

# Design Preference Elicitation, Identification and Estimation

by

Yi Ren

A dissertation submitted in partial fulfillment  
of the requirements for the degree of  
Doctor of Philosophy  
(Mechanical Engineering)  
in The University of Michigan  
2012

Doctoral Committee:

Professor Panos Y. Papalambros, Chair  
Professor Richard D. Gonzalez  
Professor Noboru Kikuchi  
Professor George Michailidis

© Yi Ren 2012  
All Rights Reserved

For my parents

## ACKNOWLEDGEMENTS

I would like to gratefully and sincerely thank Dr. Panos Papalambros for his guidance, understanding, and friendship during my graduate studies. The Chinese maxim says “It’s better to teach fishing rather than offering fish”. That is exactly what you did, with great patience. It is your consistent intellectual and financial support that allowed me to freely enjoy my research and my life in general. For everything you have done for me, Dr. Papalambros, I thank you. I would also like to acknowledge my committee members: Dr. Richard Gonzalez, Dr. Noboru Kikuchi and Dr. George Michailidis for their valuable inputs. In particular, Dr. Gonzalez has been generously sharing his expertise to enrich my research throughout years of my graduate study.

I would like to give my special thanks to Dr. Yilun Chen. You not only broadened my knowledge but more importantly, showed me the devotion, enthusiasm and honesty a researcher should have. I also want to thank all of the members of the Optimal Design Laboratory for their support and participation in my research.

Finally and most importantly, I would like to thank my parents for their understanding, patience and faith in me. They have always been supportive to my life path and I truly appreciate their challenge on my research to keep me connected with the real world. My last and special thank goes to my girlfriend Zhuqing. I want to thank you for all your efforts to make me more matured and responsible in and outside of research. Your heartfelt company during this special and busy year was an essential force driving me forward.

# TABLE OF CONTENTS

<b>DEDICATION</b> . . . . .	ii
<b>ACKNOWLEDGEMENTS</b> . . . . .	iii
<b>LIST OF FIGURES</b> . . . . .	viii
<b>LIST OF TABLES</b> . . . . .	xii
<b>ABSTRACT</b> . . . . .	xiii
<b>CHAPTER</b>	
<b>I. Introduction</b> . . . . .	1
1.1 Motivation . . . . .	1
1.2 Problem Definition . . . . .	3
1.2.1 Design space and features . . . . .	3
1.2.2 Utility: A measure of preference . . . . .	4
1.2.3 Interaction . . . . .	5
1.2.4 Subject response model . . . . .	5
1.2.5 Preference identification . . . . .	7
1.2.6 Preference elicitation . . . . .	8
1.2.7 Preference estimation . . . . .	8
1.3 Related Work . . . . .	9
1.3.1 Interactive evolutionary computation . . . . .	10
1.3.2 Recommender systems . . . . .	15
1.3.3 Conjoint analysis . . . . .	19
1.3.4 Summary . . . . .	21
1.4 Dissertation Overview . . . . .	22
<b>II. Literature Review</b> . . . . .	24
2.1 Conjoint Analysis . . . . .	25
2.1.1 Utility and choice probability . . . . .	25

2.1.2	Likelihood models . . . . .	27
2.1.3	Parameter estimation and its variance . . . . .	30
2.2	Support Vector Machine (SVM) . . . . .	31
2.2.1	Basic concepts . . . . .	32
2.2.2	Primal and dual problems . . . . .	34
2.2.3	Kernel trick . . . . .	36
2.2.4	Soft margin formulation and complexity control . . . . .	39
2.3	Conclusion . . . . .	41
<b>III. Preference Elicitation . . . . .</b>		<b>42</b>
3.1	Efficient Global Optimization (EGO) . . . . .	43
3.1.1	Kriging modeling . . . . .	45
3.1.2	Mean-squared error of the predictor . . . . .	48
3.1.3	Expected improvement . . . . .	48
3.1.4	Properties of the expected improvement . . . . .	49
3.1.5	Optimization on the expected improvement . . . . .	50
3.2	SVM Search . . . . .	51
3.2.1	The algorithm . . . . .	51
3.2.2	SVM Search Simulated Test Results . . . . .	53
3.3	EGO Search . . . . .	57
3.3.1	The algorithm . . . . .	57
3.3.2	EGO Search versus SVM Search in simulated tests . . . . .	60
3.4	Discussion on the EGO Search Algorithm . . . . .	60
3.4.1	Parameters used in the merit function . . . . .	62
3.4.2	Dimensionality and user sensitivity . . . . .	64
3.4.3	Computational cost . . . . .	65
3.4.4	EGO Search versus GA plus SVM . . . . .	65
3.5	Vehicle Exterior Styling Design Elicitation . . . . .	67
3.5.1	Software development . . . . .	67
3.5.2	Convergence test setup . . . . .	68
3.5.3	User data analysis . . . . .	70
3.5.4	Observed issues in user interactions . . . . .	74
3.6	Concluding Remarks . . . . .	77
<b>IV. Augmented Preference Elicitation . . . . .</b>		<b>79</b>
4.1	Preference Modeling with Comparison Tree . . . . .	80
4.1.1	Terminology and definition . . . . .	80
4.1.2	Learning based on pairwise comparison . . . . .	81
4.1.3	Simulated test results and discussion . . . . .	81
4.2	Variations of the Merit Function . . . . .	83
4.2.1	Kushner's criterion . . . . .	86
4.2.2	Generalized Expected Improvement (GEI) . . . . .	86
4.2.3	Lower Confidence Bounding function (LCB) . . . . .	87

4.2.4	Locating the Regional Extreme (LRE)	87
4.2.5	Switching criterion	88
4.2.6	Computational difficulty	88
4.3	Fast EGO and Its Simulated Test Results	89
4.3.1	The geometric meaning of $\hat{\sigma}$	90
4.3.2	A computationally inexpensive merit function for EGO	90
4.3.3	Simulated test results	93
4.3.4	Discussion	94
4.4	Concluding Remarks	98

**V. Preference Estimation and Identification** . . . . . 99

5.1	Active Learning Background	102
5.1.1	Non-adaptive query	102
5.1.2	Active learning in conjoint analysis	103
5.1.3	Active learning in machine learning	104
5.2	Preference Estimation	111
5.2.1	Problem definition	111
5.2.2	Design features	112
5.2.3	Estimation of $\mathbf{w}$	113
5.2.4	Active learning on preference estimation	115
5.2.5	Test setup	118
5.2.6	D-optimal design setup	119
5.2.7	Performance of active learning, $D$ -optimal and random designs on 2D problems	120
5.2.8	Limitation of the linearity assumption	122
5.2.9	Identification of preference features	123
5.2.10	Robustness of active learning	124
5.3	Preference Identification	126
5.3.1	Justification of the loss function	126
5.3.2	Application of active learning	129
5.3.3	Active learning on preference identification	132
5.3.4	Simulated tests and results	132
5.4	Concluding Remarks	135

**VI. Collaborative Filtering in Preference Elicitation** . . . . . 137

6.1	Problem Formulation	137
6.2	Heuristics	138
6.2.1	Simplification of a search tree (JUMP0)	139
6.2.2	Exploration of more efficient trees (JUMP1)	139
6.2.3	2D Demonstration of the exploration heuristic	142
6.2.4	Enhancement to JUMP1 (JUMP2)	145
6.2.5	2D Demonstration of the enhanced heuristic	148
6.2.6	Update of initial guess (JUMP3)	149

6.2.7	2D Demonstration of initial guess update . . . . .	151
6.3	Discussion . . . . .	152
6.4	Relationship With Collaborative Filtering . . . . .	156
6.5	Concluding Remarks . . . . .	157
<b>VII.</b>	<b>Conclusions . . . . .</b>	<b>158</b>
7.1	Summary . . . . .	158
7.2	Contributions . . . . .	161
7.3	Assumptions and Limitations . . . . .	162
7.4	Future Work . . . . .	163
7.4.1	Preference modeling with enriched user interaction data . . . . .	163
7.4.2	Calibration on weighted search . . . . .	164
7.4.3	Feature addition and subtraction . . . . .	164
7.4.4	Linking user preferences and designer decisions . . . . .	165
7.4.5	Search towards a mutual target . . . . .	166
<b>BIBLIOGRAPHY</b>	<b>. . . . .</b>	<b>167</b>



## LIST OF FIGURES

### Figure

1.1	General framework of an IGA . . . . .	11
1.2	Multi-armed roulette wheel parent selection . . . . .	13
1.3	An example of crossover. Here the two grayed designs are parents and the white one is a child. The child acquires the hood design of the first parent and the silhouette design of the second parent. . . . .	13
2.1	A separation hyperplane ( $\hat{\mathbf{w}}^T \mathbf{x}$ ) for a two dimensional training set. Figure from <i>Cristianini and Shawe-Taylor (2000)</i> . . . . .	33
2.2	$\gamma$ is the margin between two classes. Figure from <i>Cristianini and Shawe-Taylor (2000)</i> . . . . .	34
2.3	Mapping data to a different space where they can be linearly separated. Figure from <i>Cristianini and Shawe-Taylor (2000)</i> . . . . .	36
2.4	Two classes separated by a hyperplane in the feature space (Gaussian kernel). Figure from <i>Cristianini and Shawe-Taylor (2000)</i> . . . . .	38
3.1	EGO operation concept. . . . .	44
3.2	SVM Search scattering and classification. . . . .	52
3.3	SVM Search design space reduction. . . . .	53
3.4	Proposed SVM Search algorithm . . . . .	54
3.5	Proposed EGO Search algorithm . . . . .	58
3.6	Comparison between EGO Search and SVM Search. . . . .	61
3.7	Comparison between EGO Search and GA plus SVM. . . . .	66
3.8	Online human-computer interaction interface and data visualization. (a): The interactive environment at “/convergencetest.html” allows the user to zoom, pan, rotate each design and updates the guesses once the user hits the “Next” button; (b): The data visualization window at “/log.html” has all user tests listed at the top, number of iterations in the middle, and the cumulated data at the bottom. The red curve represents the target design, the highlighted dark curve(s) represent the preferred design at this point and the rest all not preferred. . . . .	69
3.9	Normalized Euclidean distance from each sampled design to the target in each test. The circled design is the one submitted by the user, while the triangle design has the lowest Euclidean distance to the target. . . . .	72

3.10	Visual comparison between user test results and the targets from side and perspective views. . . . .	73
3.11	Visual comparison between samples in the last iteration and the target. Data generated from Test0 on “log.html”. Euclidean distances to the target are listed under the designs. . . . .	74
3.12	Designs labeled as preferred in the first four iterations, compared with the target and the one with the minimum Euclidean distance within all samples. Euclidean distances to the target are listed under the designs. . . . .	75
3.13	Features that capture the roof design. . . . .	75
3.14	Samples from data “Test0” in the feature and Euclidean space. Multidimensional scaling is applied to both measures to create 2D visualization of the data. . . . .	76
4.1	2D Six-Hump Camelback . . . . .	83
4.2	2D Branin . . . . .	83
4.3	10 dimensional Gaussian, $\lambda^{\text{user}} = 5$ . . . . .	84
4.4	20 dimensional Gaussian, $\lambda^{\text{user}} = 5$ . . . . .	84
4.5	31 dimensional Gaussian, $\lambda^{\text{user}} = 5$ . . . . .	84
4.6	Violation rates at each iteration in each test . . . . .	85
4.7	The computational costs at the first 30 iterations when optimizing the 31-dimensional Gaussian in Equation (3.19). . . . .	89
4.8	Merit functions during the search on a Branin function (Equation (3.17)). Bright areas indicate high function values. . . . .	91
4.9	Comparison between the mean-squared error function and the minimum distance function under a set of sampled designs (represented as white crosses). Bright areas indicate high function values. . . . .	92
4.10	Convergence performance of searches with $f_{\text{fast}}$ and $f_{\text{lcb}}$ as their merit functions. . . . .	95
4.11	Computational cost of searches with $f_{\text{fast}}$ and $f_{\text{lcb}}$ as their merit functions. . . . .	96
4.12	Contour plot of the merit function at different $r^2$ values. . . . .	97
5.1	Geometrical representation of the version space. In this 2D case, the version space is the highlighted arc of the circle. Each normal vector of a constraining hyperplane represents a sample point $\mathbf{v}_i$ and the label $y_i$ determines which side of the hyperplane is feasible for $\mathbf{w}$ . The solution $\hat{\mathbf{w}}$ of a classification problem is the center of the largest hypersphere within the gray cone $y_i(\mathbf{w}^T \mathbf{v}) > 0$ , $i = 1 \dots n$ . The bisection of the version space $\mathcal{V}$ can be approximated by cutting through the current solution $\hat{\mathbf{w}}$ , i.e., $\hat{\mathbf{w}}^T \mathbf{v} = 0$ . . . . .	106
5.2	Penalty <sub>ML</sub> and Penalty <sub>SL</sub> in the range $u \in [-5, 5]$ for preference realization; $\theta = 1$ , $h = 1$ . . . . .	115
5.3	Penalty <sub>ML</sub> and Penalty <sub>SL</sub> in the range $u \in [-2, 2]$ for preference identification; $\sigma_\varepsilon = 1$ , $h = 1$ . . . . .	128

5.4	Demonstration of active learning for preference identification. Each block on the 2D space represents a design. The black blocks represent designs such that $f(\mathbf{x}) < 0$ and the white ones represent $f(\mathbf{x}) > 0$ . The circled blocks are designs with $y = -1$ and dotted ones with $y = 1$ .	134
6.1	We jump from vertex $n_s$ to vertex $n_{s+\delta}$ if there is no alternative path from $n_s$ and through $n_{s+1}$ .	140
6.2	Concept of JUMP1: Jumping to a conclusion (leaf vertex) based on previous knowledge, and search from there.	141
6.3	JUMP1 algorithm: From the current status $n_s$ , find a set $\mathbf{N}_s$ from $T_t$ that is most similar to $n_s$ . Find the leaf vertex set $\mathbf{N}_s^*$ accessible from $\mathbf{N}_s$ . Pick the next query design as the one from $\mathbf{N}_s^*$ that has the highest average path length $\bar{P}$ .	142
6.4	Setup of the 2D demonstration for JUMP1: The test optimal solutions are uniformly distributed on the shaded locations. The indices are of ascendant order starting from the left down corner as 1 and ending at the top right corner as 100.	143
6.5	Average path lengths for 10 batches of random tests with and without JUMP1. This comparison shows that JUMP1 can effectively reduce the search cost with different $d$ .	144
6.6	Histogram of path lengths from 200 random tests with and without the search tree generated from the experiment in Figure 6.5.	144
6.7	The evolution of search path for a preference function optimized at index 86. JUMP1 explores different paths and finds better ones than from the original search algorithm.	146
6.8	The average query size of the two clusters (optimal solutions at $\{\mathbf{x}_{15,24,25,33,34,35,36,44,45,55}\}$ and in $\{\mathbf{x}_{68,77,78,86,87,88,89,97,98}\}$ ) along number of batches of tests.	146
6.9	JUMP2 algorithm: From the current status $n_s$ , find a set $\mathbf{N}_s$ from $T_t$ that is most similar to $n_s$ . Find the leaf vertex set $\mathbf{N}_s^*$ accessible from $\mathbf{N}_s$ . Pick the next query design as the one from $\mathbf{N}_s^*$ that has the lowest average path length to all other leaf vertices accessible from it.	147
6.10	Performance of JUMP1 and JUMP2 on 1000 random tests with update frequency: 1, 10, 100 tests per update. Both algorithms switch to heuristic search after every 2 iterations. For visualization purpose, the 1000 test results are grouped and we show the average query size of each group.	149
6.11	Performance of JUMP2 and JUMP3 on 1000 random tests. Both algorithms update their search strategy after every test and switch to heuristic search after every 2 iterations. The figure is segmented to show performance of JUMP3 with different initial guesses. For visualization purposes, the 1000 test results are grouped and we show the average query size of each group.	150

6.12	Performance of JUMP3 with fixed and adaptive initial guess change frequency on 1000 random tests. Both algorithms update their search strategy after every test and switch to heuristic search after every 2 iterations. The figure is segmented to show performance with different initial guesses. For visualization purpose, the 1000 test results are grouped and we show the average query size of each group. . . .	152
6.13	Densities of the optimal designs along time. The five figures show how the optimal designs are distributed at different stages of the experiment. The grey scale indicates the frequency of occurrence. . .	153
6.14	Performance of JUMP2, JUMP3 and the default algorithm on 1000 random tests. Both JUMP2 and JUMP3 update their search strategy after every test and switch to heuristic search after every 2 iterations. For visualization purposes, the 1000 test results are grouped and we show the average query size of each group. . . . .	153
6.15	A situation where optimal designs are not clustered. The grey scale indicates the frequency of occurrence. . . . .	154
6.16	Performance of JUMP2, JUMP3 and the default algorithm on the 1000 random tests with optimal designs located in Figure 6.15. Both JUMP2 and JUMP3 update their search strategy after every test and switch to heuristic search after every 2 iterations. For visualization purposes, the 1000 test results are grouped and we show the average query size of each group. . . . .	155
6.17	Performance of JUMP2, JUMP3 and the default algorithm on the 1000 random tests on a design space with 5 dimensions and 3 levels for each. The optimal designs of these random tests are uniformly generated on the entire space. Both JUMP2 and JUMP3 update their search strategy after every test and switch to heuristic search after every 7 iterations. For visualization purposes, the 1000 test results are grouped and we show the average query size of each group. . . .	155
6.18	Performance of JUMP2, JUMP3 and the default algorithm on the 1000 random tests on a design space with 5 dimensions and 3 levels for each. The optimal designs of these random tests are uniformly generated on a small set of indices of the space. Both JUMP2 and JUMP3 update their search strategy after every test and switch to heuristic search after every 7 iterations. For visualization purposes, the 1000 test results are grouped and we show the average query size of each group. . . . .	156

## LIST OF TABLES

### Table

1.1	Overview of related work . . . . .	23
3.1	Means and standard deviations of the final error from SVM Search, GA and Random Search (Lower values are better) . . . . .	57
3.2	Impact of the weights of the weighted-sum merit function (Lower values are better) . . . . .	63
3.3	Impact of the spread of the expected improvement merit function (Lower values are better) . . . . .	64
3.4	Impact of dimensionality and spread of the utility (Lower values are better) . . . . .	65
5.1	Generalization errors of active learning, $D$ -optimal and random (Lower values are better) . . . . .	121
5.2	Generalization errors under the linearity assumption . . . . .	123
5.3	$p$ -values of $H_0 : \hat{w}_i = 0$ for estimators on linear and polynomial preferences (Highlighted are significantly lower $p$ -values) . . . . .	124
5.4	Robustness of active learning (Lower values are better) . . . . .	125
5.5	Preference identification test settings . . . . .	135
5.6	Preference identification test results . . . . .	135

## ABSTRACT

Understanding user preference has long been a challenging topic in the design research community. Econometric methods have been adopted to link design and market, achieving design solutions sound from both engineering and business perspectives. This approach, however, only refines existing designs from revealed or stated preference data. What is needed for generating new designs is an environment for concept exploration and a channel to collect and analyze preferences on newly-explored concepts. This dissertation focuses on the development of querying techniques that learn and extract individual preferences efficiently. Throughout the dissertation, we work in the context of a human-computer interaction where in each iteration the subject is asked to choose preferred designs out of a set. The computer learns from the subject and creates the next query set so that the responses from the subject will yield the most information on the subject's preferences. The challenges of this research are: (1) To learn subject preferences within short interactions with enormous candidate designs; (2) To facilitate real-time interactions with efficient computation.

Three problems are discussed surrounding how information-rich queries can be made. The major effort is devoted to *preference elicitation*, where we discuss how to locate the most preferred design of a subject. Using efficient global optimization, we develop search algorithms that combine exploration of new concepts and exploitation of existing knowledge, achieving near-optimal solutions with a small number of queries. For design demonstration, the elicitation algorithm is incorporated with an online 3D car modeler. The effectiveness of the algorithm is confirmed by real user

tests on finding car models close to the users' targets. In *preference identification*, we consider designs as binary labeled, and the objective is to classify preferred designs from not-preferred ones. We show that this classification problem can be formulated and solved by the same active learning technique used for *preference estimation*, where the objective is to estimate a preference function. Conceptually, this dissertation discusses how to extract preference information effectively by asking relevant but not redundant questions during an interaction.