that the proposed approach outperforms the “naive” approach where subsequent questions are asked randomly. Based on our findings, we provide algorithmic recommendations (type of classifier, sampling strategy and uncertainty measure) for modeling different underlying value functions in case we have information about form of the preference structure. Additionally, we provide an extension of the algorithm where we have information about input distribution of the criteria values. For the case where we have no information about functional form of the preference structure of the DM, which is usually the case we are faced with, we recommend using the US algorithm with Margin uncertainty measure and training RF as the classifier.

Our approach is a novel one because different than the MCDA sorting approaches proposed in the literature that aim at sorting limited number of alternatives of the problem under consideration with maximum accuracy, we consider preference modeling as a learning process. To our knowledge, this is the first study in the MCDA literature that approaches preference modeling as an evolutionary learning process. In this respect, this study can be regarded as a pioneering approach. Even though our work is not comparable with MCDA sorting methodologies, our trained model offers a sorting tool, as well. Thereby, our proposed approach also addresses the need for model-free sorting methodologies that are capable of modeling interactions while implemented interactively with the DM.

Our main objective is to develop a predictive model that will be used to classify unseen instances with high accuracy. At the end of the learning process, our approach provides a black-box model where the model produces outputs (class predictions) for a given set of criteria/attribute values. In this context, produced model can be used as an aide for optimization purposes where criteria/attribute regions of high preference can be found by performing a grid search. Developed

models can be used in various fields such as quality engineering and marketing, as well as many others that need robust ordinal classifiers modeling preference structure of the DM (Doumpos & Zopounidis, 2002). With these properties, our approach can be compared with those approaches developed in the ACA because similar to our approach, ACA aims to develop robust predictive models that learns efficiently via questionnaires that are specifically structured to accelerate learning process. We have not conducted an evaluation to compare our study with those of ACA, however, it is our intention to perform such an evaluation as a future work. Nevertheless, we expect our proposed approach to outperform methodologies of ACA since ACA methodologies are not capable of modeling interacting attributes (Rao, 2014).

There are three subjects that we consider as future work. One of them is developing a "starting strategy". As we explain in Section 3.2, we perform a stratified random sampling at the beginning of the process in order to generate an initial reference set. Samples from each stratum (or class) are generated randomly. Including at least one sample from each stratum is essential for classifiers because they recognize number of classes of the problem with regard to examples in the training set. On the other hand, there may be additional information regarding criteria domain based on the problem under consideration. In quality engineering, for instance, the DM may define upper or lower specification limits on the criteria (or attribute) domain that will help us localize class-separating boundaries. Asking initial questions in the vicinity of these boundaries may help accelerate the initial learning process.

Another potential area of improvement is development of new uncertainty measures for the ordinal classification problem. Measures proposed in the literature are developed for nominal classification problem, hence, they do not consider additional information embedded in the ordinal structure of the data. Although we have developed measures of that sort, as explained in Section 3.3,

those measures have not provided better results. This might be partially due to the fact that, classifiers we use in the study are nominal classifiers and we make them exploit ordinal structure of the data with Frank and Hall (2001) approach. The main problem with this approach is that generated posterior probabilities do not add up to one. Uncertainty measures consider posterior probabilities for measuring uncertainty, therefore, they suffer from this property of Frank and Hall (2001). Even though there are special algorithms for ordinal classification, our trials have shown that they generally failed with a small number of training data. That is another reason why we prefer using SVM and RF as base learners and made them exploit ordinal information with Frank and Hall (2001) approach. Consequently, special algorithms and uncertainty measures developed for ordinal classification may further improve performance of the proposed approach.

Yet another future work we consider is to develop a web interface for the algorithm where the DM is interacted via the interface and information about the learning process is provided such as rate of learning, prediction performance of the learner at that point, etc. Thus, the DM would be informed about the progress of the learning process and consequences of terminating the model development process at a particular stage.

REFERENCES

Abe, N., & Mamitsuka, H. (1998). Query learning atrategies using boosting and bagging. In *Machine Learning: Proceedings of the Fifteenth International Conference (ICML'98)*.

Abernethy, J., Evgeniou, T., Toubia, O., & Vert, J. (2008). Robust Adaptive Choice Questionnaires. *IEEE Transactions on Knowledge and Data Engineering*, *20*(2), 145–155.

Angilella, S., Greco, S., & Matarazzo, B. (2009). Non-additive robust ordinal regression with Choquet integral, bipolar and level dependent Choquet integrals. *Proceedings of the Joint 2009 International Fuzzy Systems Association World Congress and 2009 European Society of Fuzzy Logic and Technology Conference*, 1194–1199.

Breiman, L. (1996). Bagging Predictors. *Machine Learning*, *140*(24), 123–140.

Breiman, L. (2001). Random forest. *Machine Learning*, *45*(1), 2–35.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Belmont, CA.

Buğdacı, A. G., Köksalan, M., Özpeynirci, S., & Serin, Y. (2013). an Interactive Probabilistic Approach To Multi-Criteria Sorting. *IIE Transactions*, *45*, 1048–1058. doi:10.1080/0740817X.2012.721945

Campbell, C., & Ying, Y. (2011). *Learning with Support Vector Machines*. *Synthesis Lectures on Artificial Intelligence and Machine Learning* (Vol. 5). doi:10.2200/S00324ED1V01Y201102AIM010

Cardoso, J. S., & Sousa, R. (2011). Measuring the Performance of Ordinal

Classification. *International Journal of Pattern Recognition and Artificial Intelligence*, *25*(08), 1173–1195. doi:10.1142/S0218001411009093

Chu, W., & Keerthi, S. S. (2007). Support vector ordinal regression. *Neural Computation*, *19*(3), 792–815. doi:10.1162/neco.2007.19.3.792

Corrente, S. (2012). *Hierarcy and interaction of criteria in robust ordinal regression*. University of Catania.

Corrente, S., Doumpos, M., Greco, S., Słowiński, R., & Zopounidis, C. (2015). Multiple criteria hierarchy process for sorting problems based on ordinal regression with additive value functions. *Annals of Operations Research*. doi:10.1007/s10479-015-1898-1

Corrente, S., Greco, S., & Słowiński, R. (2012). Multiple criteria hierarchy process in robust ordinal regression. *Decision Support Systems*, *53*(3), 660–674. doi:10.1016/j.dss.2012.03.004

Cui, D., & Curry, D. (2005). Prediction in Marketing Using the Support Vector Machine. *Marketing Science*, *24*(4), 595–615. doi:10.1287/mksc.1050.0123

Çelik, B., Karasakal, E., & İyigün, C. (2015). A probabilistic multiple criteria sorting approach based on distance functions. *Expert Systems with Applications*, *42*(7), 3610–3618. doi:10.1016/j.eswa.2014.11.049

Devaud, J. M., Groussaud, G., & Jaquet-Lagreze, E. (1980). UTADIS: Une méthode de construction de fonctions d'utilité additives rendant compte de jugements globaux. In *Proceedings of the European woking group on MCDA*. Bochum, Germany.

Dolgun, L. E. (2014). *Tercihsel bağımlılık altında alternatifler için aralık ölçeğinde tercih puanlarının belirlenmesi*. Unpublished Dissertation. Eskişehir Osmangazi University.

Doumpos, M., & Grigoroudis, E. (Eds.). (2013). *Multicriteria Decision Aid and Artificial Intelligence: Links, Theory and Applications*. John Wiley & Sons.

Doumpos, M., & Zopounidis, C. (2002). *Multicriteria Decision Aid and Classification Methods*. Dordrecht: Kluwer Academic Publishers.

Doumpos, M., & Zopounidis, C. (2011). Preference disaggregation and statistical learning for multicriteria decision support: A review. *European Journal of Operational Research*, *209*(3), 203–214. doi:10.1016/j.ejor.2010.05.029

Duan, K.-B., & Keerthi, S. S. (2005). Which Is the Best Multiclass SVM Method? An Empirical Study. *Multiple Classifier Systems*, *3541*, 278–285. doi:10.1007/b136985

Dyer, J. S. (2005). MAUT – Multiattribute Utility Theory. In J. Figueira, S. Greco, & M. Ehrgott (Eds.), *Multiple Criteria Decision Analysis: State of the Art Surveys* (pp. 265–295). Springer.

Evgeniou, T., Boussios, C., & Zacharia, G. (2005). Generalized Robust Conjoint Estimation. *Marketing Science*, *24*(3), 415–429. doi:10.1287/mksc.1040.0100

Ferri, C., Hernández-Orallo, J., & Modroiu, R. (2009). An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, *30*(1), 27–38. doi:10.1016/j.patrec.2008.08.010

Figueira, J. R., Greco, S., & Słowiński, R. (2009). Building a set of additive value functions representing a reference preorder and intensities of preference: GRIP method. *European Journal of Operational Research*, *195*(2), 460–486. doi:10.1016/j.ejor.2008.02.006

Frank, E., & Hall, M. (2001). A simple approach to ordinal classification. In L. De Raedt & P. Flach (Eds.), *Machine Learning: ECML 2001* (Vol. 2167, pp. 145–156). Springer Berlin Heidelberg. doi:10.1007/3-540-44795-4_13

Freund, Y., & Schapire, R. E. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139. doi:10.1006/jcss.1997.1504

Friedman, J. H. (1991). Multivariate Adaptive Regression Splines. *The Annals of Statistics*, *19*(1), 1–141.

Greco, S., Kadziński, M., & Słowiński, R. (2011). Selection of a representative value function in robust multiple criteria sorting. *Computers & Operations Research*, *38*, 1620–1637. doi:10.1016/j.cor.2011.02.033

Greco, S., Matarazzo, B., & Slowinski, R. (2001). Rough sets theory for multicriteria decision analysis. *European Journal of Operational Research*, *129*(1), 1–47. doi:10.1016/S0377-2217(00)00167-3

Greco, S., Matarazzo, B., & Slowinski, R. (2002). Rough sets methodology for sorting problems in presence of multiple attributes and criteria. *European Journal of Operational Research*, *138*(2), 247–259. doi:10.1016/S0377-2217(01)00244-2

Greco, S., Mousseau, V., & Słowiński, R. (2010). Multiple criteria sorting with a set of additive value functions. *European Journal of Operational Research*, *207*(3), 1455–1470. doi:10.1016/j.ejor.2010.05.021

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*. Springer. doi:10.1007/b94608

Hühn, J. C., & Hüllermeier, E. (2009). Is an ordinal class structure useful in classifier learning? *International Journal of Data Mining, Modelling and Management*, *1*(1), 45–67. doi:10.1504/IJDMMM.2008.022537

Jacquet-Lagreze, E., & Siskos, Y. (2001). Preference disaggregation : 20 years of MCDA experience. *European Journal of Operational Research*, *130*, 233–245.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning with Applications in R*. doi:10.1007/978-1-4614-7138-7

Jaquet-Lagreze, E., & Siskos, Y. (1982). Assessing a set of additive utility functions for multicriteria decision making: The UTA method. *European Journal of Operational Research*, *10*, 151–164.

Keeney, R. L., & Raiffa, H. (1993). *Decisions with Multiple Objectives*. New York, USA: Cambridge University Press.

Kotsiantis, S. B. (2007). Supervised Machine Learning: a review of classification

techniques. *Informatica*, *31*, 249–268.

Kotz, S., Balakrishnan, N., Read, C. B., & Vidakovic, B. (2006). *Encyclopedia of Statistical Sciences*. New Jersey: John Wiley & Sons.

Köksalan, M., & Bilgin Özpeynirci, S. (2009). An interactive sorting method for additive utility functions. *Computers and Operations Research*, *36*(9), 2565–2572. doi:10.1016/j.cor.2008.11.006

Köksalan, M., Mousseau, V., Özpeynirci, Ö., & Bilgin Özpeynirci, S. (2009). A new outranking-based approach for assigning alternatives to ordered classes. *Naval Research Logistics*, *56*, 74–85. doi:10.1002/nav

Köksalan, M., & Ulu, C. (2003). An interactive approach for placing alternatives in preference classes. *European Journal of Operational Research*, *144*(2), 429–439. doi:10.1016/S0377-2217(02)00138-8

Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, *22*(1), 79–86.

Leroy, A., Mousseau, V., & Pirlot, M. (2011). Learning the parameters of a multiple criteria sorting method. In *Algorithmic Decision Theory* (pp. 219–233). Springer Berlin Heidelberg. doi:10.1007/978-3-642-24873-3_17

Lim, T.-S., Loh, W.-Y., & Shih, Y.-S. (2000). A comparison of prediction accuracy, complexity, and training time of thiry-three old and new classification algorithms. *Machine Learning*, *40*(1992), 203–229. doi:10.1023/A:1007608224229

Madjarov, G., Kocev, D., Gjorgjevikj, D., & Džeroski, S. (2012). An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, *45*(9), 3084–3104. doi:10.1016/j.patcog.2012.03.004

Marichal, J. (2000). An axiomatic approach of the discrete choquet integral as a tool to aggregate interacting criteria. *IEEE Transactions on Fuzzy Systems*, *8*(6), 800–807.

Martinez, W. L., & Martinez, A. R. (2002). *Computational Statistics Handbook with MATLAB*. Boca Raton: Chapman & Hall/CRC.

Max, K., Weston, S., Keefer, C., Engelhardt, A., Cooper, T., Mayer, Z., … Scrucca, L. (2015). caret: Classification and Regression Training. R package version 6.0-47. http://CRAN.R-project.org/package=caret.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., & Leisch, F. (2014). e1071: Misc Functions of the Department of Statistics (e1071). R package version 1.6-4. http://cran.r-preject.org/package=e1071.

Minitab 16 Statistical Software. (2010). [Computer Software]. State College, PA: Minitab, Inc. Retrieved from www.minitab.com

Montgomery, D. C. (2009). *Design and Analysis of Experiments*. Asia: John Wiley & Sons.

Pawlak, Z. (1982). Rough sets. *International Journal of Information and Computer Sciences*, *11*, 341–356.

Pinto da Costa, J. F., Alonso, H., & Cardoso, J. S. (2008). The unimodal model for the classification of ordinal data. *Neural Networks*, *21*(1), 78–91. doi:10.1016/j.neunet.2007.10.003

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Los Altos, California: Morgan Kaufmann Publishers.

R Core Team. (2014). R: A language and environment for statistical computing. R Foundation for Statistical Computing. Vienna, Austria. Retrieved from http://www.r-project.org/

Rao, V. R. (2014). *Applied Conjoint Analysis*. Springer Berlin Heidelberg. doi:10.1007/978-3-540-87753-0

Roy, B. (1996). *Multicriteria Methodology for Decision Aiding*. New York, USA: Springer.

Schölkopf, B., & Smola, A. J. (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. London, England: The MIT Press. doi:10.1109/TNN.2005.848998

Seni, G., & Elder, J. F. (2010). *Ensemble Methods in Data Mining: Improving Accuracy Through Combining Predictions*. Morgan & Claypool Publishers.

Settles, B. (2012). *Active Learning*. *Synthesis Lectures on Artificial Intelligence and Machine Learning* (Vol. 6). Morgan & Claypool Publishers.

Siskos, Y., Grigoroudis, E., & Matsatsinis, N. F. (2005). UTA Methods. In J. Figueira, S. Greco, & M. Ehrgott (Eds.), *Multiple Criteria Decision Analysis: State of the Art Surveys*.

Soylu, B. (2011). A multi-criteria sorting procedure with Tchebycheff utility function. *Computers and Operations Research*, *38*(8), 1091–1102. doi:10.1016/j.cor.2010.09.009

Stone, B. C. J., Hansen, M. H., Kooperberg, C., Truong, Y. K., Laboratories, B., & Hill, C. (1997). Polynomial splines and their tensor products in extended linear modeling. *The Annals of Statistics*, *25*(4), 1371–1470.

Su, C., & Hsiao, Y. (2009). Multiclass MTS for Simultaneous Feature Selection and Classification. *IEEE Transactions on Knowledge and Data Engineering*, *21*(2), 192–205. doi:10.1109/TKDE.2008.128

Tabachnick, B. G., & Fidell, L. S. (2007). *Using Multivariate Statistics*. Pearson Education, Inc. doi:10.1037/022267

Teichert, T., & Shehu, E. (2013). Evolutionary conjoint. In A. Gustaffson, A. Herrmann, & F. Huber (Eds.), *Conjoint Measurement: Methods and Applications* (p. 373). Springer Berlin Heidelberg.

Toubia, O., Hauser, J., & Garcia, R. (2007). Probabilistic Polyhedral Methods for Adaptive Choice-Based Conjoint Analysis: Theory and Application. *Marketing Science*, *26*(5), 596–610. doi:10.1287/mksc.1060.0257

Toubia, O., Simester, D. I., Hauser, J. R., & Dahan, E. (2003). Fast Polyhedral Adaptive Conjoint Estimation. *Marketing Science*, *22*(3), 273–303. doi:10.1287/mksc.22.3.273.17743

Ulu, C., & Köksalan, M. (2001). An interactive procedure for selecting acceptable alternatives in the presence of multiple criteria. *Naval Research Logistics*, *48*(7), 592–606. doi:10.1002/nav.1036

Ulu, C., & Köksalan, M. (2014). An interactive approach to multicriteria sorting for quasiconcave value functions. *Naval Research Logistics*, *61*, 447–457. doi:10.1002/nav.21595

Waegeman, W., & Boullart, L. (2009). An ensemble of Weighted Support Vector Machines for Ordinal Regression. *International Journal of Computer Systems Science and Engineering*, *3*(1), 47–51.

Waegeman, W., De Baets, B., & Boullart, L. (2009). Kernel-based learning methods for preference aggregation. *A Quarterly Journal of Operations Research*, *7*(2), 169–189. doi:10.1007/s10288-008-0085-5

Weber, G.-W., Batmaz, İ., Köksal, G., Taylan, P., & Yerlikaya-Özkurt, F. (2012). CMARS: a new contribution to nonparametric regression with multivariate adaptive regression splines supported by continuous optimization. *Inverse Problems in Science and Engineering*, *20*(3), 371–400.

Yu, W. (1992). *ELECTRE TRI: Aspects methodologiques et manuel d'utilisation*.

Zopounidis, C., & Doumpos, M. (2002). Multicriteria classification and sorting methods: A literature review. *European Journal of Operational Research*, *138*(2), 229–246. doi:10.1016/S0377-2217(01)00243-0

APPENDIX A

DETAILED PERFORMANCE MEASURE RESULTS OF THE ALGORITHMS ACROSS ALL FACTORS CONSIDERED

Table A.1 Detailed Acc results (Uncertainty Sampling).

# of Queries	Ref Set Size	Underlying Value Function	# of attr.	# of class.	Uncertainty Sampling						Rand RF	Rand SVM
					LC		M		E			
					RF	SVM	RF	SVM	RF	SVM		
30	10	Linear	3	2	0.919	0.948	0.914	**0.949**	0.918	0.945	0.887	0.900
			3	5	0.616	0.734	0.626	0.722	0.569	0.736	0.613	0.698
			6	2	0.852	0.851	**0.857**	0.839	0.844	0.841	0.815	0.805
			6	5	0.396	0.470	0.473	0.497	0.397	0.431	0.479	0.476
		Multiplicative	3	2	0.912	0.946	0.909	**0.948**	0.911	0.942	0.880	0.906
			3	5	0.585	**0.850**	0.642	0.816	0.579	0.829	0.610	0.707
			6	2	0.848	0.840	0.847	0.842	**0.852**	0.842	0.809	0.803
			6	5	0.418	0.477	0.463	0.495	0.411	0.507	0.465	0.490
		Tchebycheff	3	2	**1.000**	0.823	**1.000**	0.835	**1.000**	0.815	0.976	0.895
			3	5	**0.962**	0.775	0.948	0.726	0.959	0.757	0.894	0.700
			6	2	**1.000**	0.811	1.000	0.827	1.000	0.813	0.969	0.786
			6	5	**0.958**	0.466	0.956	0.489	0.944	0.476	0.875	0.480
		Complex Nonmonotonic	3	2	0.943	0.822	0.944	0.825	**0.944**	0.829	0.915	0.896
			3	5	0.706	0.745	0.747	0.735	0.696	0.749	0.725	0.712
			6	2	0.877	0.846	**0.885**	0.842	0.880	0.841	0.847	0.801
			6	5	0.497	0.479	**0.546**	0.490	0.498	0.479	0.530	0.451
	30	Linear	3	2	0.923	0.958	0.932	**0.960**	0.925	0.957	0.891	0.924
			3	5	0.653	**0.796**	0.682	0.772	0.637	0.786	0.655	0.725
			6	2	0.867	0.867	0.871	0.864	0.863	0.862	0.833	0.840
			6	5	0.428	0.515	0.492	**0.535**	0.417	0.509	0.490	0.529

Table A.1 (continued).

# of Queries	Ref Set Size	Underlying Value Function	# of attr.	# of class.	Uncertainty Sampling						Rand RF	Rand SVM
					LC		M		E			
					RF	SVM	RF	SVM	RF	SVM		
30	30	Multiplicative	3	2	0.917	**0.960**	0.917	0.958	0.919	0.957	0.891	0.923
			3	5	0.634	**0.869**	0.678	0.840	0.620	0.855	0.662	0.728
			6	2	0.861	0.868	0.859	0.870	0.864	**0.871**	0.830	0.839
			6	5	0.458	0.562	0.496	0.563	0.444	0.553	0.502	0.532
		Tchebycheff	3	2	1.000	0.897	**1.000**	0.893	1.000	0.892	0.980	0.914
			3	5	**0.976**	0.798	0.976	0.774	0.970	0.814	0.928	0.753
			6	2	**1.000**	0.851	1.000	0.848	1.000	0.849	0.977	0.817
			6	5	**0.981**	0.521	0.976	0.521	0.967	0.523	0.925	0.520
		Complex Nonmonotonic	3	2	0.949	0.897	**0.950**	0.901	0.948	0.891	0.917	0.921
			3	5	0.727	**0.794**	0.762	0.762	0.737	0.774	0.758	0.752
			6	2	0.895	0.871	**0.895**	0.870	0.892	0.869	0.865	0.837
			6	5	0.529	0.517	0.555	0.517	0.515	0.516	0.544	0.493
100	10	Linear	3	2	0.958	**0.985**	0.960	0.984	0.956	0.982	0.928	0.942
			3	5	0.699	**0.882**	0.730	0.854	0.669	0.868	0.720	0.809
			6	2	0.913	0.907	**0.914**	0.909	0.914	0.908	0.870	0.879
			6	5	0.395	0.623	0.530	0.624	0.377	0.628	0.527	0.618
		Multiplicative	3	2	0.952	0.983	0.951	0.983	0.948	**0.984**	0.909	0.942
			3	5	0.665	**0.914**	0.715	0.904	0.625	0.902	0.703	0.816
			6	2	0.904	0.896	0.894	**0.904**	0.896	0.902	0.868	0.878
			6	5	0.420	0.633	0.543	0.629	0.415	**0.641**	0.548	0.612
		Tchebycheff	3	2	**1.000**	0.885	**1.000**	0.887	**1.000**	0.878	0.992	0.942
			3	5	0.999	0.866	**0.999**	0.864	0.998	0.852	0.960	0.811
			6	2	**1.000**	0.894	**1.000**	0.894	**1.000**	0.896	0.989	0.857
			6	5	**0.999**	0.628	0.999	0.632	0.998	0.643	0.957	0.594
		Complex Nonmonotonic	3	2	0.965	0.864	0.964	0.856	0.963	0.859	0.938	0.945
			3	5	0.775	**0.858**	0.817	0.852	0.769	0.853	0.801	0.802
			6	2	0.916	0.908	**0.923**	0.908	0.919	0.906	0.891	0.874
			6	5	0.524	0.608	0.612	**0.619**	0.484	0.617	0.598	0.584
	30	Linear	3	2	0.956	0.986	0.958	**0.986**	0.958	0.984	0.923	0.949
			3	5	0.722	**0.886**	0.751	0.876	0.714	0.883	0.732	0.812
			6	2	0.914	0.911	0.912	0.913	**0.915**	0.913	0.887	0.887
			6	5	0.412	0.642	0.538	**0.660**	0.387	0.646	0.532	0.620
		Multiplicative	3	2	0.953	**0.986**	0.954	0.985	0.951	0.984	0.915	0.948
			3	5	0.697	0.911	0.738	**0.913**	0.676	0.905	0.716	0.805
			6	2	0.906	0.911	0.901	0.911	0.901	**0.913**	0.867	0.888
			6	5	0.453	0.642	0.565	0.656	0.444	0.653	0.557	0.630
		Tchebycheff	3	2	**1.000**	0.936	**1.000**	0.939	**1.000**	0.934	0.988	0.940
			3	5	0.999	0.877	**0.999**	0.870	0.999	0.867	0.965	0.824
			6	2	**1.000**	0.899	**1.000**	0.902	**1.000**	0.898	0.990	0.868
			6	5	**0.999**	0.652	0.999	0.655	0.999	0.653	0.971	0.622

Table A.1 (continued).

# of Queries	Ref Set Size	Underlying Value Function	# of attr.	# of class.	Uncertainty Sampling						Rand RF	Rand SVM
					LC		M		E			
					RF	SVM	RF	SVM	RF	SVM		
100	30	Complex Nonmonotonic	3	2	0.963	0.912	0.965	0.919	0.965	0.915	0.939	0.946
			3	5	0.793	0.871	0.831	0.864	0.795	**0.874**	0.821	0.812
			6	2	0.920	0.915	**0.926**	0.915	0.918	0.914	0.895	0.885
			6	5	0.541	0.607	0.629	0.618	0.527	0.613	0.613	0.597

Table A.2 Detailed Acc results (Query By Bagging).

# of Queries	Ref Set Size	Underlying Value Function	# of attr.	# of class.	Query By Bagging						Rand RF	Rand SVM
					LC		M		E			
					RF	SVM	RF	SVM	RF	SVM		
30	10	Linear	3	2	0.921	0.933	0.917	0.937	0.915	0.944	0.887	0.900
			3	5	0.573	**0.742**	0.617	0.725	0.548	0.738	0.613	0.698
			6	2	0.849	0.806	0.847	0.804	0.844	0.805	0.815	0.805
			6	5	0.392	0.450	0.477	**0.510**	0.399	0.424	0.479	0.476
		Multiplicative	3	2	0.910	0.943	0.903	0.942	0.910	0.947	0.880	0.906
			3	5	0.578	0.768	0.630	0.753	0.558	0.762	0.610	0.707
			6	2	0.843	0.775	0.840	0.806	0.844	0.776	0.809	0.803
			6	5	0.414	0.486	0.467	**0.519**	0.424	0.506	0.465	0.490
		Tchebycheff	3	2	0.999	0.928	0.999	0.926	1.000	0.929	0.976	0.895
			3	5	0.955	0.768	0.937	0.729	0.949	0.759	0.894	0.700
			6	2	0.996	0.795	0.999	0.817	0.999	0.799	0.969	0.786
			6	5	0.935	0.456	0.940	0.452	0.916	0.430	0.875	0.480
		Complex Nonmonotonic	3	2	0.941	0.940	0.942	0.939	0.938	0.935	0.915	0.896
			3	5	0.694	0.743	0.742	0.734	0.694	**0.755**	0.725	0.712
			6	2	0.875	0.809	0.875	0.811	0.860	0.820	0.847	0.801
			6	5	0.496	0.473	0.530	0.470	0.484	0.441	0.530	0.451
	30	Linear	3	2	0.928	0.954	0.931	0.954	0.926	0.955	0.891	0.924
			3	5	0.637	0.793	0.671	0.767	0.625	0.793	0.655	0.725
			6	2	0.863	0.838	0.869	0.842	**0.871**	0.844	0.833	0.840
			6	5	0.418	0.509	0.485	0.529	0.419	0.497	0.490	0.529
		Multiplicative	3	2	0.920	0.953	0.918	0.955	0.917	0.956	0.891	0.923
			3	5	0.622	0.779	0.660	0.775	0.604	0.791	0.662	0.728
			6	2	0.862	0.844	0.857	0.841	0.854	0.854	0.830	0.839
			6	5	0.460	0.555	0.500	0.550	0.445	**0.568**	0.502	0.532

Table A.2 (continued).

# of Queries	Ref Set Size	Underlying Value Function	# of attr.	# of class.	Query By Bagging						Rand RF	Rand SVM
					LC		M		E			
					RF	SVM	RF	SVM	RF	SVM		
30	30	Tchebycheff	3	2	1.000	0.941	1.000	0.938	1.000	0.940	0.980	0.914
			3	5	0.972	0.792	0.970	0.777	0.967	0.797	0.928	0.753
			6	2	1.000	0.830	1.000	0.836	**1.000**	0.813	0.977	0.817
			6	5	0.973	0.503	0.969	0.527	0.960	0.510	0.925	0.520
		Complex Nonmonotonic	3	2	0.945	0.948	0.947	0.945	0.947	0.948	0.917	0.921
			3	5	0.741	0.776	0.765	0.762	0.745	0.784	0.758	0.752
			6	2	0.877	0.842	0.887	0.861	0.885	0.848	0.865	0.837
			6	5	0.532	0.522	**0.561**	0.523	0.529	0.536	0.544	0.493
100	10	Linear	3	2	0.957	0.979	0.959	0.982	0.955	0.982	0.928	0.942
			3	5	0.675	0.881	0.729	0.849	0.662	0.864	0.720	0.809
			6	2	0.906	0.894	0.902	0.890	0.900	0.894	0.870	0.879
			6	5	0.379	0.632	0.517	**0.633**	0.347	0.615	0.527	0.618
		Multiplicative	3	2	0.949	0.981	0.948	0.980	0.945	0.977	0.909	0.942
			3	5	0.647	0.872	0.720	0.859	0.611	0.863	0.703	0.816
			6	2	0.891	0.891	0.897	0.892	0.896	0.884	0.868	0.878
			6	5	0.430	0.612	0.535	0.629	0.426	0.641	0.548	0.612
		Tchebycheff	3	2	**1.000**	0.967	**1.000**	0.968	**1.000**	0.964	0.992	0.942
			3	5	0.999	0.865	0.998	0.861	0.994	0.857	0.960	0.811
			6	2	**1.000**	0.875	**1.000**	0.882	**1.000**	0.869	0.989	0.857
			6	5	0.998	0.618	0.998	0.618	0.993	0.610	0.957	0.594
		Complex Nonmonotonic	3	2	**0.968**	0.960	0.965	0.958	0.966	0.957	0.938	0.945
			3	5	0.759	0.858	0.823	0.851	0.752	0.855	0.801	0.802
			6	2	0.912	0.890	0.915	0.896	0.912	0.895	0.891	0.874
			6	5	0.499	0.585	0.608	0.606	0.482	0.609	0.598	0.584
	30	Linear	3	2	0.957	0.980	0.958	0.983	0.959	0.981	0.923	0.949
			3	5	0.706	0.886	0.742	0.867	0.697	0.881	0.732	0.812
			6	2	0.912	0.896	0.908	0.895	0.912	0.899	0.887	0.887
			6	5	0.400	0.641	0.542	0.638	0.374	0.649	0.532	0.620
		Multiplicative	3	2	0.955	0.980	0.956	0.983	0.953	0.981	0.915	0.948
			3	5	0.685	0.873	0.729	0.871	0.655	0.875	0.716	0.805
			6	2	0.901	0.895	0.901	0.895	0.907	0.894	0.867	0.888
			6	5	0.462	0.659	0.544	0.644	0.450	**0.669**	0.557	0.630
		Tchebycheff	3	2	**1.000**	0.970	**1.000**	0.972	**1.000**	0.971	0.988	0.940
			3	5	0.999	0.876	0.999	0.874	0.996	0.869	0.965	0.824
			6	2	**1.000**	0.879	**1.000**	0.889	**1.000**	0.882	0.990	0.868
			6	5	0.998	0.643	0.998	0.638	0.996	0.652	0.971	0.622
		Complex Nonmonotonic	3	2	0.967	0.964	**0.970**	0.962	0.966	0.965	0.939	0.946
			3	5	0.776	0.870	0.836	0.857	0.788	0.867	0.821	0.812
			6	2	0.917	0.900	0.925	0.901	0.918	0.899	0.895	0.885
			6	5	0.536	0.614	0.620	0.617	0.520	**0.635**	0.613	0.597

Table A.3 Detailed BCA results (Uncertainty Sampling).

# of Queries	Ref Set Size	Underlying Value Function	# of attr.	# of class.	Uncertainty Sampling						Rand RF	Rand SVM
					LC		M		E			
					RF	SVM	RF	SVM	RF	SVM		
30	10	Linear	3	2	0.919	0.948	0.914	**0.949**	0.919	0.945	0.887	0.900
			3	5	0.608	**0.735**	0.616	0.716	0.565	0.730	0.596	0.685
			6	2	0.852	0.851	**0.857**	0.838	0.844	0.841	0.815	0.805
			6	5	0.362	0.423	0.419	0.456	0.361	0.406	0.422	0.445
		Multiplicative	3	2	0.912	0.946	0.909	**0.948**	0.911	0.942	0.880	0.906
			3	5	0.504	0.794	0.547	0.747	0.504	**0.801**	0.512	0.616
			6	2	0.848	0.840	0.847	0.842	**0.852**	0.842	0.809	0.803
			6	5	0.391	0.436	0.438	0.466	0.383	0.482	0.436	0.465
		Tchebycheff	3	2	**1.000**	0.829	**1.000**	0.840	**1.000**	0.820	0.975	0.895
			3	5	**0.965**	0.764	0.952	0.706	0.954	0.752	0.867	0.685
			6	2	**1.000**	0.811	**1.000**	0.826	**1.000**	0.814	0.968	0.786
			6	5	**0.956**	0.458	0.955	0.478	0.938	0.469	0.861	0.463
		Complex Nonmonotonic	3	2	0.944	0.801	0.944	0.806	**0.945**	0.809	0.916	0.895
			3	5	0.688	0.730	0.727	0.709	0.674	0.731	0.709	0.676
			6	2	0.872	0.844	**0.883**	0.839	0.876	0.839	0.847	0.792
			6	5	0.464	0.458	**0.519**	0.464	0.476	0.463	0.498	0.425
	30	Linear	3	2	0.923	0.958	0.932	**0.960**	0.925	0.957	0.891	0.924
			3	5	0.645	**0.797**	0.668	0.764	0.628	0.780	0.635	0.712
			6	2	0.867	0.866	0.871	0.864	0.863	0.863	0.833	0.839
			6	5	0.381	0.473	0.439	0.494	0.371	0.471	0.432	0.489
		Multiplicative	3	2	0.917	**0.960**	0.917	0.958	0.919	0.957	0.891	0.923
			3	5	0.543	**0.841**	0.571	0.792	0.524	0.836	0.548	0.635
			6	2	0.861	0.867	0.858	0.870	0.864	**0.871**	0.830	0.838
			6	5	0.429	0.533	0.471	0.537	0.424	0.529	0.474	0.509
		Tchebycheff	3	2	1.000	0.899	**1.000**	0.896	1.000	0.895	0.980	0.914
			3	5	0.977	0.799	**0.978**	0.763	0.964	0.808	0.912	0.748
			6	2	**1.000**	0.851	1.000	0.847	1.000	0.848	0.976	0.816
			6	5	**0.979**	0.515	0.975	0.510	0.964	0.513	0.913	0.504
		Complex Nonmonotonic	3	2	0.949	0.888	**0.951**	0.892	0.949	0.880	0.915	0.920
			3	5	0.693	**0.787**	0.730	0.728	0.705	0.765	0.724	0.703
			6	2	**0.893**	0.869	0.892	0.868	0.890	0.867	0.861	0.832
			6	5	0.494	0.488	**0.532**	0.493	0.489	0.491	0.526	0.478
100	10	Linear	3	2	0.958	**0.985**	0.960	0.984	0.956	0.982	0.929	0.943
			3	5	0.703	**0.880**	0.724	0.850	0.673	0.869	0.705	0.801
			6	2	0.913	0.907	0.914	0.909	**0.914**	0.908	0.870	0.879
			6	5	0.366	0.576	0.472	0.583	0.343	0.574	0.464	0.572
		Multiplicative	3	2	0.952	0.983	0.951	0.983	0.948	**0.984**	0.909	0.942
			3	5	0.593	0.876	0.636	0.831	0.552	**0.885**	0.596	0.734
			6	2	0.904	0.896	0.895	**0.904**	0.897	0.902	0.868	0.877
			6	5	0.389	0.614	0.511	0.606	0.383	0.631	0.514	0.595
		Tchebycheff	3	2	**1.000**	0.888	**1.000**	0.890	**1.000**	0.882	0.992	0.942
			3	5	0.999	0.860	**0.999**	0.850	0.998	0.851	0.946	0.800
			6	2	**1.000**	0.894	**1.000**	0.894	**1.000**	0.895	0.989	0.857
			6	5	**0.999**	0.621	**0.999**	0.618	0.998	0.629	0.952	0.577
		Complex Nonmonotonic	3	2	0.966	0.847	0.965	0.837	0.965	0.841	0.938	0.945
			3	5	0.757	0.851	0.792	0.834	0.751	0.852	0.781	0.776
			6	2	0.915	0.906	**0.922**	0.906	0.918	0.905	0.891	0.871
			6	5	0.479	0.584	0.582	0.595	0.443	**0.599**	0.558	0.563

Table A.3 (continued).

# of Queries	Ref Set Size	Underlying Value Function	# of attr.	# of class.	Uncertainty Sampling						Rand RF	Rand SVM
					LC		M		E			
					RF	SVM	RF	SVM	RF	SVM		
100	30	Linear	3	2	0.956	0.986	0.958	**0.986**	0.957	0.984	0.923	0.949
			3	5	0.726	0.884	0.742	0.873	0.716	0.881	0.718	0.803
			6	2	0.914	0.911	0.912	0.913	**0.915**	0.913	0.887	0.886
			6	5	0.377	0.593	0.485	**0.617**	0.351	0.595	0.465	0.576
		Multiplicative	3	2	0.952	**0.986**	0.954	0.985	0.952	0.984	0.915	0.948
			3	5	0.625	0.874	0.644	0.857	0.596	**0.880**	0.609	0.739
			6	2	0.906	0.911	0.901	0.911	0.901	**0.913**	0.867	0.888
			6	5	0.416	0.628	0.533	0.634	0.408	0.639	0.530	0.608
		Tchebycheff	3	2	**1.000**	0.937	**1.000**	0.941	**1.000**	0.935	0.989	0.940
			3	5	0.999	0.876	**0.999**	0.865	0.999	0.866	0.956	0.813
			6	2	**1.000**	0.899	**1.000**	0.902	**1.000**	0.898	0.990	0.867
			6	5	**0.999**	0.644	**0.999**	0.640	0.999	0.643	0.968	0.607
		Complex Nonmonotonic	3	2	0.964	0.902	0.966	0.910	0.966	0.906	0.937	0.945
			3	5	0.764	0.869	0.797	0.848	0.773	**0.872**	0.794	0.783
			6	2	0.919	0.913	0.924	0.914	0.917	0.913	0.892	0.883
			6	5	0.496	0.579	0.595	0.594	0.489	0.590	0.584	0.577

Table A.4 Detailed BCA results (Query By Bagging).

# of Queries	Ref Set Size	Underlying Value Function	# of attr.	# of class.	Query By Bagging						Rand RF	Rand SVM
					LC		M		E			
					RF	SVM	RF	SVM	RF	SVM		
30	10	Linear	3	2	0.922	0.933	0.917	0.937	0.914	0.944	0.887	0.900
			3	5	0.567	0.735	0.609	0.715	0.546	0.728	0.596	0.685
			6	2	0.849	0.806	0.846	0.805	0.844	0.805	0.815	0.805
			6	5	0.361	0.414	0.423	**0.463**	0.369	0.394	0.422	0.445
		Multiplicative	3	2	0.910	0.943	0.903	0.942	0.910	0.947	0.880	0.906
			3	5	0.500	0.680	0.531	0.671	0.488	0.667	0.512	0.616
			6	2	0.843	0.776	0.841	0.806	0.844	0.775	0.809	0.803
			6	5	0.371	0.447	0.434	**0.493**	0.383	0.464	0.436	0.465
		Tchebycheff	3	2	0.999	0.928	0.999	0.925	1.000	0.929	0.975	0.895
			3	5	0.942	0.758	0.925	0.711	0.932	0.747	0.867	0.685
			6	2	0.996	0.792	0.999	0.815	0.999	0.798	0.968	0.786
			6	5	0.930	0.451	0.935	0.445	0.910	0.432	0.861	0.463
		Complex Nonmonotonic	3	2	0.941	0.939	0.942	0.939	0.939	0.935	0.916	0.895
			3	5	0.669	0.727	0.719	0.702	0.665	**0.735**	0.709	0.676
			6	2	0.873	0.805	0.875	0.806	0.860	0.816	0.847	0.792
			6	5	0.452	0.440	0.496	0.447	0.449	0.411	0.498	0.425
	30	Linear	3	2	0.928	0.954	0.931	0.954	0.926	0.955	0.891	0.924
			3	5	0.627	0.786	0.653	0.756	0.616	0.783	0.635	0.712
			6	2	0.863	0.838	0.870	0.842	**0.871**	0.844	0.833	0.839
			6	5	0.381	0.484	0.432	**0.495**	0.379	0.465	0.432	0.489

Table A.4 (continued).

# of Queries	Ref Set Size	Underlying Value Function	# of attr.	# of class.	Query By Bagging						Rand RF	Rand SVM
					LC		M		E			
					RF	SVM	RF	SVM	RF	SVM		
30	30	Multiplicative	3	2	0.920	0.953	0.918	0.956	0.917	0.956	0.891	0.923
			3	5	0.538	0.703	0.565	0.693	0.520	0.701	0.548	0.635
			6	2	0.862	0.843	0.857	0.840	0.854	0.854	0.830	0.838
			6	5	0.418	0.533	0.474	0.529	0.413	**0.545**	0.474	0.509
		Tchebycheff	3	2	1.000	0.941	1.000	0.938	1.000	0.940	0.980	0.914
			3	5	0.968	0.788	0.966	0.770	0.959	0.796	0.912	0.748
			6	2	1.000	0.829	1.000	0.836	1.000	0.811	0.976	0.816
			6	5	0.971	0.499	0.967	0.512	0.959	0.505	0.913	0.504
		Complex Nonmonotonic	3	2	0.944	0.948	0.948	0.945	0.947	0.949	0.915	0.920
			3	5	0.711	0.753	0.734	0.725	0.711	0.770	0.724	0.703
			6	2	0.875	0.838	0.885	0.859	0.882	0.843	0.861	0.832
			6	5	0.504	0.496	0.530	0.501	0.496	0.512	0.526	0.478
100	10	Linear	3	2	0.957	0.979	0.959	0.982	0.955	0.982	0.929	0.943
			3	5	0.678	0.880	0.723	0.844	0.665	0.865	0.705	0.801
			6	2	0.906	0.894	0.902	0.890	0.900	0.894	0.870	0.879
			6	5	0.358	0.579	0.469	**0.592**	0.329	0.564	0.464	0.572
		Multiplicative	3	2	0.949	0.981	0.948	0.981	0.945	0.977	0.909	0.942
			3	5	0.582	0.876	0.630	0.819	0.547	0.859	0.596	0.734
			6	2	0.891	0.891	0.897	0.892	0.896	0.884	0.868	0.877
			6	5	0.392	0.595	0.498	0.607	0.375	**0.631**	0.514	0.595
		Tchebycheff	3	2	**1.000**	0.968	**1.000**	0.968	**1.000**	0.964	0.992	0.942
			3	5	0.998	0.861	0.998	0.851	0.993	0.848	0.946	0.800
			6	2	**1.000**	0.875	**1.000**	0.882	**1.000**	0.870	0.989	0.857
			6	5	0.997	0.613	0.998	0.605	0.992	0.607	0.952	0.577
		Complex Nonmonotonic	3	2	**0.968**	0.961	0.965	0.959	0.966	0.959	0.938	0.945
			3	5	0.743	**0.857**	0.803	0.833	0.737	0.855	0.781	0.776
			6	2	0.911	0.890	0.914	0.894	0.913	0.893	0.891	0.871
			6	5	0.448	0.547	0.561	0.582	0.434	0.575	0.558	0.563
	30	Linear	3	2	0.957	0.980	0.958	0.983	0.959	0.981	0.923	0.949
			3	5	0.709	**0.885**	0.736	0.862	0.701	0.881	0.718	0.803
			6	2	0.912	0.896	0.909	0.895	0.912	0.899	0.887	0.886
			6	5	0.372	0.596	0.484	0.599	0.345	0.599	0.465	0.576
		Multiplicative	3	2	0.955	0.980	0.956	0.983	0.953	0.981	0.915	0.948
			3	5	0.607	0.880	0.639	0.827	0.582	0.870	0.609	0.739
			6	2	0.901	0.895	0.901	0.895	0.907	0.894	0.867	0.888
			6	5	0.420	0.653	0.516	0.622	0.396	**0.658**	0.530	0.608
		Tchebycheff	3	2	**1.000**	0.970	**1.000**	0.972	**1.000**	0.971	0.989	0.940
			3	5	0.999	0.876	0.999	0.867	0.996	0.864	0.956	0.813
			6	2	**1.000**	0.879	**1.000**	0.889	**1.000**	0.882	0.990	0.867
			6	5	0.998	0.638	0.998	0.625	0.996	0.645	0.968	0.607
		Complex Nonmonotonic	3	2	0.968	0.965	**0.970**	0.964	0.966	0.966	0.937	0.945
			3	5	0.761	0.871	0.806	0.837	0.774	0.868	0.794	0.783
			6	2	0.917	0.898	**0.924**	0.899	0.919	0.898	0.892	0.883
			6	5	0.491	0.586	0.587	0.591	0.477	**0.606**	0.584	0.577

Table A.5 Detailed Kappa results (Uncertainty Sampling).

# of Queries	Ref Set Size	Underlying Value Function	# of attr.	# of class.	Uncertainty Sampling						Rand RF	Rand SVM
					LC		M		E			
					RF	SVM	RF	SVM	RF	SVM		
30	10	Linear	3	2	0.838	0.897	0.828	**0.897**	0.836	0.891	0.773	0.800
			3	5	0.521	0.668	0.532	0.652	0.463	0.670	0.514	0.621
			6	2	0.704	0.702	**0.713**	0.677	0.687	0.681	0.629	0.610
			6	5	0.234	0.322	0.322	0.362	0.231	0.282	0.329	0.337
		Multiplicative	3	2	0.824	0.892	0.818	**0.896**	0.822	0.884	0.759	0.812
			3	5	0.461	**0.777**	0.531	0.724	0.456	0.748	0.487	0.614
			6	2	0.696	0.681	0.693	0.684	**0.704**	0.684	0.619	0.606
			6	5	0.244	0.313	0.304	0.342	0.236	0.357	0.303	0.338
		Tchebycheff	3	2	1.000	0.651	**1.000**	0.673	**1.000**	0.633	0.951	0.790
			3	5	**0.951**	0.709	0.932	0.645	0.947	0.686	0.863	0.613
			6	2	**1.000**	0.623	0.999	0.652	1.000	0.627	0.937	0.571
			6	5	**0.947**	0.329	0.945	0.358	0.929	0.342	0.842	0.343
		Complex Nonmonotonic	3	2	0.884	0.622	0.886	0.629	**0.887**	0.636	0.828	0.789
			3	5	0.621	0.670	0.674	0.657	0.607	0.675	0.647	0.626
			6	2	0.749	0.688	**0.766**	0.680	0.756	0.676	0.692	0.590
			6	5	0.357	0.337	**0.421**	0.350	0.361	0.340	0.400	0.299
	30	Linear	3	2	0.846	0.916	0.864	**0.920**	0.849	0.914	0.782	0.849
			3	5	0.567	**0.746**	0.602	0.715	0.547	0.732	0.567	0.655
			6	2	0.734	0.733	0.742	0.728	0.726	0.725	0.666	0.679
			6	5	0.267	0.384	0.349	**0.408**	0.252	0.377	0.343	0.402
		Multiplicative	3	2	0.834	**0.920**	0.834	0.915	0.838	0.914	0.781	0.846
			3	5	0.520	**0.805**	0.573	0.759	0.501	0.787	0.552	0.640
			6	2	0.722	0.735	0.717	0.740	0.728	**0.742**	0.660	0.676
			6	5	0.296	0.429	0.345	0.433	0.284	0.421	0.352	0.394
		Tchebycheff	3	2	1.000	0.794	**1.000**	0.787	1.000	0.786	0.959	0.827
			3	5	**0.969**	0.740	0.969	0.708	0.961	0.759	0.907	0.682
			6	2	**1.000**	0.702	0.999	0.695	1.000	0.697	0.953	0.633
			6	5	**0.975**	0.398	0.970	0.397	0.958	0.399	0.905	0.394
		Complex Nonmonotonic	3	2	0.896	0.785	**0.898**	0.793	0.895	0.772	0.831	0.840
			3	5	0.643	**0.734**	0.689	0.691	0.657	0.708	0.684	0.676
			6	2	0.787	0.739	**0.787**	0.736	0.781	0.736	0.726	0.668
			6	5	0.396	0.382	0.434	0.384	0.382	0.382	0.422	0.358
100	10	Linear	3	2	0.915	**0.969**	0.919	0.967	0.911	0.965	0.856	0.885
			3	5	0.626	**0.852**	0.662	0.817	0.589	0.835	0.648	0.760
			6	2	0.825	0.814	0.827	0.818	**0.827**	0.816	0.740	0.757
			6	5	0.235	0.521	0.397	0.522	0.207	0.524	0.388	0.512
		Multiplicative	3	2	0.905	0.967	0.902	0.965	0.897	**0.968**	0.817	0.884
			3	5	0.566	**0.870**	0.626	0.855	0.517	0.854	0.606	0.755
			6	2	0.807	0.792	0.788	**0.808**	0.792	0.804	0.736	0.755
			6	5	0.245	0.526	0.403	0.519	0.240	0.539	0.409	0.499
		Tchebycheff	3	2	**1.000**	0.771	**1.000**	0.776	**1.000**	0.759	0.984	0.883
			3	5	0.999	0.827	**0.999**	0.824	0.997	0.810	0.948	0.755
			6	2	**1.000**	0.787	**1.000**	0.788	**1.000**	0.791	0.978	0.713
			6	5	**0.999**	0.533	**0.999**	0.537	0.997	0.551	0.945	0.488
		Complex Nonmonotonic	3	2	0.930	0.712	0.928	0.693	0.926	0.701	0.874	0.887
			3	5	0.710	**0.816**	0.763	0.808	0.701	0.811	0.743	0.743
			6	2	0.831	0.813	**0.844**	0.813	0.837	0.809	0.780	0.744
			6	5	0.386	0.500	0.504	**0.515**	0.336	0.514	0.484	0.471

Table A.5 (continued).

# of Queries	Ref Set Size	Underlying Value Function	# of attr.	# of class.	Uncertainty Sampling						Rand RF	Rand SVM
					LC		M		E			
					RF	SVM	RF	SVM	RF	SVM		
100	30	Linear	3	2	0.911	0.971	0.917	**0.972**	0.915	0.969	0.846	0.897
			3	5	0.654	**0.858**	0.688	0.845	0.644	0.853	0.663	0.764
			6	2	0.828	0.823	0.825	0.825	**0.829**	0.825	0.774	0.773
			6	5	0.253	0.542	0.408	**0.566**	0.219	0.547	0.394	0.515
		Multiplicative	3	2	0.905	**0.971**	0.907	0.971	0.903	0.968	0.831	0.896
			3	5	0.606	0.866	0.653	**0.868**	0.579	0.857	0.623	0.743
			6	2	0.812	0.822	0.801	0.823	0.802	**0.826**	0.735	0.776
			6	5	0.285	0.540	0.432	0.555	0.274	0.554	0.422	0.523
		Tchebycheff	3	2	**1.000**	0.872	**1.000**	0.879	**1.000**	0.867	0.977	0.880
			3	5	0.999	0.842	**0.999**	0.833	0.998	0.829	0.954	0.773
			6	2	**1.000**	0.798	**1.000**	0.804	**1.000**	0.796	0.980	0.735
			6	5	**0.999**	0.563	0.999	0.566	0.998	0.563	0.964	0.523
		Complex Nonmonotonic	3	2	0.925	0.816	0.929	0.831	0.929	0.823	0.876	0.889
			3	5	0.731	0.834	0.779	0.824	0.734	**0.837**	0.768	0.756
			6	2	0.838	0.827	**0.849**	0.828	0.833	0.826	0.787	0.767
			6	5	0.409	0.499	0.525	0.514	0.393	0.507	0.506	0.489

Table A.6 Detailed Kappa results (Query By Bagging).

# of Queries	Ref Set Size	Underlying Value Function	# of attr.	# of class.	Query By Bagging						Rand RF	Rand SVM
					LC		M		E			
					RF	SVM	RF	SVM	RF	SVM		
30	10	Linear	3	2	0.843	0.866	0.835	0.874	0.829	0.887	0.773	0.800
			3	5	0.467	**0.677**	0.522	0.656	0.438	0.672	0.514	0.621
			6	2	0.698	0.611	0.693	0.609	0.688	0.610	0.629	0.610
			6	5	0.229	0.299	0.328	**0.374**	0.239	0.269	0.329	0.337
		Multiplicative	3	2	0.819	0.886	0.806	0.885	0.819	0.894	0.759	0.812
			3	5	0.454	0.694	0.514	0.675	0.430	0.686	0.487	0.614
			6	2	0.686	0.551	0.680	0.612	0.688	0.551	0.619	0.606
			6	5	0.230	0.323	0.301	**0.373**	0.244	0.350	0.303	0.338
		Tchebycheff	3	2	0.998	0.855	0.998	0.851	0.999	0.858	0.951	0.790
			3	5	0.942	0.701	0.919	0.649	0.934	0.688	0.863	0.613
			6	2	0.993	0.584	0.998	0.631	0.997	0.596	0.937	0.571
			6	5	0.918	0.318	0.924	0.311	0.894	0.288	0.842	0.343
		Complex Nonmonotonic	3	2	0.881	0.877	0.882	0.876	0.875	0.867	0.828	0.789
			3	5	0.604	0.667	0.667	0.654	0.603	**0.684**	0.647	0.626
			6	2	0.746	0.613	0.748	0.614	0.717	0.633	0.692	0.590
			6	5	0.351	0.322	0.399	0.325	0.341	0.285	0.400	0.299
	30	Linear	3	2	0.855	0.907	0.861	0.909	0.851	0.909	0.782	0.849
			3	5	0.546	0.740	0.587	0.708	0.532	0.740	0.567	0.655
			6	2	0.726	0.677	0.739	0.683	**0.743**	0.688	0.666	0.679
			6	5	0.259	0.382	0.340	0.404	0.261	0.365	0.343	0.402

Table A.6 (continued).

# of Queries	Ref Set Size	Underlying Value Function	# of attr.	# of class.	Query By Bagging						Rand RF	Rand SVM
					LC		M		E			
					RF	SVM	RF	SVM	RF	SVM		
30	30	Multiplicative	3	2	0.840	0.907	0.835	0.911	0.833	0.912	0.781	0.846
			3	5	0.509	0.709	0.554	0.703	0.487	0.723	0.552	0.640
			6	2	0.723	0.686	0.713	0.681	0.708	0.709	0.660	0.676
			6	5	0.289	0.425	0.351	0.419	0.278	**0.439**	0.352	0.394
		Tchebycheff	3	2	0.999	0.881	0.999	0.876	0.999	0.879	0.959	0.827
			3	5	0.964	0.732	0.961	0.712	0.957	0.738	0.907	0.682
			6	2	0.999	0.658	1.000	0.671	**1.000**	0.622	0.953	0.633
			6	5	0.966	0.377	0.961	0.403	0.950	0.385	0.905	0.394
		Complex Nonmonotonic	3	2	0.887	0.893	0.893	0.888	0.891	0.895	0.831	0.840
			3	5	0.663	0.708	0.695	0.690	0.668	0.720	0.684	0.676
			6	2	0.751	0.678	0.771	0.719	0.767	0.690	0.726	0.668
			6	5	0.403	0.391	**0.439**	0.393	0.398	0.409	0.422	0.358
100	10	Linear	3	2	0.913	0.957	0.918	0.964	0.911	0.964	0.856	0.885
			3	5	0.596	0.852	0.661	0.811	0.579	0.830	0.648	0.760
			6	2	0.811	0.789	0.804	0.780	0.801	0.787	0.740	0.757
			6	5	0.219	0.528	0.383	**0.533**	0.180	0.508	0.388	0.512
		Multiplicative	3	2	0.899	0.962	0.896	0.961	0.891	0.954	0.817	0.884
			3	5	0.545	0.835	0.632	0.814	0.502	0.822	0.606	0.755
			6	2	0.782	0.782	0.794	0.784	0.791	0.768	0.736	0.755
			6	5	0.254	0.498	0.390	0.519	0.242	**0.539**	0.409	0.499
		Tchebycheff	3	2	**1.000**	0.935	**1.000**	0.936	**1.000**	0.928	0.984	0.883
			3	5	0.998	0.826	0.998	0.821	0.992	0.815	0.948	0.755
			6	2	**1.000**	0.749	**1.000**	0.763	**1.000**	0.739	0.978	0.713
			6	5	0.997	0.521	0.997	0.518	0.991	0.511	0.945	0.488
		Complex Nonmonotonic	3	2	**0.934**	0.919	0.928	0.915	0.930	0.913	0.874	0.887
			3	5	0.688	0.816	0.771	0.807	0.680	0.813	0.743	0.743
			6	2	0.822	0.778	0.828	0.789	0.822	0.787	0.780	0.744
			6	5	0.352	0.467	0.495	0.499	0.330	0.499	0.484	0.471
	30	Linear	3	2	0.914	0.960	0.916	0.966	0.917	0.962	0.846	0.897
			3	5	0.634	0.857	0.678	0.833	0.623	0.851	0.663	0.764
			6	2	0.823	0.791	0.817	0.790	0.824	0.798	0.774	0.773
			6	5	0.242	0.542	0.412	0.540	0.207	0.551	0.394	0.515
		Multiplicative	3	2	0.910	0.960	0.911	0.966	0.905	0.962	0.831	0.896
			3	5	0.590	0.836	0.643	0.830	0.553	0.838	0.623	0.743
			6	2	0.801	0.790	0.802	0.789	0.814	0.787	0.735	0.776
			6	5	0.293	0.565	0.406	0.540	0.271	**0.576**	0.422	0.523
		Tchebycheff	3	2	**1.000**	0.940	**1.000**	0.944	**1.000**	0.942	0.977	0.880
			3	5	0.998	0.841	0.999	0.838	0.995	0.832	0.954	0.773
			6	2	**1.000**	0.758	**1.000**	0.778	**1.000**	0.764	0.980	0.735
			6	5	0.998	0.551	0.997	0.543	0.995	0.561	0.964	0.523
		Complex Nonmonotonic	3	2	0.933	0.926	**0.938**	0.924	0.931	0.928	0.876	0.889
			3	5	0.710	0.832	0.786	0.815	0.725	0.828	0.768	0.756
			6	2	0.833	0.798	0.848	0.799	0.835	0.797	0.787	0.767
			6	5	0.401	0.507	0.514	0.512	0.383	**0.533**	0.506	0.489

Table A.7 Detailed MAEO results (Uncertainty Sampling).

# of Queries	Ref Set Size	Underlying Value Function	# of attr.	# of class.	Uncertainty Sampling						Rand RF	Rand SVM
					LC		M		E			
					RF	SVM	RF	SVM	RF	SVM		
30	10	Linear	3	5	0.412	0.267	0.402	0.285	0.463	0.268	0.439	0.314
			6	5	0.882	0.724	0.749	0.641	0.915	0.774	0.764	0.691
		Multiplicative	3	5	0.529	**0.150**	0.436	0.185	0.535	0.171	0.488	0.336
			6	5	0.728	0.628	0.666	0.592	0.745	0.591	0.685	0.601
		Tchebycheff	3	5	**0.038**	0.226	0.053	0.279	0.041	0.244	0.108	0.308
			6	5	**0.042**	0.660	0.044	0.619	0.056	0.648	0.131	0.674
		Complex Nonmonotonic	3	5	0.298	0.256	0.263	0.273	0.313	0.253	0.287	0.305
			6	5	0.597	0.625	**0.544**	0.625	0.595	0.640	0.571	0.684
	30	Linear	3	5	0.360	**0.204**	0.334	0.229	0.376	0.214	0.375	0.285
			6	5	0.828	0.598	0.694	**0.568**	0.860	0.609	0.734	0.586
		Multiplicative	3	5	0.448	**0.132**	0.388	0.162	0.483	0.145	0.417	0.302
			6	5	0.658	0.493	0.616	0.504	0.690	0.506	0.629	0.536
		Tchebycheff	3	5	**0.024**	0.202	0.024	0.227	0.030	0.187	0.072	0.252
			6	5	**0.019**	0.546	0.024	0.558	0.033	0.547	0.076	0.584
		Complex Nonmonotonic	3	5	0.283	**0.207**	0.248	0.243	0.273	0.227	0.255	0.260
			6	5	0.544	0.564	**0.515**	0.568	0.574	0.558	0.548	0.614
100	10	Linear	3	5	0.302	**0.118**	0.278	0.146	0.333	0.132	0.287	0.193
			6	5	0.837	0.423	0.606	0.420	0.898	0.430	0.637	0.439
		Multiplicative	3	5	0.383	**0.086**	0.322	0.096	0.434	0.098	0.342	0.191
			6	5	0.645	0.379	0.511	0.393	0.663	**0.375**	0.525	0.426
		Tchebycheff	3	5	0.001	0.134	**0.001**	0.137	0.002	0.148	0.040	0.190
			6	5	0.001	0.391	**0.001**	0.398	0.002	0.376	0.043	0.458
		Complex Nonmonotonic	3	5	0.226	0.142	0.189	0.149	0.232	0.147	0.207	0.202
			6	5	0.517	0.414	0.432	0.418	0.571	**0.409**	0.465	0.469
	30	Linear	3	5	0.280	**0.114**	0.256	0.124	0.287	0.117	0.275	0.189
			6	5	0.802	0.396	0.585	**0.378**	0.862	0.396	0.627	0.430
		Multiplicative	3	5	0.337	0.089	0.295	**0.088**	0.368	0.095	0.325	0.201
			6	5	0.603	0.367	0.486	0.361	0.628	0.358	0.522	0.402
		Tchebycheff	3	5	0.001	0.123	**0.001**	0.130	0.001	0.133	0.035	0.176
			6	5	**0.001**	0.358	0.001	0.366	0.001	0.358	0.029	0.423
		Complex Nonmonotonic	3	5	0.211	0.129	0.176	0.136	0.205	**0.126**	0.186	0.192
			6	5	0.496	0.414	0.414	0.415	0.518	0.411	0.448	0.451

Table A.8 Detailed MAEO results (Query By Bagging).

# of Queries	Ref Set Size	Underlying Value Function	# of attr.	# of class.	Query By Bagging						Rand RF	Rand SVM
					LC		M		E			
					RF	SVM	RF	SVM	RF	SVM		
30	10	Linear	3	5	0.455	**0.260**	0.406	0.280	0.495	0.264	0.439	0.314
			6	5	0.892	0.733	0.740	**0.641**	0.882	0.786	0.764	0.691
		Multiplicative	3	5	0.534	0.249	0.453	0.266	0.565	0.252	0.488	0.336
			6	5	0.721	0.607	0.656	**0.563**	0.705	0.592	0.685	0.601
		Tchebycheff	3	5	0.045	0.233	0.063	0.276	0.051	0.244	0.108	0.308
			6	5	0.065	0.665	0.060	0.679	0.085	0.715	0.131	0.674
		Complex Nonmonotonic	3	5	0.314	0.260	0.269	0.274	0.316	**0.247**	0.287	0.305
			6	5	0.599	0.665	0.555	0.642	0.605	0.706	0.571	0.684
	30	Linear	3	5	0.378	0.208	0.345	0.236	0.394	0.207	0.375	0.285
			6	5	0.835	0.603	0.715	0.577	0.839	0.631	0.734	0.586
		Multiplicative	3	5	0.462	0.233	0.407	0.240	0.488	0.220	0.417	0.302
			6	5	0.667	0.502	0.609	0.514	0.688	**0.486**	0.629	0.536
		Tchebycheff	3	5	0.028	0.208	0.030	0.224	0.033	0.203	0.072	0.252
			6	5	0.027	0.574	0.031	0.554	0.040	0.572	0.076	0.584
		Complex Nonmonotonic	3	5	0.264	0.225	0.244	0.244	0.261	0.218	0.255	0.260
			6	5	0.538	0.561	0.516	0.563	0.546	0.546	0.548	0.614
100	10	Linear	3	5	0.327	0.119	0.273	0.151	0.342	0.136	0.287	0.193
			6	5	0.848	0.427	0.617	**0.410**	0.899	0.454	0.637	0.439
		Multiplicative	3	5	0.406	0.128	0.312	0.142	0.451	0.137	0.342	0.191
			6	5	0.626	0.403	0.519	0.395	0.644	0.379	0.525	0.426
		Tchebycheff	3	5	0.001	0.135	0.002	0.139	0.006	0.143	0.040	0.190
			6	5	0.002	0.403	0.002	0.414	0.007	0.418	0.043	0.458
		Complex Nonmonotonic	3	5	0.242	**0.142**	0.180	0.149	0.249	0.145	0.207	0.202
			6	5	0.542	0.449	0.434	0.430	0.568	0.426	0.465	0.469
	30	Linear	3	5	0.295	0.114	0.261	0.133	0.305	0.119	0.275	0.189
			6	5	0.798	0.401	0.582	0.399	0.854	0.396	0.627	0.430
		Multiplicative	3	5	0.360	0.127	0.304	0.131	0.392	0.125	0.325	0.201
			6	5	0.594	0.353	0.514	0.377	0.625	**0.344**	0.522	0.402
		Tchebycheff	3	5	0.002	0.124	0.001	0.126	0.004	0.131	0.035	0.176
			6	5	0.002	0.370	0.002	0.387	0.004	0.365	0.029	0.423
		Complex Nonmonotonic	3	5	0.224	0.130	0.167	0.143	0.213	0.133	0.186	0.192
			6	5	0.501	0.407	0.422	0.419	0.525	**0.392**	0.448	0.451

Table A.9 Detailed MSEO results (Uncertainty Sampling).

# of Queries	Ref Set Size	Underlying Value Function	# of attr.	# of class.	Uncertainty Sampling						Rand RF	Rand SVM
					LC		M		E			
					RF	SVM	RF	SVM	RF	SVM		
30	10	Linear	3	5	27.609	18.433	23.312	15.334	39.215	14.273	25.938	14.711
			6	5	100.615	89.272	56.307	**42.610**	97.559	109.418	55.489	54.070
		Multiplicative	3	5	44.443	**6.434**	22.988	7.022	43.515	8.579	28.065	13.613
			6	5	63.508	56.321	44.986	41.469	62.800	45.808	46.955	40.503
		Tchebycheff	3	5	1.462	10.829	2.758	14.356	1.067	13.921	3.959	16.994
			6	5	1.304	59.725	**1.066**	39.633	1.896	53.119	6.443	42.210
		Complex Nonmonotonic	3	5	16.392	16.901	11.983	12.147	18.474	13.695	13.176	13.285
			6	5	45.855	49.840	**32.392**	40.101	43.631	50.707	35.237	51.121
	30	Linear	3	5	20.744	8.449	16.325	8.495	22.963	8.455	17.403	12.016
			6	5	79.538	43.730	43.607	32.155	92.741	44.469	48.860	**31.026**
		Multiplicative	3	5	27.089	**4.834**	17.037	5.353	38.424	6.346	18.507	11.620
			6	5	46.831	27.107	40.886	**25.441**	56.058	30.662	41.058	32.836
		Tchebycheff	3	5	0.466	9.214	0.617	9.654	0.637	7.405	1.611	10.555
			6	5	**0.179**	41.648	0.304	35.766	0.699	39.147	2.030	31.183
		Complex Nonmonotonic	3	5	14.799	**8.590**	10.455	9.070	13.010	12.137	10.279	10.160
			6	5	38.108	36.775	29.112	33.040	38.010	38.801	31.646	41.061
100	10	Linear	3	5	18.815	2.793	13.819	3.610	24.481	4.276	11.899	5.972
			6	5	108.532	26.924	37.866	18.944	105.425	25.505	37.770	18.213
		Multiplicative	3	5	22.617	**1.600**	13.668	1.909	30.886	2.263	12.564	5.275
			6	5	62.656	22.805	30.233	**19.816**	66.847	20.429	29.188	20.495
		Tchebycheff	3	5	0.002	3.807	**0.001**	3.172	0.004	5.125	0.790	5.981
			6	5	0.002	22.756	**0.002**	17.990	0.005	20.130	0.912	19.993
		Complex Nonmonotonic	3	5	11.027	**3.737**	6.805	3.864	12.411	4.620	6.477	6.478
			6	5	45.381	26.740	21.292	**19.076**	53.619	23.478	23.758	24.014
	30	Linear	3	5	16.412	**2.385**	10.094	2.571	17.067	2.825	10.031	5.253
			6	5	98.434	20.992	33.012	**13.926**	109.800	18.927	38.110	18.069
		Multiplicative	3	5	16.407	1.797	10.266	**1.531**	21.481	2.300	11.026	5.878
			6	5	55.679	20.695	27.395	**15.255**	60.483	18.555	29.235	17.605
		Tchebycheff	3	5	0.002	3.069	**0.001**	2.912	0.002	4.079	0.483	4.936
			6	5	**0.001**	20.250	0.002	16.134	0.003	21.171	0.339	18.270
		Complex Nonmonotonic	3	5	9.556	3.135	5.682	3.234	9.130	**2.999**	5.445	5.902
			6	5	40.902	27.585	19.215	**19.039**	43.023	24.214	20.785	20.515

Table A.10 Detailed MSEO results (Query By Bagging).

# of Queries	Ref Set Size	Underlying Value Function	# of attr.	# of class.	Query By Bagging						Rand RF	Rand SVM
					LC		M		E			
					RF	SVM	RF	SVM	RF	SVM		
30	10	Linear	3	5	33.892	12.455	25.339	13.464	41.485	**12.144**	25.938	14.711
			6	5	83.861	94.737	52.559	43.460	85.011	111.483	55.489	54.070
		Multiplicative	3	5	42.625	9.451	24.256	9.835	49.257	10.901	28.065	13.613
			6	5	63.344	50.465	43.353	**38.004**	59.429	46.931	46.955	40.503
		Tchebycheff	3	5	**0.947**	10.718	1.898	14.726	1.015	11.709	3.959	16.994
			6	5	2.880	57.380	1.587	51.956	5.185	73.400	6.443	42.210
		Complex Nonmonotonic	3	5	18.668	14.502	12.106	12.894	19.698	**11.396**	13.176	13.285
			6	5	47.885	60.476	36.034	47.025	46.582	63.947	35.237	51.121
	30	Linear	3	5	23.019	8.020	17.060	8.928	25.890	**7.313**	17.403	12.016
			6	5	79.057	45.831	48.118	33.649	78.214	54.704	48.860	**31.026**
		Multiplicative	3	5	34.287	8.095	19.204	7.839	32.999	7.556	18.507	11.620
			6	5	53.058	30.372	40.322	30.130	52.438	28.905	41.058	32.836
		Tchebycheff	3	5	**0.289**	8.290	0.335	8.720	0.439	8.214	1.611	10.555
			6	5	0.295	44.899	0.433	31.893	0.796	43.531	2.030	31.183
		Complex Nonmonotonic	3	5	12.862	10.455	9.964	9.899	12.784	9.231	10.279	10.160
			6	5	33.939	37.703	**27.579**	33.984	34.456	35.547	31.646	41.061
100	10	Linear	3	5	24.515	**2.608**	12.163	3.833	26.146	3.942	11.899	5.972
			6	5	104.872	24.055	34.730	**16.849**	117.036	28.833	37.770	18.213
		Multiplicative	3	5	28.561	3.872	11.424	3.285	36.380	4.349	12.564	5.275
			6	5	63.487	25.547	32.767	20.298	66.317	21.125	29.188	20.495
		Tchebycheff	3	5	0.002	3.591	0.002	3.483	0.017	4.257	0.790	5.981
			6	5	0.005	25.166	0.005	20.252	0.029	27.207	0.912	19.993
		Complex Nonmonotonic	3	5	13.565	4.363	5.779	3.928	14.590	4.528	6.477	6.478
			6	5	54.308	31.336	22.892	21.171	58.625	24.812	23.758	24.014
	30	Linear	3	5	18.917	2.610	11.073	2.997	19.577	2.613	10.031	5.253
			6	5	88.177	19.576	32.624	16.308	114.588	20.124	38.110	18.069
		Multiplicative	3	5	19.289	4.001	11.636	2.631	25.274	3.564	11.026	5.878
			6	5	53.887	17.326	29.931	17.238	56.675	16.108	29.235	17.605
		Tchebycheff	3	5	0.003	3.259	0.001	2.785	0.010	3.561	0.483	4.936
			6	5	0.003	21.927	0.004	17.161	0.008	19.580	0.339	18.270
		Complex Nonmonotonic	3	5	11.841	3.422	4.779	3.265	9.881	3.841	5.445	5.902
			6	5	41.064	26.793	19.356	20.218	44.380	20.446	20.785	20.515

APPENDIX B

MEAN PERFORMANCE MEASURES FOR DIFFERENT REFERENCE AND QUERY SIZES

Table B.1 Mean Acc results for different reference set and query sizes (Uncertainty Sampling).

# of Queries	Ref. Set Size	Underlying Value Function	Uncertainty Sampling						Rand RF	Rand SVM
			LC		M		E			
			RF	SVM	RF	SVM	RF	SVM		
30	10	Linear	0.696	0.751	0.717	**0.752**	0.682	0.738	0.698	0.720
		Multiplicative	0.691	0.778	0.715	0.775	0.688	**0.780**	0.691	0.727
		Tchebycheff	**0.980**	0.719	0.976	0.719	0.976	0.715	0.928	0.715
		Complex Nonmonotonic	0.756	0.723	**0.780**	0.723	0.755	0.724	0.754	0.715
Average			0.781	0.743	**0.797**	0.742	0.775	0.740	0.768	0.719
100	10	Linear	0.741	**0.849**	0.783	0.843	0.729	0.847	0.761	0.812
		Multiplicative	0.735	0.856	0.776	0.855	0.721	**0.857**	0.757	0.812
		Tchebycheff	**0.999**	0.818	1.000	0.819	0.999	0.817	0.974	0.801
		Complex Nonmonotonic	0.795	0.809	**0.829**	0.808	0.784	0.809	0.807	0.801
Average			0.818	0.833	**0.847**	0.831	0.808	0.833	0.825	0.806
30	30	Linear	0.718	**0.784**	0.744	0.783	0.710	0.779	0.717	0.754
		Multiplicative	0.717	**0.814**	0.737	0.808	0.712	0.809	0.721	0.755
		Tchebycheff	**0.989**	0.767	0.988	0.759	0.984	0.769	0.952	0.751
		Complex Nonmonotonic	0.775	0.770	**0.791**	0.762	0.773	0.762	0.771	0.751
Average			0.800	0.784	**0.815**	0.778	0.795	0.780	0.790	0.753
100	30	Linear	0.751	0.856	0.790	**0.859**	0.743	0.857	0.769	0.817
		Multiplicative	0.752	0.862	0.789	**0.866**	0.743	0.864	0.764	0.818
		Tchebycheff	0.999	0.841	**1.000**	0.842	0.999	0.838	0.979	0.814
		Complex Nonmonotonic	0.804	0.826	0.838	0.829	0.801	0.829	0.817	0.810
Average			0.827	0.847	**0.854**	0.849	0.822	0.847	0.832	0.814

Table B.2 Mean Acc results for different reference set and query sizes (Query By Bagging).

# of Queries	Ref. Set Size	Underlying Value Function	Query By Bagging						Rand RF	Rand SVM
			LC		M		E			
			RF	SVM	RF	SVM	RF	SVM		
30	10	Linear	0.684	0.733	0.715	0.744	0.676	0.728	0.698	0.720
		Multiplicative	0.686	0.743	0.710	0.755	0.684	0.748	0.691	0.727
		Tchebycheff	0.971	0.737	0.969	0.731	0.966	0.729	0.928	0.715
		Complex Nonmonotonic	0.751	0.741	0.772	0.738	0.744	0.738	0.754	0.715
Average			0.773	0.738	0.791	0.742	0.767	0.736	0.768	0.719
100	10	Linear	0.729	0.847	0.777	0.839	0.716	0.839	0.761	0.812
		Multiplicative	0.729	0.839	0.775	0.840	0.720	0.841	0.757	0.812
		Tchebycheff	0.999	0.832	0.999	0.832	0.997	0.825	0.974	0.801
		Complex Nonmonotonic	0.784	0.823	0.828	0.828	0.778	0.829	0.807	0.801
Average			0.810	0.835	0.845	0.835	0.803	0.833	0.825	0.806
30	30	Linear	0.711	0.774	0.739	0.773	0.710	0.772	0.717	0.754
		Multiplicative	0.716	0.783	0.734	0.780	0.705	0.792	0.721	0.755
		Tchebycheff	0.986	0.766	0.985	0.769	0.982	0.765	0.952	0.751
		Complex Nonmonotonic	0.774	0.772	0.790	0.773	0.776	0.779	0.771	0.751
Average			0.797	0.774	0.812	0.774	0.793	0.777	0.790	0.753
100	30	Linear	0.744	0.851	0.788	0.846	0.735	0.853	0.769	0.817
		Multiplicative	0.751	0.852	0.783	0.848	0.741	0.855	0.764	0.818
		Tchebycheff	0.999	0.842	0.999	0.843	0.998	0.844	0.979	0.814
		Complex Nonmonotonic	0.799	0.837	0.837	0.834	0.798	**0.841**	0.817	0.810
Average			0.823	0.845	0.852	0.843	0.818	0.848	0.832	0.814

Table B.3 Mean BCA results for different reference set and query sizes (Uncertainty Sampling).

# of Queries	Ref. Set Size	Underlying Value Function	Uncertainty Sampling						Rand RF	Rand SVM
			LC		M		E			
			RF	SVM	RF	SVM	RF	SVM		
30	10	Linear	0.685	0.739	0.701	**0.740**	0.672	0.730	0.680	0.709
		Multiplicative	0.664	0.754	0.685	0.751	0.662	**0.767**	0.659	0.698
		Tchebycheff	**0.980**	0.716	0.977	0.713	0.973	0.714	0.918	0.707
		Complex Nonmonotonic	0.742	0.708	**0.768**	0.705	0.743	0.710	0.742	0.697
Average			0.768	0.729	**0.783**	0.727	0.763	0.730	0.750	0.703
100	10	Linear	0.735	**0.837**	0.767	0.831	0.721	0.833	0.742	0.798
		Multiplicative	0.710	0.842	0.748	0.831	0.695	**0.850**	0.722	0.787
		Tchebycheff	1.000	0.816	**1.000**	0.813	0.999	0.814	0.970	0.794
		Complex Nonmonotonic	0.779	0.797	0.815	0.793	0.769	0.799	0.792	0.789
Average			0.806	0.823	**0.833**	0.817	0.796	0.824	0.806	0.792
30	30	Linear	0.704	**0.774**	0.728	0.771	0.697	0.768	0.698	0.741
		Multiplicative	0.687	**0.800**	0.704	0.789	0.683	0.798	0.686	0.726
		Tchebycheff	**0.989**	0.766	0.988	0.754	0.982	0.766	0.945	0.745
		Complex Nonmonotonic	0.757	0.758	**0.776**	0.745	0.758	0.751	0.757	0.734
Average			0.784	0.774	**0.799**	0.765	0.780	0.771	0.771	0.737

Table B.3 (continued).

# of Queries	Ref. Set Size	Underlying Value Function	Uncertainty Sampling						Rand RF	Rand SVM
			LC		M		E			
			RF	SVM	RF	SVM	RF	SVM		
100	30	Linear	0.743	0.843	0.774	**0.847**	0.735	0.843	0.748	0.804
		Multiplicative	0.725	0.849	0.758	0.847	0.714	**0.854**	0.730	0.796
		Tchebycheff	1.000	0.839	**1.000**	0.837	0.999	0.836	0.976	0.807
		Complex Nonmonotonic	0.786	0.816	0.820	0.816	0.787	0.820	0.802	0.797
Average			0.813	0.837	0.838	0.837	0.809	0.838	0.814	0.801

Table B.4 Mean BCA results for different reference set and query sizes (Query By Bagging).

# of Queries	Ref. Set Size	Underlying Value Function	Query By Bagging						Rand RF	Rand SVM
			LC		M		E			
			RF	SVM	RF	SVM	RF	SVM		
30	10	Linear	0.675	0.722	0.699	0.730	0.669	0.718	0.680	0.709
		Multiplicative	0.656	0.711	0.677	0.728	0.656	0.714	0.659	0.698
		Tchebycheff	0.967	0.732	0.965	0.724	0.960	0.726	0.918	0.707
		Complex Nonmonotonic	0.734	0.728	0.758	0.724	0.728	0.724	0.742	0.697
Average			0.758	0.723	0.775	0.726	0.753	0.720	0.750	0.703
100	10	Linear	0.725	0.833	0.763	0.827	0.712	0.826	0.742	0.798
		Multiplicative	0.704	0.836	0.743	0.825	0.691	0.838	0.722	0.787
		Tchebycheff	0.999	0.829	0.999	0.826	0.996	0.822	0.970	0.794
		Complex Nonmonotonic	0.768	0.814	0.811	0.817	0.763	**0.821**	0.792	0.789
Average			0.799	0.828	0.829	0.824	0.790	0.827	0.806	0.792
30	30	Linear	0.700	0.765	0.722	0.762	0.698	0.762	0.698	0.741
		Multiplicative	0.684	0.758	0.703	0.755	0.676	0.764	0.686	0.726
		Tchebycheff	0.985	0.764	0.983	0.764	0.979	0.763	0.945	0.745
		Complex Nonmonotonic	0.758	0.759	0.774	0.758	0.759	0.769	0.757	0.734
Average			0.782	0.762	0.796	0.759	0.778	0.764	0.771	0.737
100	30	Linear	0.738	0.839	0.772	0.835	0.729	0.840	0.748	0.804
		Multiplicative	0.721	0.852	0.753	0.832	0.709	0.851	0.730	0.796
		Tchebycheff	0.999	0.841	0.999	0.838	0.998	0.841	0.976	0.807
		Complex Nonmonotonic	0.784	0.830	0.822	0.823	0.784	**0.834**	0.802	0.797
Average			0.810	0.840	0.837	0.832	0.805	**0.841**	0.814	0.801

Table B.5 Mean Kappa results for different reference set and query sizes (Uncertainty Sampling).

# of Queries	Ref. Set Size	Underlying Value Function	Uncertainty Sampling						Rand RF	Rand SVM
			LC		M		E			
			RF	SVM	RF	SVM	RF	SVM		
30	10	Linear	0.574	**0.647**	0.599	0.647	0.555	0.631	0.561	0.592
		Multiplicative	0.556	0.666	0.586	0.662	0.554	**0.668**	0.542	0.592
		Tchebycheff	**0.974**	0.578	0.969	0.582	0.969	0.572	0.898	0.579
		Complex Nonmonotonic	0.653	0.579	**0.687**	0.579	0.653	0.582	0.642	0.576
Average			0.689	0.618	**0.710**	0.617	0.683	0.613	0.661	0.585
100	10	Linear	0.650	**0.789**	0.701	0.781	0.634	0.785	0.658	0.728
		Multiplicative	0.631	0.789	0.680	0.787	0.612	**0.791**	0.642	0.723
		Tchebycheff	0.999	0.730	**0.999**	0.731	0.999	0.727	0.964	0.710
		Complex Nonmonotonic	0.714	0.710	**0.760**	0.707	0.700	0.709	0.720	0.711
Average			0.749	0.754	**0.785**	0.752	0.736	0.753	0.746	0.718
30	30	Linear	0.603	**0.695**	0.639	0.693	0.593	0.687	0.590	0.646
		Multiplicative	0.593	**0.722**	0.617	0.712	0.588	0.716	0.586	0.639
		Tchebycheff	**0.986**	0.658	0.984	0.647	0.980	0.660	0.931	0.634
		Complex Nonmonotonic	0.681	0.660	**0.702**	0.651	0.679	0.649	0.666	0.635
Average			0.716	0.684	**0.736**	0.676	0.710	0.678	0.693	0.639
100	30	Linear	0.661	0.798	0.709	**0.802**	0.652	0.799	0.669	0.737
		Multiplicative	0.652	0.800	0.698	**0.804**	0.640	0.801	0.653	0.734
		Tchebycheff	0.999	0.769	**0.999**	0.770	0.999	0.764	0.969	0.728
		Complex Nonmonotonic	0.726	0.744	0.771	0.749	0.722	0.748	0.734	0.725
Average			0.760	0.778	**0.794**	0.781	0.753	0.778	0.756	0.731

Table B.6 Mean Kappa results for different reference set and query sizes (Query By Bagging).

# of Queries	Ref. Set Size	Underlying Value Function	Query By Bagging						Rand RF	Rand SVM
			LC		M		E			
			RF	SVM	RF	SVM	RF	SVM		
30	10	Linear	0.559	0.613	0.594	0.628	0.549	0.609	0.561	0.592
		Multiplicative	0.547	0.614	0.576	0.636	0.545	0.620	0.542	0.592
		Tchebycheff	0.963	0.615	0.960	0.611	0.956	0.607	0.898	0.579
		Complex Nonmonotonic	0.646	0.620	0.674	0.617	0.634	0.617	0.642	0.576
Average			0.679	0.615	0.701	0.623	0.671	0.614	0.661	0.585
100	10	Linear	0.635	0.781	0.692	0.772	0.617	0.772	0.658	0.728
		Multiplicative	0.620	0.769	0.678	0.769	0.606	0.771	0.642	0.723
		Tchebycheff	0.999	0.758	0.999	0.759	0.996	0.748	0.964	0.710
		Complex Nonmonotonic	0.699	0.745	0.756	0.752	0.691	0.753	0.720	0.711
Average			0.738	0.763	0.781	0.763	0.728	0.761	0.746	0.718
30	30	Linear	0.597	0.677	0.632	0.676	0.597	0.676	0.590	0.646
		Multiplicative	0.591	0.682	0.613	0.678	0.576	0.696	0.586	0.639
		Tchebycheff	0.982	0.662	0.980	0.666	0.977	0.656	0.931	0.634
		Complex Nonmonotonic	0.676	0.668	0.699	0.673	0.681	0.679	0.666	0.635
Average			0.711	0.672	0.731	0.673	0.708	0.676	0.693	0.639

Table B.6 (continued).

# of Queries	Ref. Set Size	Underlying Value Function	Query By Bagging						Rand RF	Rand SVM
			LC		M		E			
			RF	SVM	RF	SVM	RF	SVM		
100	30	Linear	0.653	0.788	0.706	0.782	0.643	0.791	0.669	0.737
		Multiplicative	0.649	0.788	0.691	0.782	0.636	0.791	0.653	0.734
		Tchebycheff	0.999	0.772	0.999	0.776	0.997	0.775	0.969	0.728
		Complex Nonmonotonic	0.719	0.766	0.771	0.763	0.718	**0.772**	0.734	0.725
Average			0.755	0.778	0.792	0.775	0.749	0.782	0.756	0.731

Table B.7 Mean MAEO results for different reference set and query sizes (Uncertainty Sampling).

# of Queries	Ref. Set Size	Underlying Value Function	Uncertainty Sampling						Rand RF	Rand SVM
			LC		M		E			
			RF	SVM	RF	SVM	RF	SVM		
30	10	Linear	0.647	0.495	0.576	0.463	0.689	0.521	0.601	0.503
		Multiplicative	0.628	0.389	0.551	0.389	0.640	**0.381**	0.587	0.468
		Tchebycheff	**0.040**	0.443	0.048	0.449	0.049	0.446	0.120	0.491
		Complex Nonmonotonic	0.448	0.440	**0.404**	0.449	0.454	0.447	0.429	0.495
Average			0.441	0.442	**0.395**	0.437	0.458	0.449	0.434	0.489
100	10	Linear	0.570	**0.271**	0.442	0.283	0.616	0.281	0.462	0.316
		Multiplicative	0.514	**0.233**	0.416	0.245	0.549	0.237	0.433	0.309
		Tchebycheff	0.001	0.263	**0.001**	0.267	0.002	0.262	0.042	0.324
		Complex Nonmonotonic	0.372	0.278	0.310	0.283	0.401	**0.278**	0.336	0.335
Average			0.364	**0.261**	0.292	0.270	0.392	0.264	0.318	0.321
30	30	Linear	0.594	0.401	0.514	**0.398**	0.618	0.411	0.555	0.435
		Multiplicative	0.553	**0.312**	0.502	0.333	0.586	0.325	0.523	0.419
		Tchebycheff	**0.022**	0.374	0.024	0.393	0.032	0.367	0.074	0.418
		Complex Nonmonotonic	0.413	0.385	0.381	0.405	0.423	0.393	0.401	0.437
Average			0.395	0.368	**0.355**	0.382	0.415	0.374	0.388	0.427
100	30	Linear	0.541	0.255	0.420	**0.251**	0.575	0.256	0.451	0.309
		Multiplicative	0.470	0.228	0.391	**0.224**	0.498	0.227	0.423	0.302
		Tchebycheff	0.001	0.240	**0.001**	0.248	0.001	0.246	0.032	0.300
		Complex Nonmonotonic	0.354	0.271	0.295	0.276	0.362	0.269	0.317	0.321
Average			0.341	**0.249**	0.277	0.250	0.359	0.249	0.306	0.308

Table B.8 Mean MAEO results for different reference set and query sizes (Query By Bagging).

# of Queries	Ref. Set Size	Underlying Value Function	Query By Bagging						Rand RF	Rand SVM
			LC		M		E			
			RF	SVM	RF	SVM	RF	SVM		
30	10	Linear	0.674	0.497	0.573	**0.460**	0.689	0.525	0.601	0.503
		Multiplicative	0.627	0.428	0.554	0.414	0.635	0.422	0.587	0.468
		Tchebycheff	0.055	0.449	0.062	0.477	0.068	0.479	0.120	0.491
		Complex Nonmonotonic	0.457	0.463	0.412	0.458	0.460	0.477	0.429	0.495
Average			0.453	0.459	0.400	0.452	0.463	0.476	0.434	0.489
100	10	Linear	0.588	0.273	0.445	0.280	0.621	0.295	0.462	0.316
		Multiplicative	0.516	0.265	0.415	0.269	0.547	0.258	0.433	0.309
		Tchebycheff	0.002	0.269	0.002	0.276	0.007	0.281	0.042	0.324
		Complex Nonmonotonic	0.392	0.296	0.307	0.290	0.408	0.286	0.336	0.335
Average			0.374	0.276	0.292	0.279	0.396	0.280	0.318	0.321
30	30	Linear	0.607	0.405	0.530	0.407	0.616	0.419	0.555	0.435
		Multiplicative	0.564	0.368	0.508	0.377	0.588	0.353	0.523	0.419
		Tchebycheff	0.027	0.391	0.031	0.389	0.036	0.388	0.074	0.418
		Complex Nonmonotonic	0.401	0.393	**0.380**	0.404	0.404	0.382	0.401	0.437
Average			0.400	0.389	0.362	0.394	0.411	0.385	0.388	0.427
100	30	Linear	0.547	0.257	0.421	0.266	0.579	0.257	0.451	0.309
		Multiplicative	0.477	0.240	0.409	0.254	0.508	0.235	0.423	0.302
		Tchebycheff	0.002	0.247	0.002	0.256	0.004	0.248	0.032	0.300
		Complex Nonmonotonic	0.362	0.268	0.295	0.281	0.369	**0.262**	0.317	0.321
Average			0.347	0.253	0.282	0.264	0.365	0.251	0.306	0.308

Table B.9 Mean MSEO results for different reference set and query sizes (Uncertainty Sampling).

# of Queries	Ref. Set Size	Underlying Value Function	Uncertainty Sampling						Rand RF	Rand SVM
			LC		M		E			
			RF	SVM	RF	SVM	RF	SVM		
30	10	Linear	64.1120	53.8523	39.8091	28.9723	68.3872	61.8456	40.7132	34.3902
		Multiplicative	53.9755	31.3777	33.9872	24.2459	53.1579	27.1935	37.5102	27.0581
		Tchebycheff	**1.3825**	35.2771	1.9122	26.9944	1.4812	33.5196	5.2014	29.6020
		Complex Nonmonotonic	31.1238	33.3703	**22.1876**	26.1240	31.0525	32.2007	24.2068	32.2028
Average			37.6484	38.4694	**24.4740**	26.5841	38.5197	38.6899	26.9079	30.8133
100	10	Linear	63.6732	14.8586	25.8424	11.2772	64.9529	14.8905	24.8347	12.0926
		Multiplicative	42.6368	12.2026	21.9506	**10.8627**	48.8665	11.3461	20.8758	12.8847
		Tchebycheff	0.0016	13.2819	**0.0014**	10.5806	0.0044	12.6275	0.8512	12.9871
		Complex Nonmonotonic	28.2044	15.2384	14.0486	**11.4701**	33.0153	14.0487	15.1175	15.2463
Average			33.6290	13.8954	15.4607	**11.0476**	36.7098	13.2282	15.4198	13.3026
30	30	Linear	50.1411	26.0894	29.9659	**20.3251**	57.8520	26.4618	33.1312	21.5211
		Multiplicative	36.9601	15.9705	28.9615	**15.3968**	47.2411	18.5041	29.7824	22.2283
		Tchebycheff	0.3226	25.4308	0.4603	22.7098	0.6677	23.2760	1.8208	20.8691
		Complex Nonmonotonic	26.4534	22.6825	19.7835	21.0548	25.5101	25.4693	20.9625	25.6105
Average			28.4693	22.5433	**19.7928**	19.8716	32.8177	23.4278	21.4242	22.5572

Table B.9 (continued).

# of Queries	Ref. Set Size	Underlying Value Function	Uncertainty Sampling						Rand RF	Rand SVM
			LC		M		E			
			RF	SVM	RF	SVM	RF	SVM		
100	30	Linear	57.4230	11.6884	21.5526	**8.2488**	63.4335	10.8759	24.0705	11.6610
		Multiplicative	36.0431	11.2462	18.8305	**8.3932**	40.9817	10.4274	20.1302	11.7416
		Tchebycheff	0.0018	11.6591	**0.0017**	9.5231	0.0022	12.6250	0.4109	11.6029
		Complex Nonmonotonic	25.2292	15.3597	12.4484	**11.1366**	26.0765	13.6063	13.1148	13.2084
Average			29.6743	12.4883	13.2083	**9.3254**	32.6235	11.8836	14.4316	12.0534

Table B.10 Mean MSEO results for different reference set and query sizes (Query By Bagging).

# of Queries	Ref. Set Size	Underlying Value Function	Query By Bagging						Rand RF	Rand SVM
			LC		M		E			
			RF	SVM	RF	SVM	RF	SVM		
30	10	Linear	58.8768	53.5961	38.9490	**28.4616**	63.2483	61.8131	40.7132	34.3902
		Multiplicative	52.9846	29.9580	33.8043	**23.9197**	54.3427	28.9156	37.5102	27.0581
		Tchebycheff	1.9134	34.0493	1.7427	33.3412	3.0996	42.5545	5.2014	29.6020
		Complex Nonmonotonic	33.2765	37.4889	24.0703	29.9596	33.1402	37.6712	24.2068	32.2028
Average			36.7628	38.7731	24.6416	28.9205	38.4577	42.7386	26.9079	30.8133
100	10	Linear	64.6935	13.3316	23.4463	**10.3410**	71.5912	16.3875	24.8347	12.0926
		Multiplicative	46.0242	14.7097	22.0955	11.7916	51.3487	12.7369	20.8758	12.8847
		Tchebycheff	0.0036	14.3784	0.0038	11.8677	0.0232	15.7321	0.8512	12.9871
		Complex Nonmonotonic	33.9365	17.8493	14.3357	12.5495	36.6077	14.6700	15.1175	15.2463
Average			36.1644	15.0672	14.9703	11.6374	39.8927	14.8816	15.4198	13.3026
30	30	Linear	51.0381	26.9253	32.5889	21.2882	52.0519	31.0083	33.1312	21.5211
		Multiplicative	43.6723	19.2338	29.7628	18.9846	42.7182	18.2302	29.7824	22.2283
		Tchebycheff	**0.2919**	26.5941	0.3840	20.3065	0.6173	25.8728	1.8208	20.8691
		Complex Nonmonotonic	23.4003	24.0789	**18.7714**	21.9411	23.6203	22.3889	20.9625	25.6105
Average			29.6006	24.2080	20.3768	20.6301	29.7519	24.3750	21.4242	22.5572
100	30	Linear	53.5471	11.0930	21.8480	9.6527	67.0824	11.3685	24.0705	11.6610
		Multiplicative	36.5881	10.6638	20.7834	9.9343	40.9743	9.8357	20.1302	11.7416
		Tchebycheff	0.0030	12.5930	0.0025	9.9729	0.0090	11.5701	0.4109	11.6029
		Complex Nonmonotonic	26.4525	15.1074	12.0674	11.7412	27.1307	12.1432	13.1148	13.2084
Average			29.1477	12.3643	13.6753	10.3253	33.7991	11.2294	14.4316	12.0534

APPENDIX C

RESULTS OF THE PAIRWISE T-TESTS COMPARING THE BEST ALGORITHMS AND THE RANDOM APPROACH

Paired T-Test and CI: RF-QBB-LC; RF-Rand

Paired T for RF-QBB-LC - RF-Rand

	N	Mean	StDev	SE Mean
RF-QBB-LC	100	0,94602	0,01839	0,00184
RF-Rand	100	0,92568	0,00988	0,00099
Difference	100	0,020340	0,009832	0,000983

95% lower bound for mean difference: 0,018708
T-Test of mean difference = 0 (vs > 0): T-Value = 20,69
P-Value = 0,000

Figure C.1 Pairwise t-test for Acc (Reference set size 10. Value function has 3 attributes and 2 classes.)

Paired T-Test and CI: SVM-US-LC; SVM-Rand

Paired T for SVM-US-LC - SVM-Rand

	N	Mean	StDev	SE Mean
SVM-US-LC	100	0,77434	0,08664	0,00866
SVM-Rand	100	0,71632	0,07229	0,00723
Difference	100	0,05802	0,01644	0,00164

95% lower bound for mean difference: 0,05529
T-Test of mean difference = 0 (vs > 0): T-Value = 35,29
P-Value = 0,000

Figure C.2 Pairwise t-test for Acc (Reference set size 10. Value function has 3 attributes and 5 classes.)

Paired T-Test and CI: RF-US-Mar; RF-Rand

Paired T for RF-US-Mar - RF-Rand

	N	Mean	StDev	SE Mean
RF-US-Mar	100	0,88958	0,02630	0,00263
RF-Rand	100	0,86873	0,01901	0,00190
Difference	100	0,020854	0,008585	0,000858

95% lower bound for mean difference: 0,019429
T-Test of mean difference = 0 (vs > 0): T-Value = 24,29
P-Value = 0,000

Figure C.3 Pairwise t-test for Acc (Reference set size 10. Value function has 6 attributes and 2 classes.)

Paired T-Test and CI: SVM-US-Mar; SVM-Rand

Paired T for SVM-US-Mar - SVM-Rand

	N	Mean	StDev	SE Mean
SVM-US-Mar	100	0,51926	0,07758	0,00776
SVM-Rand	100	0,48567	0,07589	0,00759
Difference	100	0,033587	0,009356	0,000936

95% lower bound for mean difference: 0,032034
T-Test of mean difference = 0 (vs > 0): T-Value = 35,90
P-Value = 0,000

Figure C.4 Pairwise t-test for Acc (Reference set size 10. Value function has 6 attributes and 5 classes.)

```
Paired T-Test and CI: RF-QBB-Mar; RF-Rand

Paired T for RF-QBB-Mar - RF-Rand

              N      Mean     StDev   SE Mean
RF-QBB-Mar  100   0,94988   0,01586   0,00159
RF-Rand     100   0,92182   0,01142   0,00114
Difference  100  0,028059  0,007572  0,000757


95% lower bound for mean difference: 0,026802
T-Test of mean difference = 0 (vs > 0): T-Value = 37,06
P-Value = 0,000
```

Figure C.5 Pairwise t-test for Acc (Reference set size 30. Value function has 3 attributes and 2 classes.)

```
Paired T-Test and CI: SVM-US-Ent; SVM-Rand

Paired T for SVM-US-Ent - SVM-Rand

              N     Mean    StDev  SE Mean
SVM-US-Ent  100  0,80429  0,06327  0,00633
SVM-Rand    100  0,75283  0,05442  0,00544
Difference  100  0,05146  0,01343  0,00134


95% lower bound for mean difference: 0,04923
T-Test of mean difference = 0 (vs > 0): T-Value = 38,32
P-Value = 0,000
```

Figure C.6 Pairwise t-test for Acc (Reference set size 30. Value function has 3 attributes and 5 classes.)

Paired T-Test and CI: RF-US-Mar; RF-Rand

Paired T for RF-US-Mar - RF-Rand

	N	Mean	StDev	SE Mean
RF-US-Mar	100	0,89973	0,01958	0,00196
RF-Rand	100	0,87532	0,01330	0,00133
Difference	100	0,024417	0,008313	0,000831

95% lower bound for mean difference: 0,023037
T-Test of mean difference = 0 (vs > 0): T-Value = 29,37
P-Value = 0,000

Figure C.7 Pairwise t-test for Acc (Reference set size 30. Value function has 6 attributes and 2 classes.)

Paired T-Test and CI: SVM-US-Mar; SVM-Rand

Paired T for SVM-US-Mar - SVM-Rand

	N	Mean	StDev	SE Mean
SVM-US-Mar	100	0,54293	0,05402	0,00540
SVM-Rand	100	0,52875	0,05284	0,00528
Difference	100	0,014188	0,007806	0,000781

95% lower bound for mean difference: 0,012892
T-Test of mean difference = 0 (vs > 0): T-Value = 18,18
P-Value = 0,000

Figure C.8 Pairwise t-test for Acc (Reference set size 30. Value function has 6 attributes and 5 classes.)

```
Paired T-Test and CI: SVM-QBB-LC; SVM-Rand

Paired T for SVM-QBB-LC - SVM-Rand

               N      Mean    StDev  SE Mean
SVM-QBB-LC   100   0,23187  0,09915  0,00991
SVM-Rand     100   0,30804  0,09617  0,00962
Difference   100  -0,07617  0,01045  0,00104


95% upper bound for mean difference: -0,07443
T-Test of mean difference = 0 (vs < 0): T-Value = -72,91
P-Value = 0,000
```

Figure C.9 Pairwise t-test for MAEO (Reference set size 10. Value function has 3 attributes and 5 classes.)

```
Paired T-Test and CI: SVM-US-Ent; SVM-Rand

Paired T for SVM-US-Ent - SVM-Rand

               N      Mean    StDev  SE Mean
SVM-US-Ent   100    0,6119   0,1921   0,0192
SVM-Rand     100    0,6472   0,1700   0,0170
Difference   100  -0,03526  0,02854  0,00285


95% upper bound for mean difference: -0,03052
T-Test of mean difference = 0 (vs < 0): T-Value = -12,35
P-Value = 0,000
```

Figure C.10 Pairwise t-test for MAEO (Reference set size 10. Value function has 6 attributes and 5 classes.)

Paired T-Test and CI: SVM-US-Ent; SVM-Rand

Paired T for SVM-US-Ent - SVM-Rand

	N	Mean	StDev	SE Mean
SVM-US-Ent	100	0,19880	0,06989	0,00699
SVM-Rand	100	0,25975	0,06547	0,00655
Difference	100	-0,06096	0,01308	0,00131

95% upper bound for mean difference: -0,05878
T-Test of mean difference = 0 (vs < 0): T-Value = -46,59
P-Value = 0,000

Figure C.11 Pairwise t-test for MAEO (Reference set size 30. Value function has 3 attributes and 5 classes.)

Paired T-Test and CI: SVM-QBB-Ent; SVM-Rand

Paired T for SVM-QBB-Ent - SVM-Rand

	N	Mean	StDev	SE Mean
SVM-QBB-Ent	100	0,51385	0,09990	0,00999
SVM-Rand	100	0,56621	0,09489	0,00949
Difference	100	-0,05236	0,01396	0,00140

95% upper bound for mean difference: -0,05004
T-Test of mean difference = 0 (vs < 0): T-Value = -37,50
P-Value = 0,000

Figure C.12 Pairwise t-test for MAEO (Reference set size 30. Value function has 6 attributes and 5 classes.)

CIRRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Erişkin, Levent
Nationality: Turkish (TC)
Date and Place of Birth: 7 September 1976, Gölcük
Marital Status: Married
Phone: +90 505 795 8361
email: levent.eriskin@gmail.com

EDUCATION

Degree	Institution	Year of Graduation
MS	US Naval Post Graduate School, Monterey/CA, USA	2003
BS	Turkish Naval Academy, İstanbul	1998
High School	Turkish Naval High School, İstanbul	1994

WORK EXERIENCE

Year	Place	Enrollment
2014-	Decision Support Dept., TuN	Logistics Analysis Division Head
2008-2014	Decision Support Dept., TuN	Project Officer
2006-2008	TCG Kemalreis Frigate	Operations Officer
2005-2006	TCG Kemalreis Frigate	Navigations Officer
2004-2005	Personnel Dept., TuN	Project Officer
2000-2001	5th Destroyer Division	Training Officer
1998-2000	TCG Barbaros Frigate	Communications Officer

CONFERENCE PRESENTATIONS

Erişkin, L., Köksal, G., "Estimating Non-Additive Value Functions With Active Learning in the Ordinal Classification Setting ", INFORMS Annual Meeting, Philadelphia, Pennsylvania, USA, 2015.

Erişkin, L., Köksal, G., “Toplanır Olmayan Değer Fonksiyonlarının Sınıflandırma Durumu İçin Aktif Öğrenme ile Tahmin Edilmesi”, National Operations Research and Industrial Engineering (YAEM) Congress, Ankara, Turkey, 2015.

Dolgun, L.E, Erişkin, L., Köksal, G., "Analysis of Multivariate Loss Functions under Preferential and Statistical Dependencies", INFORMS Annual Meeting, Minneapolis, Minnesota, USA, 2013.

AWARDS

Headquarters Achievement Award 2010

Early Promotion to Lieutenant 2005

US Naval Post Graduate School Distinguished Graduate 2001

Turkish Naval Academy Distinguished Graduate (2/224, Awarded by the President of the Grand National Assembly of Turkey) 1998

Turkish Naval High School Distinguished Graduate (3/240) (Awarded by the Commander of Naval Training and Education Command) 1994

FOREIGN LANGUAGES

Advanced English

HOBBIES

Riding motorcycle, playing basketball, reading books, watching movies.