SUPERVISED LEARNING WITH IMPLICIT PREFERENCES AND CONSTRAINTS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Yunsong Guo

August 2010

SUPERVISED LEARNING WITH IMPLICIT PREFERENCES AND CONSTRAINTS

Yunsong Guo, Ph.D.

Cornell University 2010

In classical Combinatorial Optimization, a well-defined objective function is to be optimized satisfying a set of deterministic constraints. Approximation algorithms and heuristic methods are often applied to the problems proven to be difficult, or NP-Complete and beyond. However, in many real-world problem domains the objective (or utility or preference) function an individual is trying to optimize is not explicitly known. Furthermore, preferences can take many different forms and it is difficult to pre-define the correct format of the true utility function one is optimizing. To circumvent such limitations, we model these problems as machine learning tasks with implicit preferences that can be inferred from observations of the choices the individual made in the past. This approach contrasts with the traditional approach that learns the parameters of an utility or preference function, whose functional form is explicitly defined *a priori*.

We study a set of different learning problem domains in which the preferences, or utility functions, are not explicitly defined. These include structural learning, document citation ranking and resource capacity constraint satisfaction. Our goal is to accurately make predictions for future instances, assuming the same underlying preferences as those expressed in past observations, without explicitly modeling them. The new algorithms and techniques we propose to optimize our learning formulations are shown to be very effective. For situations where both the prediction accuracy and the explicit forms of preferences are important, we provide an Induc-

tive Logic Programming (ILP) based algorithm to extract the preferences from a "black-box" machine learning model for intuitive human interpretation.

BIOGRAPHICAL SKETCH

Yunsong Guo earned his PhD degree from the Computer Science Department of Cornell University in 2010. His main research interests lie in the field of Machine Learning with a focus on discriminative predictive models that utilize implicit user and system preferences, and Data Mining on large scale datasets to infer meaningful patterns. Before coming to Cornell, he received a B.Comp in Computer Science from the National University of Singapore in 2005. Yunsong grew up in Chengdu, a beautiful city in the southwest of China.

To my parents, whose support has enabled me to go this far.

负笈三万里
十载任漂流
南海烟如火
西陆雪满秋
故园长入梦
大道本通幽
笑看飞鸿影
逍遥过五洲

— 郭仲三

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my family members. In particular, my parents and my grandfather have enormously inspired me to pursue the path of becoming a scientist to the best of my ability.

My sincere gratitude goes to my advisor Carla Gomes, who not only gave me thorough guidance in the respective research domains with the highest standards, but also entrusted me with a significant amount of freedom to explore what science and life can offer during my enjoyable half-a-decade at Cornell. Among the numerous lessons she has taught me, one that I particular appreciate is the importance of being both determined and flexible. I especially want to thank Thorsten Joachims, whose in-depth knowledge and insightful comments have often enabled me to make various breakthroughs in many of my research projects. The same appreciation also goes to my other dissertation committee members, Bart Selman, David Williamson and Dexter Kozen, all of whom have provided me with invaluable guidance and support during the different stages of my Ph.D study. In addition, my special thanks are for Andrew Lim and Brian Rodrigues, who are my earlier academic mentors that continued to offer me much appreciated support.

Last but not least, I want to thank all my research collaborators and buddies, with whom numerous creative ideas are generated, thousands of lines of code are developed and tens of sleepless nights before deadlines are worked.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Implicit Preference Learning for Challenging Problems

Many real-world problems can be formulated as optimization problems. Such problems fall into one of two scenarios. In the first scenario, one knows exactly what needs to be achieved and what constraints must be satisfied. Such problems can be formulated as optimization problems with a deterministic objective function and explicit constraints. Well known examples in this category include the binary Knapsack problem and the Traveling Salesman Problem (TSP). The difficulty of solving these problems optimally varies substantially. Some problems entail closed form analytical solutions while others are proven to be NP-Complete and beyond. The above two example problems are both NP-Complete, but the binary Knapsack problem has a Fully Polynomial Time Approximation Scheme (FPTAS), while the TSP has no polynomial-time algorithm that approximates the optimal solution bounded by a ratio which is a function of the input size unless P=NP [44]. Once a discrete optimization problem is identified to be "hard", or NP-Complete, approximation algorithms and heuristic methods are often developed instead of finding an exact solution.

In the second scenario, the objective function of a problem can hardly be unambiguously formulated. For example, when we are choosing holiday gifts for our friends, we do not have a naturally-defined utility to maximize. Instead, we can often infer useful information from our friends' past purchase histories of what gifts may please them most. It is therefore more appropriate to model these problems as preference learning tasks in the machine learning domain. The term "preference"

has different meanings in varying contexts. It can represent, for instance, the pairwise ranking of similar patents or an optimal subset selected from a larger ground set of commodities. In this setting the ultimate goal is to predict the decisions an individual would most likely make in the future based on the historical choices the same individual or another similar group of individuals have made in past similar situations.

Preference learning methods can be categorized into two groups: explicit learning methods, in which the preference function is modeled as a parametric function on the training dataset; and implicit learning methods, which make no assumption on the underlying utility function but aim to mimic the behavior of users. All such machine learning problems can also be divided into either the structured learning or non-structured learning domains. In structured machine learning, the prediction of a model is a structure, e.g., a tree, a sequence and a set. Usually there are an exponential number of possible prediction outcomes for an instance, which is one of the main difficulties to solve problems in this domain. Non-structured learning problems do not have explicitly structured input and output. The traditional multiclass classification and learning to rank tasks are examples of non-structured learning problems.

The main research focus of this dissertation is on problems where an individual only has implicit preferences with no explicitly defined function form for the utility function he or she is trying to optimize. To make things worse, not only the preferences but also constraints can be implicit, as will be seen in chapter 3 when the problem of learning with resource capacity constraints is studied. Fortunately, although we do not know the explicit preferences, we can infer useful information from historical observations: what our friends bought before provides valuable in-

formation when we are choosing gifts for them; the waterpoints visited by a herder in the past few years indicate what properties the herders value in a waterpoint in the coming year; and the composition of stocks in a portfolio chosen by a hedge fund manager reveals what other new stocks he may pick after additional Initial Public Offerings (IPOs) in the future. In this dissertation we are interested in building implicit models to infer a user's behavior in future instances (which is a direct consequence of the user's preferences) as close as possible, over a range of structured and non-structured learning problems.

1.2 A Variety of Learning Problems Studied

The content of this dissertation is based on several published research papers and ongoing research projects at the time of writing. While special-purpose heuristics are sometimes effective when the exact objective function to optimize and deterministic constraints to satisfy are common knowledge, a machine learning approach is more appropriate if such knowledge is expensive or impossible to obtain. In particular, structural learning problems also arise in the domain of learning with implicit preferences. Instead of predicting a simple numeric label for an instance, as in the various classification and regression formulations, some problems require a learning model to be able to predict more complex structures. They require different techniques to solve, as one typical difficulty is the exponential number of possible predictions for each instance. Chapter 2 considers the problem of learning an optimal subset from a larger ground set of items, where the optimality criterion is defined by an unknown preference function. We model the problem as a discriminative structural learning problem and solve it using a Structural Support Vector Machine (SSVM) that optimizes a "set accuracy" performance measure represent-

ing set similarities. Our approach departs from previous approaches since we do not explicitly learn a pre-defined preference function. Experimental results on both a synthetic problem domain and a real-world face image subset selection problem show that our method significantly outperforms previous learning approaches for these problems.

In some cases not only are the preferences implicit, but the constraints one needs to satisfy are also not available at the individual level. Chapter 3 studies the problem of learning while enforcing the resource capacity constraints. In this problem setting, each instance requires a certain amount of a resource and each class has a finite resource capacity. The sum of the required resources from all instances assigned to a class should not exceed the class resource capacity. Notably this constraint has to be enforced at the global level rather than the individual instance level. The goal is to learn a model that classifies as accurately as possible for new instances that satisfy the class resource capacity constraints. Like the previous example problems, the individual's utility function is unknown to us and classification predictions will be based on historical preference observations. This problem is representative of a larger class of problems. It was motivated by a real problem concerning African pastoral herding in which policymakers have to assign tasks or activities to resources with capacity constraints, taking into consideration the historical preferences of the herders. We propose a SVM-based non-convex optimization model, SVM_{cap}, and solve it using a gradient descent method. We also propose two new evaluation metrics for resource capacity violations. Experimental results on both the pastoral dataset and a synthetic forest fire dataset show that our method can provide results as accurate as the benchmark SVM classification method, and reduce the amount of resource capacity violation by as much as 90%.

Heterogeneous implicit preferences from various parties may co-exist in the same learning problem. One such example concerns the choice of what patents to cite by the applicant and the examiner in a patent application. In chapter 4 we propose an approach for automatically ranking structured documents applied to patent prior art search. Our model incorporates margin constraints that directly capture the specificities of patent citation ranking without knowing the exact preferences of either the applicant or the examiner. Our learning model combines patent domain knowledge features with meta-score features from several different general Information Retrieval methods. The training algorithm is an extension of the Pegasos algorithm [94] with performance guarantees, effectively handling hundreds of thousands of patent-pair judgements in a high dimensional feature space. Experiments on an essential wireless patent dataset show that we can perform on average 30%-40% better than many other state-of-the-art general-purpose Information Retrieval methods for the NDCG evaluation measure.

The previously described machine learning approaches share a common trait: all learn to predict as accurately as possible without explicitly formulating the user's preferences in the process. These approaches are important because in terms of practical prediction accuracy, they are often superior to models that assume a fixed form of preferences and learn to decide the preference parameters via training. However, since these models only assume implicit preference during the training and testing phases, one would not be able to recover a more tangible form of what the preferences really look like. In order to overcome this drawback, we propose a novel method in chapter 5 that is able to "explain" a learned opaque machine learning model in terms of Horn logic [54]. We developed an Inductive Logic Programming (ILP) based framework, ExOpaque, that is able to extract a set of Horn clauses from an arbitrary opaque machine learning model. The extracted Horn

clauses can describe the behavior of the opaque model with a high fidelity measure while maintaining the simplicity of the clauses for human interpretations. In addition, we propose a new method that uses generated artificial training examples and high-accuracy opaque models to boost the prediction accuracy of an ILP system without the aid of any background knowledge otherwise required by traditional ILP systems. Using this method we can interpret many "black-box" machine learning models and summarize them as a set of logic rules that humans can easily understand. These logic rules are naturally an explanation of the implicit preferences the models learned from.

In section A of the appendix we will take a different perspective to look into a real-world logistic planning problem involving multinational Free Trade Agreements. In this problem the exact preferences and constraints are common knowledge, therefore we can build a large-scale neighborhood search heuristic algorithm. A numerical study is conducted to verify the efficiency of our solution approach. This problem and our algorithm highlight the different solution approaches from the earlier chapters where preferences are all implicit.

1.3 Research Contributions

The research contributions from this dissertation are summarized below:

- We propose a structural learning framework for predicting sets of objects from implicit preferences. We show that our approach can accurately learn the concept of various kinds of implicit preference tasks where the number of possible candidate subsets is in the exponential scale. Our subset predictions closely resemble the actual selected subsets of individuals, outperforming another state-of-the-art model that formulates individuals' subset selection preferences explicitly. This work was published in the 21st International Joint Conference on Artificial Intelligence (IJCAI), 2009 [45]. More generally, we have conducted a survey comparison of different leading structural machine learning algorithms on sequence labeling tasks, which was published in the 24th International Conference on Machine Learning (ICML), 2007 [84].
- A structured document ranking algorithm, SVM_{pr} is proposed to automatically recommend relevant documents for citation purposes. When applied to the patent citation problem, SVM_{pr} showed remarkable improvement from several general-purpose information retrieval methods in terms of the NDCG score. The material is based on work published in the 21st International Joint Conference on Artificial Intelligence (IJCAI), 2009 [46].
- When resource capacity constraints are considered, traditional classification problems are more difficult to solve as the additional constraints have to be enforced often at the global level. We formulated this new problem as a non-convex quadratic optimization problem in chapter 3. A primal optimization technique is used to solve the optimization problem and a greedy

heuristic as well as an exact optimal procedure are following the main learning algorithm to minimize resource capacity violations. Experiments on both real and synthetic datasets result in insights that demonstrate our method to be an efficient one to incorporate the resource capacity constraints while maintaining high-accuracy predictions compared with multi-class SVM.

- A very large-scale neighborhood (VLSN) search heuristic with simulated annealing for the NP-Complete Free Trade Agreement production logistics problem was developed, empirically outperforming CPLEX using the exact integer programming formulation. This work is published in the European Journal of Operational Research (EJOR), 187(2), 2008 [47]. Our study on some other combinatorial optimization problems with explicit preferences and constraints has been published in venues such as the Journal of the Operational Research Society (JORS) [49], and Mathematical and Computer Modeling (MCM) [48].

- In order to learn to explain opaque machine learning models in a human interpretable manner, we built a general framework, ExOpaque, which takes the input to a black-box model and the model's output to mimic the model's predictions using a sequence of Horn clauses. The Horn clauses which are logic rules with interpretable meanings. ExOpaque uses Inductive Logic Programming (ILP) to "translate" an arbitrary machine learning model into Horn clauses with a high fidelity measure. Content from chapter 5 has been published in the 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI), 2007 [50].

- Part of our research effort concentrates on developing general similarity and distance measures to better reflect the true metric one is concerned with and improve the learning algorithm accuracy. For example, in chapter 2 a new

set similarity measure that differs from two traditional measures is proposed and analyzed. Subsequent optimization is based on the new measure to provide superior performance compared with benchmark methods. In addition, our work on general metric learning for better prediction accuracy was published in the 18th European Conference on Machine Learning and Practice of Knowledge Discovery in Databases (ECML/PKDD), 2009 [85].

CHAPTER 2

PREDICTING OPTIMAL SUBSETS AS A STRUCTURAL LEARNING PROBLEM

Many interesting problems can be abstracted as finding an optimal subset of items that maximizes the utility induced by a preference function defined over individual items or sets of items, given a larger ground set. Examples range from real-world scenarios, such as recommending movies from a list of titles or buying the weekly groceries from a supermarket among aisles of alternatives, to the more general binary knapsack and set covering problems. We note that selecting a subset of items from a larger ground set based on a preference function is different from ranking the items and selecting the top items to include in the subset, since items may interact with each other. For example, when selecting what to take on a backpacking trip, a bottle of water can be most essential (thus ranked highest), but it may not be as valuable as the combination of juicy fruits, which are substitutes of water, and a piece of solid food, whose individual ranks may be well below that of the water. In addition, because items in the optimal subset are normally equally preferred, learning a total ranking of items may not be possible. The problems of preference representation have been studied in the literature [12, 34, 41, 36]. The problem of computing the optimal subset, assuming a completely known preference function, has also been studied [11, 88]. In addition, recent work has shown how, given a pre-defined parameterized preference function, the parameters can be automatically learned from historical data. A heuristic greedy method is then used to select an item subset from a larger ground set, trying to maximize the utility associated with the pre-defined (with learned parameters) preference function [33].

In our work we assume that for each ground set of items, a *single* most preferred optimal subset exists according to some (unknown) preference function, and we want to build a model that can predict the subset as close as possible to the optimal one. Given the wide range of (unknown) preference functions, it is difficult to define a priori the *right* parameterized family of functions. In order to circumvent this intermediate step, we discriminatively learn to predict the optimal subsets from historical datasets, using structural machine learning, without explicitly assuming a pre-defined parameterized preference function over sets of items. As no explicit preference function is assumed during learning, our performance evaluation is based on a set similarity measure. To the best of our knowledge, our approach is the first one to formulate this problem as the task of learning to predict an optimal subset that maximizes a set similarity measure, without an explicit preference representation. We solve the structural learning for the optimal subset selection problem using Structural Support Vector Machines (SSVM), for which we guarantee that learning is performed in polynomial time and testing inference is exact, even with an exponential number of candidate subsets. Our experiments are conducted using both a synthetic dataset and a real-world face image dataset. A total of 8 subset selection tasks are used for the evaluation. Our method outperforms the previous approach significantly in all tasks in terms of several set similarity measures.

2.1 Optimal Subset Learning

2.1.1 Problem Formulation

Let $\mathcal{X}$ denote the space of possible ground sets, and $\mathcal{S}(x)$ the space of subsets given a ground set $x \in \mathcal{X}$. We assume all possible subsets are legitimate so $|\mathcal{S}(x)| = 2^{|x|}$. In addition, let $\mathcal{S}(\mathcal{X}) = \bigcup_{x \in \mathcal{X}} \mathcal{S}(x)$. An example instance is illustrated in Figure 2.1,where the ground set consists of six face images and the two optimal subsets correspond to two very different kinds of preferences on the same ground set.

The learning problem can be formulated as follows: we are given n supervised training sets (x_i,y_i), where x_i is the i^{th} ground set of items and y_i is the optimal subset computed from some implicit user preferences according to the ground set x_i with $y_i \subseteq x_i$; each item $I \in x_i$ is represented as a real valued feature vector; the goal is to learn a function F: $\mathcal{X} \rightarrow \mathcal{S}(\mathcal{X})$ which, given a new unseen ground set x, predicts the subset $\widehat{y}$ of x that minimizes the expected value of a set similarity loss $\Delta(y, \widehat{y})$, where y is the optimal subset computed from x if we knew the underlying implicit user preferences. As the distribution of instances in $\mathcal{X}$ is largely unknown, we instead construct the function f: $\mathcal{X} \rightarrow \mathcal{S}(\mathcal{X})$ that minimizes the empirical risk

$$R^{\Delta}(f) = \frac{1}{n} \sum_{i=1}^{n} \Delta(y_i, f(x_i)) \tag{2.1}$$

where $\Delta : \mathcal{S}(\mathcal{X}) \times \mathcal{S}(\mathcal{X}) \rightarrow \mathbb{R}$ is the loss function that penalizes predicting $f(x_i)$ as the optimal subset y_i.

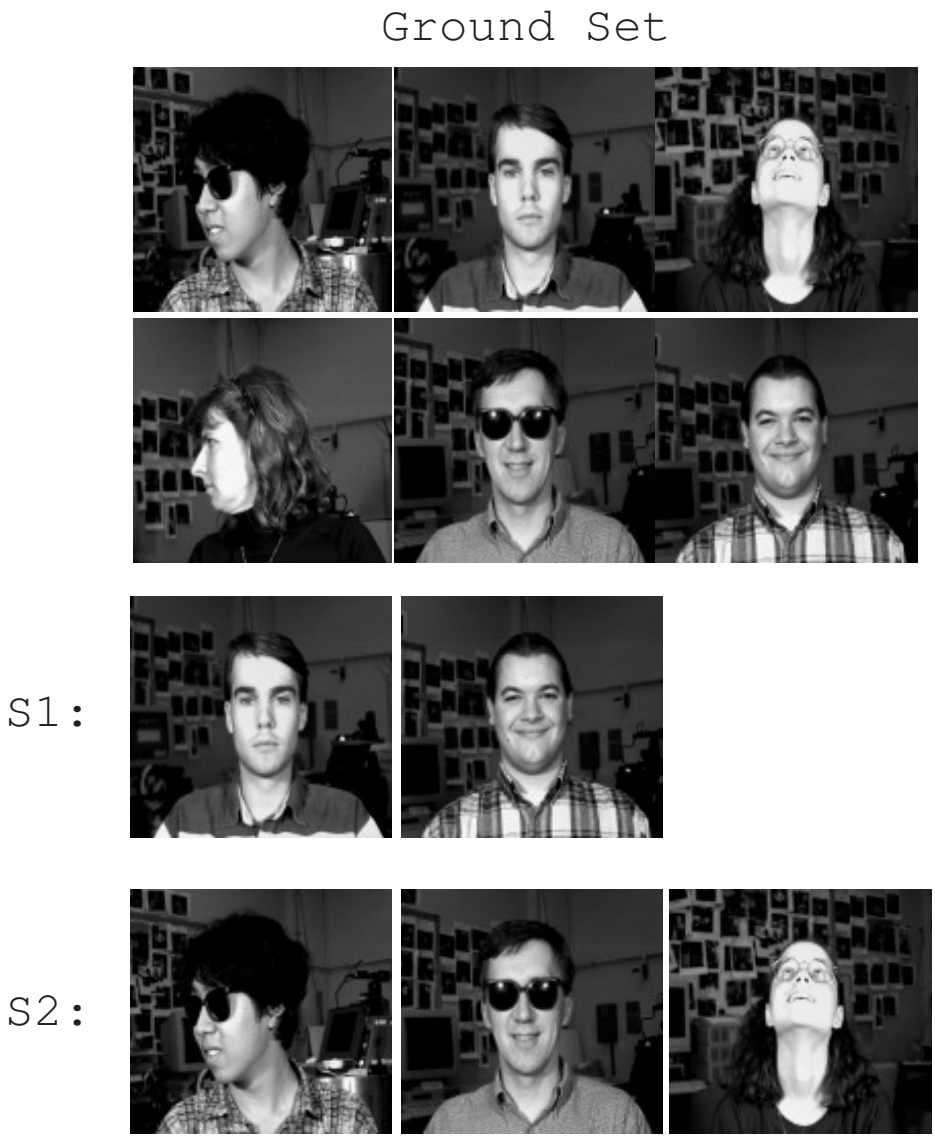

Figure 2.1: Optimal Subset Example
The two subsets correspond to two user preferences: $S1$-all face images that can be used as passport photos (looking straight and no sunglasses); $S2$-if there are more angry faces than happy faces, choose all angry faces, otherwise all happy faces that are with sunglasses or looking up.

2.1.2 The Loss Function Δ

We define the loss function $\Delta(\widehat{y}, y_i) = 1 - \sigma(\widehat{y}, y_i)$ with $\sigma(y_1, y_2)=\frac{|y_1 \cap y_2|}{\max(|y_1|,|y_2|)}$, assuming $\max(|y_1|, |y_2|)>0$. $\sigma(y_1, y_2)$ is our measure for set similarity which we refer to as *set accuracy*. Note that $\sigma(y_1, y_2)$ is the minimum of the precision and recall scores given sets y_1 and y_2. Averaged over multiple instances, set accuracy is upper bounded by the average precision and recall scores. This similarity measure differs from two other often used set similarity measures, namely the Jaccard Index defined as $J(y_1, y_2) = \frac{|y_1 \cap y_2|}{|y_1 \cup y_2|}$, and the Dice's coefficient $D(y_1, y_2) = \frac{2 \times |y_1 \cap y_2|}{|y_1| + |y_2|}$, which can be shown to be equal to the F1-score in our problem setting.

The differences of the loss functions are illustrated in Figure 2.2, where the true optimal subset size is 20, and the size of the overlap of the optimal and predicted subsets is a constant 10. Both Jaccard and F1 loss increases as the predicted size increases, while the set loss is indifferent until the size of the predicted subset reaches that of the optimal subset. We prefer set loss in our problem setting because it encourages the model to aggressively select more items into the subset to achieve a larger overlap between the predicted set and the optimal one when the sizes of two sets do not differ significantly. For example, consider the instance where the true optimal subset is {1,2,3} for a much larger ground set, and the two predicted subsets are $\widehat{y}_1 = \{1\}$ and $\widehat{y}_2 = \{1, 2, 4, 5, 6\}$. Intuitively we prefer $\widehat{y}_2$ over $\widehat{y}_1$ as it contains more information included in the optimal set. The Jaccard loss and F1-loss will have the same penalty on both predicted subsets, while set loss penalizes $\widehat{y}_1$ more. Nevertheless, we will also include the Jaccard Index and the F1-score as performance evaluation metrics in the experiment section.

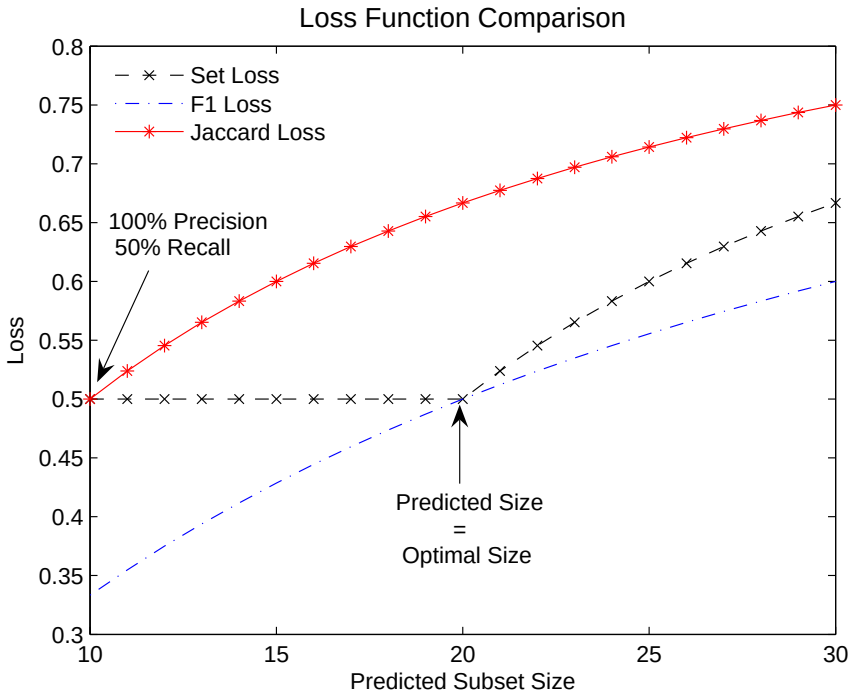

Figure 2.2: Loss Function Comparison

2.1.3 Structural Learning Formulation

The optimal subset learning problem naturally falls into the structured learning regime, as the input and output are sets of objects that can interact with other objects in the ground set. We propose to solve it using structural SVM for optimal subset learning (SVM_{OS}), formulated as follows:

Optimization Problem SVM_{OS}

$$min_{w,\xi\geq 0} \quad \frac{1}{2}\|w\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \tag{2.2}$$

$$s.t. \quad \forall 1 \leq i \leq n, \forall \widehat{y} \in \mathcal{S}(x_i) :$$

$$w^\top \phi(x_i, y_i) \geq w^\top \phi(x_i, \widehat{y}) + \Delta(\widehat{y}, y_i) - \xi_i \tag{2.3}$$

$\phi(x, y)$ in (3.1) is the feature map jointly defined by a ground set and a candidate subset. It is a well-known result in the multiclass setting that for any feasible solution w and ξ, $\sum_{i=1}^{n} \xi_i$ is an upper bound on the training loss, independent of the prediction function f. The same result holds for our structural learning setting, as we can use simple arithmetic to show that

$$\sum_{i=1}^{n} \xi_i \geq \sum_{i=1}^{n} \Delta(f(x_i; w), y_i) \tag{2.4}$$

After the weight vector w is optimized in the above quadratic formulation, the linear discriminant score for predicting subset y as the optimal subset for ground set x is $\mathcal{T}(x, y; w) = w^\top \phi(x, y)$, and the function f we use to predict subsets for an arbitrary ground set is:

$$f(x; w) = \arg \max_{y \in \mathcal{S}(x)} \mathcal{T}(x, y; w) \tag{2.5}$$

2.1.4 Optimizing for SVM$_{OS}$

There are an exponential number of constraints ($\sum_{i=1}^{n} 2^{|x_i|}$) in (3.1), which is the main difficulty in solving SVM$_{OS}$ optimally. We use Algorithm 1, a cutting plane algorithm proposed in [99] to solve SVM$_{OS}$ within a tolerance ε from optimality. Algorithm 1 iteratively adds the most violated constraint to the initially empty working set of constraints to be optimized.

Theoretically, given $\overline{R} = max_{i,y} \|\phi(x_i, y_i) - \phi(x_i, y)\|$, $\overline{\Delta} = \max_{i,y} \Delta(y_i, y)$ and a fixed ε, Algorithm 1 stops after adding at most $\max\{\frac{2n\overline{\Delta}}{\varepsilon}, \frac{8C\overline{\Delta}\overline{R}^2}{\varepsilon^2}\}$ constraints, with the solution ε-close to optimality. The proof can be found in [99].

Consequently, the polynomial runtime of Algorithm 1 is ensured if the "most

Algorithm 1: Cutting plane algorithm to solve SVM_{OS}

Input: $(x_1, y_1), \ldots, (x_n, y_n), C, \epsilon$
$P_i \leftarrow \emptyset$ for all $i = 1, \ldots, n$
Initialize $\mathbf{w}$
repeat
  for i = 1,. . .,n **do**
    $H(y) \equiv \Delta(y, y_i) + \mathcal{T}(x, y; w) - \mathcal{T}(x, y_i; w)$
    compute $\widehat{y} = \arg\max_{y \in \mathcal{S}(x_i)} H(y)$
    compute $\xi_i = \max\{0, \max_{y \in P_i} H(y)\}$
    if $H(\widehat{y}) > \xi_i + \varepsilon$ **then**
      $P_i \leftarrow P_i \cup \{\widehat{y}\}$
      $w \leftarrow$ optimize the dual of SVM_{OS} over
      $P = \cup_i P_i$
    end if
  end for
until no P_i has changed in the current iteration

violated" constraint, among the exponential number of constraints, i.e.,

$$\arg \max_{\widehat{y} \in \mathcal{S}(x)} \mathcal{T}(x, \widehat{y}; w) + \Delta(\widehat{y}, y) \tag{2.6}$$

can be computed in polynomial time.

In our problem setting, each item I is represented as a real valued vector $\upsilon(I) \in \mathbb{R}^k$. We define the feature map to be:

$$\phi(x, y) = \sum_{I \in y} \upsilon(I) - \sum_{I \in x - y} \upsilon(I) \tag{2.7}$$

If item I is selected in subset y, it contributes $w^\top \upsilon(I)$ to $\mathcal{T}(x, y; w)$, otherwise its contribution is $-w^\top \upsilon(I)$. For a given training set (x,y) and arbitrary $\widehat{y} \in \mathcal{S}(\mathcal{X})$,

we define $a = |y \cap \widehat{y}|$, the size of overlap between subset y and $\widehat{y}$; and $b = |\overline{y} \cap \overline{\widehat{y}}|$, the size of overlap of complement of y and $\widehat{y}$ in x; and $n = |x|$, the size of the ground set. We can show:

$$\Delta(\widehat{y}, y) = 1 - \frac{a}{max(|y|, n - |y| + a - b)} \tag{2.8}$$

and solve (2.6) optimally in polynomial time $\mathrm{O}(|x|^2)$ using Algorithm 2 in [58] as follows: since both a and b can only take values in {0,...,n}, $\Delta(\widehat{y}, y)$ can have at most $O(n^2)$ different values. In addition, if a and b are fixed, we can construct $\widehat{y}$ to maximize $\mathcal{T}(x, \widehat{y}; w)$ in (2.6) easily: we rank all items in x according to the score $w^\top \upsilon(I)$. Select a highest scored items in y to include in $\widehat{y}$, and do not select any of the b lowest scored items in $x - y$ in $\widehat{y}$. For the rest $n - a - b$ items, choose to include the item in $\widehat{y}$ if and only if it is not in y. By iterating through legitimate values of a and b, we can find the most violated constraint in $O(n^2)$ time with efficient implementation. During the testing phase, the optimal subset that maximizes equation (2.5) is $\{I \in x\colon\ w^\top \upsilon(I) > 0\}$.

2.2 Preference Formalism

So far we have concentrated on the structural learning algorithm, while being vague about the definition of preferences. Naturally, a preference $\mathfrak{p} : \mathcal{X} \to \mathcal{S}(\mathcal{X})$ is a function that maps any ground set in $\mathcal{X}$ to its optimal subset in $\mathcal{S}(\mathcal{X})$. Without loss of generality, we also assume that given a subset $y \in \mathcal{S}(x)$, it is easy to test if $y = \mathfrak{p}(x)$. In fact, we will show that the decision version of computing the optimal subset given the preference $\mathfrak{p}$ and a ground set x is **NP-complete** in the following.

Definition. Subset Computation Problem (SCP)

Input: ground set x, preference $\mathfrak{p}$, an item $I \in x$

Decision Question: Is $I \in \mathfrak{p}(x)$*?*

Lemma. *SCP is* **NP-*Complete*.**

Proof. See Appendix B. □

Therefore, for the experiment dataset construction, we further restrict that given preference $\mathfrak{p}$ and ground set x, the optimal subset can be efficiently computed, for example in low order polynomial time. This restriction on the preference is for computational purposes of the experiment dataset, but it is not a limitation of our learning algorithm. If an oracle exists to produce the optimal subset given ground set x and unrestricted preference $\mathfrak{p}$, SVM_{OS} is able to learn to predict the optimal subset.

2.3 Related Work

[36] learns a linear discriminant ordinal utility function from a set of qualitative preference statements by a non-parametric model similar to hard margin SVM ordinal regression. The obtained utility function essentially specifies an item-wise preference partial ordering. In addition to the fact that our model is based on structural learning, their focus is not on learning an optimal subset.

[12] proposed a general way to specify preferences over sets of items. It first defines item properties from the attribute values of an item. The set properties are defined based on item properties and the set preferences are specified over set property values using TCP-nets [13] that yield a partial ordering. While the emphasis of [12] is the representation of set preferences, in our study we emphasize

not the exact representation of set preferences, but the discriminative model in order to learn the most preferred set of items to maximize a set similarity measure, as we assume preferences are implicit and can take various forms.

[11] extends the work of [12]. They concentrate on the computational aspect of the NP-hard optimal subset selection after the set preferences are given in the formalism of [12]. They showed that they can improve the heuristic search over the property value method first pursued in [12], by carefully designed pruning mechanisms that efficiently rid many sub-optimal property combinations. They also proposed a new search over sets heuristic that deals with more general preference models. Their work clearly differs from ours, as no learning is involved and the preference model is pre-defined.

[110] learns to predict a subset of documents that is as diversified as possible using structured SVM. The diversity measure is defined as a linear function using a structured feature space that encodes the benefit of covering individual words. In this chapter we deal with a more general problem by optimizing the set accuracy metric, as diversity can be defined as a particular preference.

[34] represents the preference over sets of items by "depth" preferences and "diversity" preferences. The objective value of a given preference and a subset of items is a linear combination of its "depth" value and "diversity" value. While "depth" can be decomposed into a weighted sum of functions on item features, "diversity" is measured with a combination of item features and subset features. They also proposed several greedy methods to find optimal subsets with known preferences, and compared the objective value of subsets found by greedy and exhaustive search methods which showed that the greedy methods can find near optimal subsets much more efficiently than the exhaustive search. The preference

representation is considered more specific than [12] by [11].

[33] extends [34], in which a preference learning mechanism (denoted DDpref) is introduced to learn the preference parameters for the model proposed in [34]. DDpref learns depth preferences through kernel density estimation (KDE) [86] and diversity preferences by maximum *a posteriori* estimation. The experiments showed how DDpref improves the objective value of learned preferences as training set size increases. To our knowledge DDpref is the only existing method that learns user preferences from sets of objects and can predict optimal subsets for new ground sets of items. Therefore, we will compare our experimental results with DDpref's.

2.4 Experiment

2.4.1 Dataset Description

The main experiments are based on the CMU face image dataset we obtained through [6]. This dataset contains face images of 20 different people with various poses (straight, left, right or up); expressions (neutral, happy, sad or angry), and eyes (open or with sunglasses). Each person has 28 to 32 quality face images. Each grey-scale image contains 32*30 pixels and a pixel is represented using its integer intensity in the range [0,255]. To construct the experiment dataset for optimal subset learning, we separate the 20 people into two disjoint groups of size 12 and 8 to construct the training and testing sets. In order to generate one training example, we randomly sample 100 images of the 12 people, and compute the optimal subset using the preferences we will describe soon. The test set is generated in the same manner on the other 8 peoples' images. 200 training sets

Table 2.1: Dataset Size of Each Preference Learning Task

Experiment	n	m	card.	k	fea.
Face Images	200	100	100	variable	960
Toy Blocks	100	100	100	20	4

and 100 test sets, each with one ground set and an optimal subset, are generated for every preference task 1-5. A total of $2 * 10^4$ images (with repetitions) are used to construct the training set for each task.

The synthetic dataset was first used in [33]. This dataset concerns learning optimal subsets of toy blocks. Each toy block is described using 4 integer values to represent its size, number of sides, color and bins. For task 6-8, we obtained 100 training sets and 100 testing sets using the block generator from [33]. Each training and test set contains 100 blocks as the ground set, and the size of the optimal subsets are fixed at 20 due to the requirement of DDpref. The scale of the blocks dataset exceeds that used in [33].

Table 2.1 gives the summary of the experiment dataset size of a single preference learning task in terms of the number of training sets (n), the number of test sets (m), the number of items in each train/test set ($card.$), the size of optimal subsets (k) and the number of features to describe an item in the set ($fea.$).

The different underlying preferences to generate the dataset following our preference formalism in section 2.2 are listed below, with the first 5 preferences on the face images dataset and the last 3 on the toy blocks dataset:

1. (Independent of ground set) Choose all images that are open, straight, and

happy or neutral.

2. (Dependent) If the ground set has more images with people wearing sunglasses than open face, choose all left, otherwise choose all right.
3. (Dependent) If the ground set has fewer happy faces than sad ones, chose happy images; otherwise choose sad faces.
4. (Dependent, reverse of task 2) If the ground set has more left than right images, choose all open, otherwise choose all sunglasses
5. (Complex Dependent) If open and happy images consist of more than 30% of ground set, and sad and sunglass images are fewer than open and happy, choose all sad, otherwise all angry images.
6. (Blocks *Child*) Want to choose a variety of multi-colored, medium-sized blocks with few sides for a child to grasp.
7. (Blocks *Mosaic*) Want to create a mosaic with blocks of various shapes, with emphasis on simple, small blocks.
8. (Blocks *Tower*) Want to build a uniform tower with large, similar-sided blocks of the same color.

2.4.2 Set Accuracy Result

The results are summarized in Figure 2.3. All performance measures are averaged over the 100 test sets for each task. For the first five tasks on face images, SVM_{OS} reports the best set accuracy results by varying the C parameter from 10^{-6} to 10^{4}. We also included the results where the C parameter is chosen from 5-fold cross validation on the training set (denoted by SVM_{CV}). For DDpref, we report the best results obtained from extensively tuning the α and K parameters, where α is the

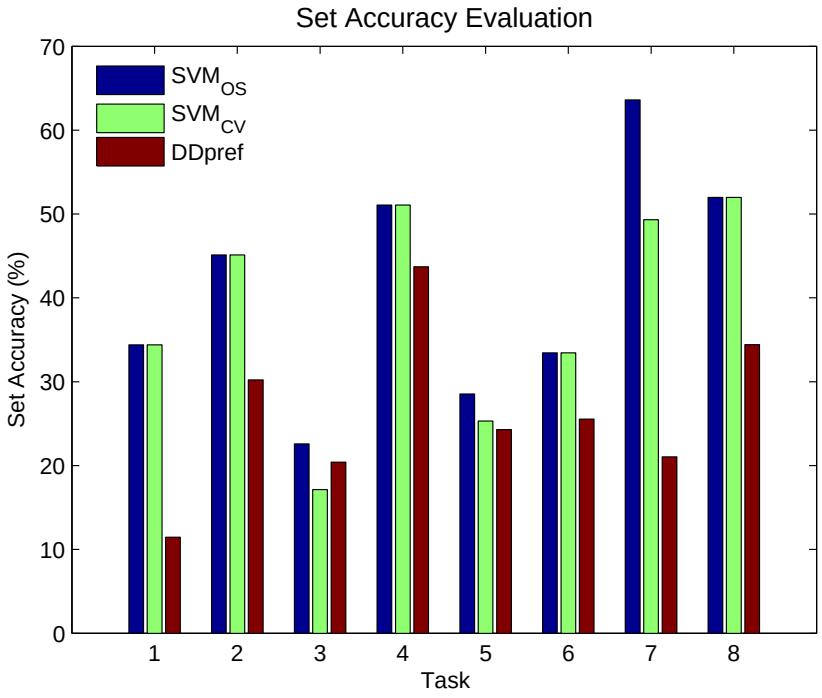

Figure 2.3: Set Accuracy Comparison

tradeoff constant between diversity and depth preferences and K is the required size of predicted subsets. The difference in performance between SVM$_{OS}$ and DDpref for all tasks except task 3 is significant at the 0.01 level with a signed rank test. The difference in task 3 is significant at the 0.1 level. SVM$_{OS}$ outperforms DDpref in all five tasks, while SVM$_{CV}$ outperforms DDpref in all tasks except for task 3, in which the cross validation happens to find a substantially suboptimal C setting. For other tasks, the cross validation on the training set often finds the best parameter. Some of the preference tasks on the image dataset, such as tasks 4 and 5, cannot be represented using our chosen feature map function from section 2.1.4. This is the case because our feature map essentially results in selecting items to include into the optimal subset based on the items' discriminant score independently from the rest of the set. This indicates that although our method provides better set accuracy measure compared with DDpref over these tasks,

adopting a more complex feature map function capable of representing preferences of the above-mentioned kind is likely to improve the results even further. We will discuss this issue in more detail at the end of this chapter.

For the three toy block tasks, we supplied DDpref with the true optimal subset size K=20, as it is a constant for the generation of the dataset. Thus, DDpref will always predict a subset with the same cardinality as the true optimal subset. All performance differences are significant at the 0.01 level. Again, both SVM$_{OS}$ and SVM$_{CV}$ predicted better subsets than DDpref in terms of the set accuracy measure.

2.4.3 More Evaluation Metrics

Results on Precision, Recall, F1 Score and Jaccard Index

Other than the proposed set accuracy measure, we also evaluated the experiment performance of SVM$_{OS}$ and DDpref using the other metrics discussed in section 2.1.2 to provide a more complete view, namely the precision and recall accuracy, the F1-score and Jaccard Index. All performance measures are calculated using the optimal subset and the predicted one from each test set, and are averaged over 100 test sets. Table 2.2 and 2.3 present the results when the best set accuracy is achieved for both SVM$_{OS}$ and DDpref.

We can conclude from Table 2.2 that SVM$_{OS}$ outperforms DDpref in all four metrics. The only exception is task 3, where DDpref achieves better recall and F1-score.

In Table 2.3, the precision, recall and F1 scores for DDpref are exactly the same

Table 2.2: Precision, Recall, F1-Score and Jaccard Index Performance on Face Dataset in Percentage

Task	SVM_{OS}				DDpref			
	Prec.	Rec.	F1	Jacc.	Prec.	Rec.	F1	Jacc.
1	35.98	51.39	40.59	27.69	11.50	19.77	14.54	8.45
2	45.91	70.88	54.12	38.95	30.40	48.99	37.52	23.26
3	23.75	35.17	27.07	16.67	20.40	*42.47*	*27.56*	16.18
4	69.95	51.19	58.43	42.12	49.00	48.49	48.74	31.79
5	28.54	60.92	37.79	24.23	24.33	52.24	33.20	20.38

Table 2.3: More Evaluation Metrics on Blocks Dataset

Task	SVM_{OS}				DDpref	
	Prec.	Rec.	F1	Jacc.	Prec/Rec/F1	Jacc.
6	33.44	49.75	39.80	26.02	25.55	16.05
7	64.63	74.05	68.59	52.95	21.04	12.44
8	52.03	73.85	60.73	44.48	34.42	23.63

in each task, as DDpref always predicts the subset with the same cardinality as the optimal subset for a fixed K=20. SVM_{OS} achieves higher scores for all metrics in the three tasks.

We note that both DDpref and SVM_{OS} also learn a preference ranking order over items. In this chapter, we focus on the performance of predicting the optimal subset and therefore do not compare the ranking performance, which can be defined separately.

Scalability Comparison

We study the scalability behavior by comparing the training time for both SVM_{OS} and DDpref. Apart from the fact that SVM_{OS} is implemented in C++ and DDpref in Java, we observed that SVM_{OS} is orders of magnitude faster than DDpref during training. Although the training time depends on the particular parameter settings, in general SVM_{OS} usually requires less than 10 minutes to train while the training time required by DDpref can be more than 10 hours on a machine with a 3.8GHz Intel CPU and 8GB memory. Figure 2.4 shows the detailed training time against different training set sizes for task 5, where the parameters are set to provide the best set accuracy. Both methods appeared to scale linearly in our experiments. During the testing phase, SVM_{OS} has a runtime linear in the number of test cases, and DDpref uses a greedy method to select the optimal subset from the learned preferences. Both SVM_{OS} and DDpref are efficient in the testing phase in terms of runtime.

2.5 Discussion

In this chapter we propose a new approach for the optimal subset selection problem. We model the problem as a structural learning problem that maximizes set similarity between the predicted subsets and the optimal ones. While previous work requires an explicit preference representation, our discriminative learning model SVM_{OS} only assumes the existence of implicit user preferences. Our model therefore does not need the intermediate step of learning the parameters of a predefined preference function. Experiment results on a variety of the subset selection tasks have shown that SVM_{OS} provides high quality testing results compared with

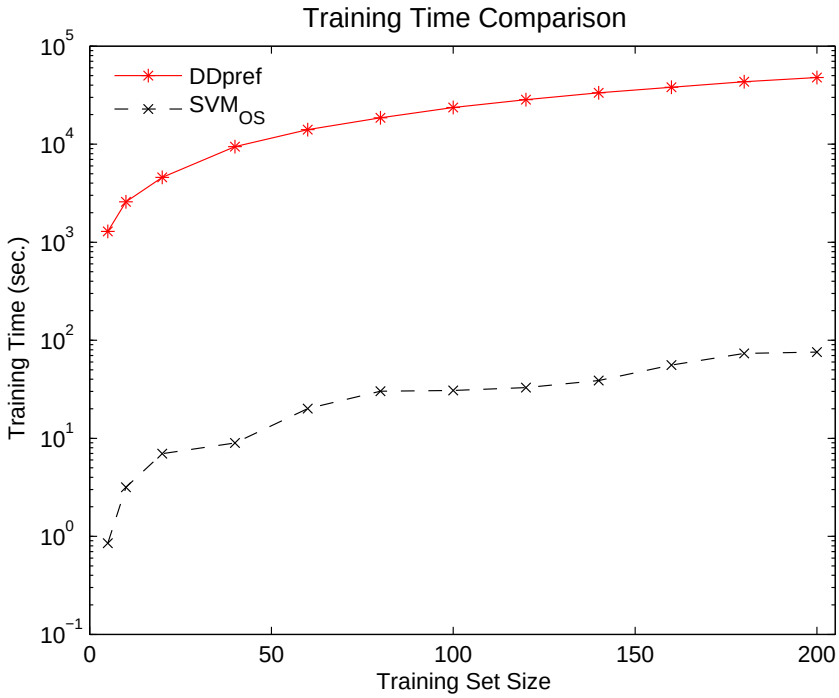

Figure 2.4: Scalability of SVM$_{OS}$ and DDpref

a recent learning model.

On the other hand, we notice that as a result of our defined feature map function in section 2.1.4, we are not modeling the preference dependencies among items *within* the optimal subset. Consequently the optimal decision of whether to include an item in the optimal subset can be made by considering the ground set of items alone. Even with this artifact our problem cannot be solved by a binary classification model because the set similarity measure and its corresponding loss function are not linearly decomposable. One important future direction is to include dependencies among items in the optimal subsets. However, additional complexities would appear when we try to optimize our quadratic formulation. We can investigate the use of new constraining factors on the feasible preference space to reduce the complexities, such as enforcing a sub-modularity condition.

Our problem of learning the optimal subsets assumes that the preferences, although implicit, are determined independently by each individual. When one's preference and utility depends on others' choices, the learning process must adapt by enforcing additional global constraints that are not expressible at the individual level. In the next chapter, we will study such a problem of learning with capacity constraints.

CHAPTER 3

LEARNING WITH RESOURCE CAPACITY CONSTRAINTS

An important class of allocation problems involves optimizing the assignment of agents to units over a relatively large, discrete set of locations, considering the agents' preferences, subject to soft-binding locational capacity constraints. The agents' preferences are only implicitly revealed by the agents' past behavior. Examples include the matching of fishing boats to harvesting locations, workers to firms, or as in the lead application considered in this chapter, the spatio-temporal allocation of animal herds to watering points during dry seasons in the arid and semi-arid regions of east Africa based on the past migratory movements of the pastoral nomadic populations. A characteristic feature of such problems is that a certain type of resource is demanded by an individual, and locations provide such a resource at varying degrees. One could consider modeling such problems as optimization problems, enforcing the natural constraint that the total demand of resource from individuals assigned to a location should not exceed the locational resource supply. However, such an optimization setting poses a major limitation: it does not capture the agents' preferences. Such preferences are in general difficult to define, since they are only implicitly observed in past historical data. Yet it would be desirable for researchers and policymakers to have access to reliable, *predictive* models that map past observable characteristics of individual units, locations and other entities into allocations of units over locations while enforcing capacity constraints, for unseen problem instances. In fact, in regard to the fishing application, Smith [96] argued that "structural attempts are essentially intractable" and "models that predict well, especially if they do well out-of-sample, can be used for policy analysis".

We propose to formulate this problem as a learning problem with resource capacity constraints. The training input includes the features of instances used in the classification task, with additional information on each instance's resource demand and each class's resource supply. The goal is to learn a model that efficiently makes *good* predictions for future instances. We propose a novel approach to the problem of learning with resource capacity constraints through an extension of Support Vector Machines [102]. In addition to the classical soft constraint used in multi-class SVM, we also introduce a soft resource capacity constraint at the location level. This leads to a non-convex quadratic optimization problem for which we develop efficient optimization algorithms. The general idea we adopt to deal with the additional set of non-convex constraints is to use an iterative gradient descent step whenever the resource capacity constraint is found to be violated. We test our proposed method, called SVM$_{cap}$, on two datasets, namely the real African pastoral dataset and a synthetic dataset with resource constraints generated from a real forest fire dataset. The results on both datasets show that our approach, SVM$_{cap}$, compared to the standard SVM classification method, can reduce the amount of resource capacity violation by as much as 90% without deteriorating accuracy. In addition, we will introduce a new post learning optimization method to further improve the learning performance on violation reductions. We also describe how SVM$_{cap}$ can be a useful tool for policymakers in deciding which new or existing locations to invest in.

3.1 Problem Formulation

The problem of learning with resource capacity constraints is very similar to the classical classification problem, except that now each instance has a positive resource demand, or load, and each class has a supply capacity. The goal of the new

problem is to make accurate predictions on a test dataset while requiring that the sum of total load of instances assigned to a class should not exceed the capacity of that class. In the training set, let $\mathbf{S}$ be the set of m classes. Class j has a resource capacity C_j, for $j \in \{1, \ldots, m\}$. $\mathbf{x}$=$\{(x_1, y_1, L_1), \ldots, (x_n, y_n, L_n)\}$ is the set of n input instances. For $i \in \{1, \ldots, n\}$, x_i is the vector of descriptive features of the instance; $y_i \in \mathbf{S}$ and L_i are the true class number and load associated with the instance.

We propose to solve the classification problem with resource capacity constraints using a SVM-based approach, SVM$_{cap}$, which incorporates the additional constraints and is formulated as follows:

SVMcap Optimization

$$min_{w,\xi \geq 0, \delta \geq 0} \quad \frac{1}{2}\|w\|^2 + \frac{l_1}{n}\sum_{i=1}^{n}\xi_i + \frac{l_2}{m}\sum_{j=1}^{m}\delta_j$$

$$s.t. \quad \forall 1 \leq i \leq n, \forall \widehat{y} \in \mathbf{S}:$$

$$w^\top \phi(x_i, y_i) \geq w^\top \phi(x_i, \widehat{y}) + \Delta(y_i, \widehat{y}) - \xi_i \tag{3.1}$$

$$\forall y_j \in \mathbf{S}, 1 \leq j \leq m:$$

$$\sum_{i=1}^{n} L_i \mathbf{1}_{\{arg\max_y w^\top \phi(x_i, y) = y_j\}} \leq C_j + \delta_j \tag{3.2}$$

In the above formulation, constraint set (1) is the classical soft constraint used in multi-class SVM [102]. It requires the discriminant score of the correct class label for instance i, $w^\top \phi(x_i, y_i)$, to be at least as large as the score for any other

class $\widehat{y}$, $w^\top \phi(x_i, \widehat{y})$, by a non-negative margin specified as $\Delta(y_i, \widehat{y})$ which is also the loss function for incorrectly assigning an instance to class $\widehat{y}$ while the true class is y_i. The definition of Δ is usually problem-specific and we will describe our choice of the loss function in the experiment section. $\phi(x, y)$ is the joint feature map vector defining the characteristic relations between instance x and class y. ξ_i is positive only when the constraint is not strictly satisfied. Constraint set (2) is our newly introduced resource capacity constraint. Each class y_j $(1 \leq j \leq m)$ has a predefined capacity C_j, and each instance x_i $(1 \leq i \leq n)$ is associated with a load L_i. When an instance is classified to have a class label y_j according to the linear discriminant score, i.e., $arg\max_y w^\top \phi(x_i, y) = y_j$, we say that the class y_j must provide resources to x_i equal to its load. The sum of loads from instances assigned to a class cannot exceed the capacity of that class. This constraint is made soft because in the applications we have in mind the load incidence can adjust somewhat to locational resource capacity, yet eventual hard bounds on resource capacity still remain. Therefore, instead of enforcing a hard constraint, we add the penalty term δ in the objective function to discourage any capacity constraint violation. l_1 and l_2 are two constants controlling the tradeoff in the objective function among model complexity represented by $\frac{1}{2}\|w\|^2$, the training set misclassification upper bound $\frac{1}{n}\sum_{i=1}^{n} \xi_i$ and the average resource capacity constraint violation $\frac{1}{m}\sum_{j=1}^{m} \delta_j$.

3.2 Optimization

SVM$_{cap}$ is a non-convex optimization problem due to constraint (2). We can still optimize the formulation by an extension of Pegasos [94]. Pegasos is a primal gradient descent method for optimizing SVM. Our method SVM$_{cap}$ builds on top of it with additional gradient descent step when a violation of the resource capacity

constraint occurs. The optimization for SVM$_{cap}$ is presented in Algorithm 2.

Algorithm 2: SVM$_{cap}$ Training Algorithm

1: Input: $\mathbf{x}$, C, ϕ, Δ, T, λ
2: w_0=$\mathbf{0}$
3: **for** t = 1, 2, ..., T **do**
4: A=$\emptyset$
5: **for** i = 1, 2, ..., $|\mathbf{x}|$ **do**
6: $\widehat{y}$= $arg\max_{y \neq y_i}(w_{t-1}^T \phi(x_i, y) + \Delta(y_i, y))$
7: **if** $w_{t-1}^T \phi(x_i, y_i) < w_{t-1}^T \phi(x_i, \widehat{y}) + \Delta(y_i, \widehat{y})$ **then**
8: A=$A \cup (x_i, y_i, \widehat{y})$
9: **end if**
10: **end for**
11: w_t=(1-$\frac{1}{t}$)w_{t-1}+$\frac{1}{\lambda t|\mathbf{x}|} \sum_{(x_i, y_i, \widehat{y}) \in A} (\phi(x_i, y_i) - \phi(x_i, \widehat{y}))$
12: **for** j = 1, 2, ..., $|\mathbf{y}|$ **do**
13: **if** $\sum_{i=1}^{|x|} L_i \mathbf{1}_{\{arg\max_y w_t^\top \phi(x_i, y) = y_j\}} > C_j$ **then**
14: UpdateCapacityViolation(j,t);
15: **end if**
16: **end for**
17: **if** $\|w_t\| > \frac{1}{\sqrt{\lambda}}$ **then**
18: w_t=$\frac{1}{\sqrt{\lambda}} \frac{\mathbf{w}_t}{\|\mathbf{w}_t\|}$
19: **end if**
20: **end for**
21: Output: $\mathbf{w}_T$

The algorithm iteratively optimizes w. The optimization process sequentially checks for any violation with respect to constraint (1) and (2) on lines 7 and 13 respectively. All instances that violate constraint (1) are recorded in the set A on line 8, and a gradient descent step used in Pegasos is carried out on line

11 with all such instances. For each class whose total assigned load exceeds its capacity, *UpdateCapacityViolation* is the process used to update the weight w. Ideally we want the process to project the current w back to the feasible regions with respect to the capacity constraints. However, it can be easily shown that the problem of finding a feasible solution for the capacity constraints with fixed load and capacity vectors is NP-complete, as we can reduce the 1-d bin packing problem to it. Therefore we use a greedy-based heuristic to adjust w if such a capacity violation happens, as illustrated in Algorithm 3.

Algorithm 3: UpdateCapacityViolation

1: Input: j, t

2: $w_{t+\frac{1}{2}}$=w_t

3: **for** i = 1, 2, ..., |**x**| **do**

4: **if** $arg\max_y w_{t+\frac{1}{2}}^\top \phi(x_i, y) = y_j$ **then**

5: $\widehat{y} = arg\max_{y \in S'} w_t^\top \phi(x_i, y)$ where S' is the set $\{y_p | \sum_{k=1}^{|x|} L_k \mathbf{1}_{\{arg\max_y w_{t+\frac{1}{2}}^\top \phi(x_k, y) = y_p\}} + L_i \leq C_p\}$

6: w_t=w_t+$\alpha * (\phi(x_i, \widehat{y})\mathbf{1}_{S' \neq \emptyset} - \phi(x_i, y_j))$

7: **end if**

8: **end for**

In Algorithm 3, for each instance x_i that is assigned to class y_j whose capacity constraint has been violated, we find the highest scored alternative class $\widehat{y}$ in terms of $w^\top \phi(x_i, \widehat{y})$ on line 5 that has enough capacity under the current instance assignment. A class has enough capacity with respect to x_i if the capacity constraint is still satisfied after x_i is moved to the class. Then we update w according to the original and newly found class for x_i on line 6. If S'=$\emptyset$ on line 5, we would only update w with respect to the current class y_j for instance x_i. We keep a copy of w_t in $w_{t+\frac{1}{2}}$ because we want to fix the instance assignment in each call to this

function in which we dynamically update w_t. α is a tuning parameter we set at $\frac{10}{\lambda t|\mathbf{x}|}$ from cross-validation results. We also used non-linear RBF and polynomial kernels in our optimization but it did not improve the empirical performance significantly. Taking the size of the feature map vector as a constant, the complexity of the overall training procedure is $O(Tnm^2)$ where T is the number of iterations we fixed at 500 in all our empirical studies.

3.3 Classification with Multiple Instances

After the training phase, w has been optimized in Algorithm 2. When presented with a test set of instances, there is no value in implementing individual and independent classification for each instance, because by doing so we cannot enforce the capacity constraints, as the capacity constraints depend on the predicted class labels of all instances. In the testing phase we aim for high accuracy and low capacity constraint violations. In order to do this, we propose a heuristic similar to the First Fit Decreasing (FFD) heuristic used for the Bin Packing problem [44]. The method is detailed in Algorithm 4.

We first classify the instances independently using the discriminant score given by w on line 3. This may result in multiple violations of class capacity constraints. For each such class whose capacity constraint has been identified as violated on line 6, B is the set of instances that can be moved to some other class without introducing new capacity violations. The while loop from line 8 to line 13 tries to move the instances with large load to other classes, maintaining a high discriminant score between the moving instance and its destination class. The final predictions for the test instances are stored in the vector $\mathbf{r}$. The complexity of the testing

Algorithm 4: SVM$_{cap}$ Testing Procedure

Input: $\mathbf{w}$, testing set $\mathbf{x}$
for i = 1, 2, ..., |$\mathbf{x}$| **do**
  r_i=$arg\max_y w_t^\top \phi(x_i, y)$
end for
for j = 1, 2, ..., |$\mathbf{y}$| **do**
  if $\sum_{k=1}^{|x|} L_k \mathbf{1}_{r_k=y_j} > C_j$ **then**
    B=$\{x_i | r_i = y_j \wedge \exists l \sum_{k=1}^{|x|} L_k \mathbf{1}_{r_k=y_l} + L_i \leq C_l \}$
    while $B \neq \emptyset$ and $\sum_{k=1}^{|x|} L_k \mathbf{1}_{r_k=y_j} > C_j$ **do**
      k=$arg\max_{x_k \in B} L_k$
      S''=$\{y_l | \sum_{i=1}^{|x|} L_i \mathbf{1}_{r_i=y_l} + L_k \leq C_l\}$
      r_k=$arg\max_{y \in S''} w_t^\top \phi(x_k, y)$
      B=B-$\{x_k\}$
    end while
  end if
end for
Output: $\mathbf{r}$

procedure is $O(nm^2)$ where n and m are the number of instances and classes in the test set.

3.4 An Optimal Post Learning Optimization Approach

In the previous section, we used a simple heuristic in the testing procedure to minimize the capacity violation. Although it is fast and experimentally efficient, we can further improve the performance using an optimal formulation for the testing phase. In this section we model the post-learning prediction problem as an

integer programming and solve it optimally. The output of the training process in 2 is the w parameter, from which we can naturally define a utility function of assigning instance x_i to class y_j, p_{ij}, to be $w^T\phi(x_i, y_j)$. Our goal for the integer programming is to maximize the sum of utilities of assigning instances to classes while enforcing the resource capacity constraint.

To formally define the optimization model, let :

N be the number of test instances,

M be the number of classes, and

L_i be the load of instance $1 \leqslant i \leqslant N$;

C_j be the capacity of class $1 \leqslant j \leqslant M$;

p_{ij} be the benefit for assigning instance i to class j

The integer programming for the optimal assignment is:

$$\max \sum_{i,j} p_{ij} x_{ij} \tag{3.3}$$

$$\sum_{i \in \{1,\ldots,k\}} L_i x_{ij} \leq C_j, \qquad \forall 1 \leqslant j \leqslant M \tag{3.4}$$

$$\sum_{j \in \{1,\ldots,m\}} x_{ij} = 1, \qquad \forall 1 \leqslant i \leqslant N \tag{3.5}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall i \in \{1, ..., N\}, 1 \leqslant j \leqslant M \tag{3.6}$$

(3.6) states that the decision variables x_{ij} are binary, representing assigning instance i to class j if it is 1, or 0 otherwise. Constraint (3.5) states that each instance needs to be assigned to exactly one class. Constraint (3.4) states that the total load of instances assigned to class j should not exceed its capacity level C_j. Our objective in (3.3) is to maximize the total utility from a feasible assignment.

The problem described above is very similar to the well known NP-complete

problem, namely the Generalized Assignment Problem (GAP) [44]. In GAP, we are given a set of items with weights and profits and a set of knapsacks with capacities. The goal is to decide which items to take and in which knapsack to put them into while not exceeding the capacities. The difference from our problem is that each item/instance needs to be assigned to exactly one knapsack (class) which makes our problem similar to Bin Packing. We can show that GAP reduces to our problem as follows: given an instance of GAP with profits $p_{ij} : i \in \{1..N\}, j \in \{1..M\}$, weights $L_i : i \in \{1..N\}$, capacities $C_j : i \in \{1..M\}$, we construct an instance of our problem with $M + 1$ classes where the first M classes are the same and the extra class has capacity $C_{M+1} = \sum_{i \in \{1..N\}} L_i$ and the corresponding item profits are $p_{i,M+1} = 0$. It is easy to see that the set of feasible solutions is the same between the two problems and the objective value of each solution is also the same in both the original and the constructed problem. Hence if we have a method to solve our problem to optimality, we can also solve GAP optimally via a polynomial time transformation of solution.

As we mentioned a natural way to define the utility of assigning instance i to class j is $w^T \phi(x_i, y_j)$. We have defined in total of three different methods to compute the profit p_{ij} in the optimization model in Table 3.1.

For each instance x_i, let y_i be the correct class label.

The *Max* definition of utility is the normalized version of *Score*. *Oracle* is for comparison purposes only since it can not be obtained during testing. It is interesting to notice that even with perfect knowledge of the class labels, the accuracy of the best assignment of instances to classes might not have accuracy of 100% because the optimization enforces a hard constraint on capacity violations. The result of applying our optimization will be presented in the experiment section.

Table 3.1: The Choice of Utility p_{ij}

Name	Definition
Score	$w^T\phi(x_i, y_j)$
Max	$\frac{w^T\phi(x_i,y_j)}{\max_{y_k\in\{1..m\}} w^T\phi(x_i,y_k)}$
Oracle	1 if y_i is the true class for x_i, 0 otherwise

3.5 Evaluation Metrics

We evaluate the model performance in terms of both accuracy and the level of capacity constraint violation. Let $\overline{S}$ be the set of classes in which the capacity constraint is violated, and $\mathbf{r}$ be the prediction vector for the instances. We define the following two measures for capacity constraint violation:

1. Aggressive Violation: $\dfrac{\sum_{i=1}^{|\mathbf{r}|} L_i \mathbf{1}_{r_i\in\overline{S}}}{\sum_i L_i}$
2. Conservative Violation: $\dfrac{\sum_{i=1}^{|\mathbf{r}|} L_i \mathbf{1}_{r_i\in\overline{S}} - \sum_{j\in\overline{S}} C_j}{\sum_i L_i}$

The Aggressive Violation measure (AV) calculates the percentage of instance load that has been assigned to a violated class. This differs from the Conservative Violation measure (CV), which only includes the excess amount of load in each violated class for the calculation. Both of the two measures are useful in different contexts. Perhaps the different emphasis of the two measures is best illustrated with an example. Suppose that there are only two classes A and B, with capacities 2 and 4 respectively. Let the total population load be 6. Consider the following two scenarios: the load assignment to A and B is (4,2), and the load assignment is (1,5).

Under the first scenario, AV = 4/6 and CV = 2/6; and in the second scenario AV = 5/6, while CV = 1/6. It is easily seen that the metrics move in opposite directions in going from the first scenario to the second. This is driven by the fact that under scenario 1 we see a high capacity violation at a lower-capacity class, while under the other scenario we see a small capacity violation at a high-capacity class. Hence the CV metric might be preferred by a policymaker concerned about overtaxing the resources of classes (which could lead to permanent damage of a locations resources), while AV might be preferred by policymakers more concerned with issues generated by high overall concentration of instances, which in the context of allocations of humans and/or animals could translate into higher short-term conflict, disease transmission, etc.

3.6 Related Work

To the best of our knowledge, the problem of learning with resource capacity constraints is a new problem and hence no directly relevant approach exists prior to SVM_{cap}.

The problem of learning with resource capacity constraints is related to the class imbalance problem, in which a classical classification task is presented with datasets where some classes contain many more instances than other classes. Such a problem setting may result in overfitting of a method if prediction accuracy is used as the optimization metric. As summarized in [26], some classical approaches and recent trends in addressing this problem include re-sampling to balance the dataset [8] and cost-sensitive learning [39] that in general assigns a higher penalty on misclassifications with the classes that have only a few instances. Our problem

bears the resemblance that each class has a capacity, where more instances are likely to be associated with classes with high resource capacities, leading to imbalanced classes. However, our problem differs from the class imbalance problem in the way that each instance has a natural load which is to be satisfied by the assigned class, and the imbalances in classes are caused by the resource allocation procedure, rather than the particularities of the dataset.

Our testing procedure is also related to the inverse optimization problem, in which an optimization problem $\mathbf{P}$, usually with a linear cost function $c \times x$, is given with a feasible solution x_0. The goal of inverse optimization is to perturb the cost coefficients c as little as possible in terms of the L_1 norm or the L_∞ norm so that x_0 is an optimal solution of $\mathbf{P}$ with respect to the perturbed cost coefficients [2]. In our problem of learning with capacity constraints, we have a similar concept that the predicted classification results are not necessarily optimal with respect to the capacity constraints so a post-learning optimization stage is applied thereafter. Our approach differs in the fact that we do not modify the cost coefficients or the constraints to artificially force the predictions to be optimal, but rather we undertake additional heuristic steps to further improve the performance on capacity constraint violations.

The optimization process we have in this chapter is an extension of the Pegasos framework [94]. Pegasos is a general purpose gradient descent method to solve the SVM convex quadratic optimization formulation. It differs from many previous methods in that Pegasos operates solely in the primal solution space with guaranteed convergence and performance bound, which gives it the advantage to be easily adapted to various modified SVM problem formulations. SVM_{cap} extends the original Pegasos optimization process by including non-convex resource capac-

ity constraints, and iteratively optimizing with respect to these constraints in the additional gradient descent step in Algorithm 3. The drawback of our extension is that, due to the non-convexity of the additional constraints, the proven optimization accuracy and convergence bounds in Pegasos do not hold anymore. However, from our experimental studies we observed that SVM_{cap} almost always converges within 500 iterations with competitive accuracy performance benchmarked against SVM optimized with Pegasos.

While the research at hand represents a novel contribution to economic models of discrete choice, by modeling decisions over a relatively large space of options, a significant toolkit exists for a range of analogous problems. The primary approach is through a probabilistic representation of the latent discrete choice problem, using models called discrete (or qualitative) response models, including the widely-used probit and logit models of binary response, and multinomial models for choice spaces with more than two elements. Much of the early work is summarized in McFadden's work [70], who received a 2000 Nobel Prize for his efforts in this area. While parameter selection in such models is carried out according to a statistical criterion, the approach of essentially deriving estimated choice weights on a vector of observables in order to classify the elements of the choice space is quite closely related. Empirical models also exist that put more structure on the mechanics of discrete choice, backing out parameters in a setup with an agent maximizing an explicit objective function subject to constraints; a representative review is given in [1]. Such models, however, typically suffer from a "curse of dimensionality" when applied to problems with a large choice and/or state space. Finally, the approach of using a penalty term in the objective of an optimization problem in order to allow for "soft" constraints is related to the penalization of over-fitting in semi-parametric penalized spline models in statistics [91].

Table 3.2: Dataset Summary

Dataset	n	m	k	$\Delta(y_i, y_j)$
Pastoral	979	311	202	if Cap(y_i)≤Cap(y_j),
				dis(y_i, y_j)
Forest Fire	517	10	24	o.w 2dis(y_i, y_j)

3.7 Experiments

3.7.1 Dataset

We will evaluate the performance of SVM$_{cap}$ and the other benchmark methods on two datasets: the pastoral dataset and the forest fire dataset. More details follow in the next section. A summary of the two datasets is presented in Table 3.2. n, m and k are the number of instances, number of classes and number of features in each dataset respectively. The loss function $\Delta(y_i, y_j)$ in the SVM$_{cap}$ formulation penalizes assigning an instance to class y_j when y_i is the true class. In both of the datasets, this function depends on the relative capacity of the two classes: if the capacity of y_j is not smaller than that of y_i, the penalty is the distance between the two water points or fire fighter stations; if the capacity of y_j is smaller, the mistake of classifying an instance to class y_j is more likely to violate the capacity constraint for y_j and therefore we set the penalty to be twice the distance between y_i and y_j in our experiments.

The African Pastoral Herding Dataset

This application emerges from a larger research program on livestock-based agro-pastoralism in regions of the world such as the arid and semi-arid lands of east Africa [71, 67]. During the dry seasons those households engaged in sustainable pastoralism form "satellite camps", herding the majority of the animals to rangelands waterpoints, dozens if not hundreds of kilometers away, in search of water and, most importantly, forage resources. At the community level there is an additional challenge: if all herders congregate on a single or small number of waterpoints the chances of resource exhaustion, conflict and disease may increase. Hence there is a constraint on the aggregate allocation of herds to waterpoints, though under more severe drought conditions we generally see a greater concentration of herders on the most reliable waterpoints, and hence the constraint is not best modeled as strictly binding; presumably, herders adjust to droughts by reducing herds' water intake. The nature of the choice problem generates an interesting classification problem as given a herder household specifics, we want to predict which water point it would choose to go to subject to the resource capacity constraints.

The data used in this chapter was collected by the USAID Global Livestock Collaborative Research Support Program (GL CRSP) "Improving Pastoral Risk Management on East African Rangelands" (PARIMA) project through quarterly household surveys from June 2000 to June 2002[1]. Data on a broad array of household demographic characteristics, herd management choices, and consumption, labor, health and educational decision-making was collected at five locations in rural Northern Kenya. 979 instances (household-water point observations) are gathered through surveys from 9 quarters. Through GPS data collected during the original

[1] aem.cornell.edu/special_programs/AFSNRM/Parima/index.htm

survey, and supplemental data collection during 2007-08, we have accurate estimates of the geographical locations of all waterpoints, which allows us to generate variables such as distances between water points. Household herds are aggregated into a scalar index in Tropical Livestock Units (TLU), as is common on the broader literature, at the rate of 0.7 TLU per camels, 1 per cattle, and 0.1 per sheep or goat. The household herd size (in TLU) serves as the resource load variable. Water point capacities are estimated according to observed herd usage over the chosen training periods. We use the primary observed waterpoint chosen in each period as the true class. There are a total of 202 binary features for each household-water point pair, which describe the distance between the household and water point, and the relative magnitude of household load and water point capacity.

The Forest Fire Dataset

In addition to the pastoral migration dataset, we also constructed a synthetic one based on a forest fire dataset from the UCI machine learning repository [6]. This dataset contains 517 instances of forest fires in the northeast region of Portugal. The original task was to predict the burned area of each forest fire from 12 numerical features. Adapting this to our problem setting, we view the area of a forest fire to be its load. Then we generate a certain number of fire fighter stations on the map whose boundaries are defined by the forest fire instances. Our problem is to allocate the forest fires as responsibilities to different fire fighter stations, while the total load of fires for a station should not exceed the capacity of that station. The locations of the stations are determined by a k-means clustering over the fires with k=10. The capacity of each station is the sum of loads of the forest fires assigned to it. We also randomly perturb 20% of the assignment of fires to the second closest station if the sum of existing load to the second closest station is smaller.

Thus we prevent a simple algorithm that considers no capacity constraint but only shortest distance assignment to happen to have a perfect score. To construct the joint feature map between a fire and a station, we included distance features and capacity features. There are a total of 24 binary features constructed for each fire and station pair such as "the distance between the fire and the station is within 1 unit" (distance is calculated in the unit of longitude and latitude degrees), or "the fire load is no greater than 10% of the capacity of the station" etc.

3.7.2 Evaluation

Benchmark Methods

All experiments are coded in C++ and run on machines with 3.8GHz Intel CPU and 8GB memory. Since we are not aware of any prior method that has been developed to address our learning problem, we choose to use the SVM classifier and a bin-packing heuristic as the benchmark methods for SVM$_{cap}$. The SVM classifier ignores constraint set (2) in the SVM$_{cap}$ optimization. We use the standard Pegasos process in [94] to train the SVM classifier and predictions are made as in normal classification problems ignoring the capacity constraints. The bin-packing heuristic uses the simple First Fit Decreasing (FFD) approach when assigning instances to classes. The FFD heuristic is not a learning-based method and for the NP-complete bin-packing problem it has an approximation bound of using no more than $\frac{11}{9}$OPT+1 bins for an instance whose optimal solution uses OPT bins [44]. Adapting this to our problem setting, the heuristic first sorts the instances in decreasing order of their loads. For each instance after sorting, FFD sequentially searches for the first class that has a remaining capacity larger than the load of the

instance and assigns the instance to that class. If no such class exists, the heuristic simply assigns the instance to the class with the largest remaining capacity.

Accuracy and Resource Capacity Violation

We compared several methods on our formulated problem of learning with resource capacity constraints, including the bin-packing heuristic, SVM and SVM$_{cap}$. The result of using 9-fold cross validation on the Pastoral Dataset is presented in Table 3.3. We used 9 folds because the dataset contains survey results from 9 quarters. In each fold we use the data of 8 quarters as training set and test on the data from the remaining quarter. The accuracy, AV and CV measures are averaged among the 9 folds and the numbers are reported with the λ parameter chosen to give the best performance. σ measures the standard deviation in each category. All numbers are in percentages, except for t, the average runtime in seconds for 1-fold including both training and testing. We observe that the ground truth has a significant violation in both AV and CV measures of resource capacity constraints. When we compare the performance of the Bin-packing heuristic, which aims solely to minimize the capacity violation, with SVM$_{cap}$, we see that SVM$_{cap}$ has a much higher accuracy measure while the violation level is kept at about 9% for AV and 5% for CV. SVM aims to improve the accuracy in its predictions without considering any capacity constraints. Both SVM and SVM_{cap} provide the highest accuracy of 20.8% and 21.2%, the difference of which is well within the average standard deviation of more than 6%, while SVM$_{cap}$ has reduced the resource capacity violations of SVM by 90% in AV and 91% in CV. The runtime of SVM$_{cap}$ and that of SVM is comparable in general.

The results on the forest fire dataset are shown in Table 3.4. Due to the way we

Table 3.3: 9-fold Cross Validation Results on the Pastoral Dataset (λ=0.1)

Method	Acc.	σ	AV	σ	CV	σ	t
Ground Truth	100	0	86.6	4.2	44.0	4.8	-
Bin-packing	4.9	1.5	<1	<1	<1	<1	<5
SVM	20.8	6.5	96.6	1.9	58.5	4.2	398
SVM$_{cap}$	**21.2**	6.9	**9.2**	10.3	**5.2**	6.3	512

Table 3.4: 10-fold Cross Validation Results on the Forest Fire Dataset (λ=0.01)

Method	Acc.	σ	AV	σ	CV	σ	t
Bin-packing	25.7	9.1	18.6	24.3	<1	<1	<1
SVM	**76.2**	4.1	74.7	16.0	12.3	6.1	1.6
SVM$_{cap}$	64.0	13.8	**35.7**	18.6	**1.4**	1.9	2.7

generate this synthetic dataset, the ground truth arrangement has an accuracy of 1 and no capacity violations, therefore we did not include it in the table. Among the three methods, we see that Bin-packing heuristic provides the worst accuracy but also least capacity constraint violations as we expected. SVM$_{cap}$'s accuracy performance is much higher than Bin-packing but 16% lower than that of SVM. SVM$_{cap}$ on the other hand reduces the capacity violation of SVM by as much as 52% in AV and 87% in CV. We conclude from our empirical studies that SVM$_{cap}$ is a competitive method whose predictions are accurate, and can maintain the resource capacity constraints extremely well compared with SVM.

Effectiveness of the Optimal Post Learning Optimization

In section 3.4 we have discussed the optimal post learning optimization approach to further improve the performance. The integer programming formulation is optimally solved using CPLEX. In this section we compare SVM with the optimization formulation in section 3.4 applied to the utility definitions of Table 3.1. The results on both datasets are presented in Table 3.5 and Table 3.6. For the Pastoral Dataset, the optimization noticeably reduces both the AV and CV violation measures to 0 for all folds of cross-validation data. Although the accuracy is 12.5% lower for *Score* and 17.3% lower for *Max* compared with SVM, the improvement in violation reduction by the post learning optimization is impressive. For the Forest Fire Dataset, not only the resource capacity violation is reduced to 0 using our optimization formulation, but also the accuracy is improved from 76.2% (SVM) to 80.8% (*Score*) and 81.6% (*Max*).

We also included the comparisons using an additional Wine dataset from [83]. The load of each wine is defined to be 1 and the capacity of each class in the test set is proportional to the capacity in the train set, to reflect the fact that the quality of wine is relative and only a percentage of wine, rather than a fixed number of bottles, can be described as premium. The results are shown in Table 3.7. We see that the post-learning optimization is good at maintaining a high accuracy level and completely resolve all capacity violations.

Though the post learning optimization exhibits significant improvement empirically compared with SVM, it comes with a price: the capacity constraints in equation (3.4) are forced to be hard constraints. Therefore when the instance loads are large compared with the resource capacities, there may be no feasible solution. One future research direction is to replace them with soft constraints through, for

Table 3.5: 9-fold Cross Validation Results on the Pastoral Dataset with Optimization (λ=0.1)

Method	Acc.	σ	AV	σ	CV	σ
SVM	20.8	6.5	96.6	1.9	58.5	4.2
Score	18.2	5.0	0	0	0	0
Max	17.2	7.4	0	0	0	0
Oracle	63.3	7.1	0	0	0	0

Table 3.6: 10-fold Cross Validation Results on the Forest Fire Dataset with Optimization (λ=0.01)

Method	Acc.	σ	AV	σ	CV	σ
SVM	76.2	4.1	74.7	16.0	12.3	6.1
Score	80.8	10.4	0	0	0	0
Max	81.6	10.1	0	0	0	0
Oracle	100	0	0	0	0	0

example, Lagrangian relaxation.

Policymaker Assistance

A key motivation of our model is to assist policymakers in making informed decisions when they want to invest to increase the number and capacity of resource points to better satisfy the resource demands of pastoralist households, or to put out a fire. The decision to invest in additional resources can be extremely beneficial where resource capacity constraints are severely violated. However, it is difficult to

Table 3.7: 10-fold Cross Validation Results on the Wine Dataset with Optimization (λ=100)

Method	Acc.	AV	σ	CV	σ
SVM	44.9	100	0	55.1	0.6
Score	37.4	0	0	0	0
Max	37.4	0	0	0	0
Oracle	92.2	0	0	0	0

determine where and to what capacity a new resource point should be constructed, especially given a limited budget. Policymakers therefore often face the problem of setting up a small number of new resource points to satisfy the needs of individuals, without knowing which ones the individuals would choose to use. SVM$_{cap}$ can help in the above decision-making process. Suppose in the pastoral scenario, we use SVM$_{cap}$ to predict where the households would go to fetch water among a set of new waterpoints that could be constructed. If a water point is predicted to attract a lot of households, it probably needs to be constructed with priority.

To evaluate the performance of SVM$_{cap}$ in improving the policy decision-making process, we have conducted the following experiment: for each fold of the test data, we view the water points as additional new resource points to be constructed and rank the water points in terms of the total load amount of households that have been predicted to go to that water point. We next calculate the total real household load on a water point from ground truth observations. There are a total of 311 water points. Let L_i , $i \in \{1, \ldots, 311\}$, be the real water point total load for the i^{th}-highest ranked water point in terms of its predicted total load, we define the amount of *unmet resource demand* after constructing k, $k \in \{1, \ldots, 311\}$,

most highly ranked water points to be $\frac{\sum_{i=k+1}^{311} L_i}{\sum_{i=1}^{311} L_i}$. This quantity explains how much more resource demand would still exist after we construct the first k water points according to the predicted ranking. Then we can plot the amount of unmet resource demand as we increase the number of constructed new water points according to our predicted ranking. The results in percentage are presented in Figure 3.1 after averaging over the 9 test folds for both SVM and SVM$_{cap}$. We clearly see that SVM$_{cap}$ is a better prediction method than SVM to guide policymakers, because it ranks the water points in a way that closely resembles the real water point total load from households. In particular, SVM$_{cap}$ ranks the water point with the largest load in reality number 2 in its predictions, while SVM ranks it more than 120 positions lower.

The Effect of Capacity Constraint Tightness

Intuitively, if the resource capacity is significantly large compared with the load of the instances, the capacity constraint is not binding in an optimal solution and our formulation in section 3.1 is equivalent to a multi-class SVM classifier. On the other hand, if the instance load is much higher than most resource capacity, there is little room for optimization to lower any violation measure. Therefore we expect both SVM and SVM$_{cap}$ would perform similarly on the two extreme scenarios. In this section we investigate the methods' performance when the capacity constraint is at various levels with respect to the instance load.

We artificially varied the total capacity of the Pastoral and Forest Fire dataset to be from 10% to 125% of the total load of all instances. The total capacity is then proportionally distributed over individual resource points according to their initial capacities. We compare SVM and SVM$_{cap}$ for each level of capacity constraint

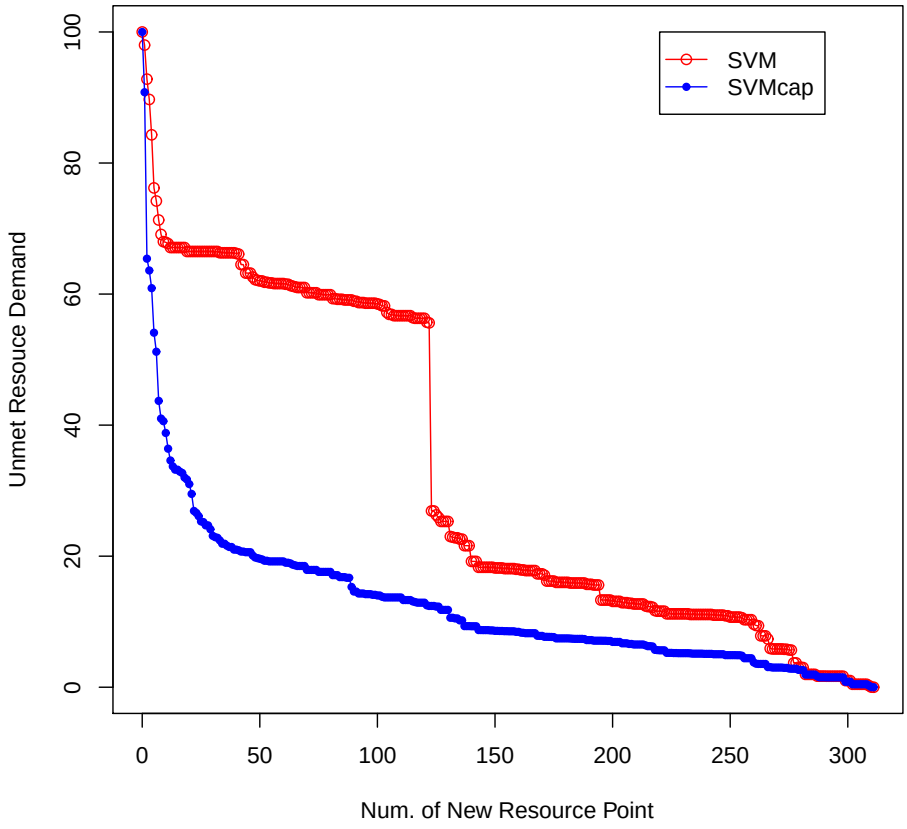

Figure 3.1: Guidance for Policymakers

tightness and the results are presented in Figure 3.2. The left panel shows the accuracy, AV and CV measures between SVM and SVM$_{cap}$ on the Pastoral dataset and the right panel shows those on the forest fire dataset. All results are averaged using cross validation. Comparing the accuracy, we see a trend that SVM$_{cap}$ has a high or comparable accuracy with SVM when the capacity constraint is tight. This follows our intuition that when there is little optimization can be done to improve the capacity violation, SVM$_{cap}$ and SVM would have similar performance. This is exactly the case if we look at the AV and CV measure figures. Once the constraint becomes easier to satisfy as the total capacity increases, SVM$_{cap}$ starts to effectively

Table 3.8: 10-fold Cross Validation Results on the Wine and Forest Fire Dataset with Optimization

Data	SVM acc.	AV	CV	Score acc.	Max acc.	Oracle acc.
Wine 100%	44.9	100	55.0	35.9	38.8	92.2
Wine 125%	44.9	100	43.8	36.8	43.4	98.5
Fire 100%	76.2	60.3	8.1	77.1	77.3	100
Fire 125%	76.2	22.8	1.8	76.2	77.2	100

learn to lower the capacity violation at the cost of reasonable accuracy decrement. When the total capacity further increases it requires relatively less effort of SVM_{cap} to obtain results with small capacity violation, therefore the accuracy performance returns to close to the level of SVM. This "U" shaped accuracy plot and the continuously decreasing violation plots are all in accordance with our expectation.

We also compared SVM with our post-learning optimization approach on instances with feasible Integer Programming solutions, i.e, with a total capacity no less than 100% the total load. The results on the Forest Fire and Wine datasets are presented in Table 3.8. All optimization options give 0 capacity violations. It is clear that the Max optimization option keeps a comparable or even higher accuracy level compared with SVM while reduce the violations to 0. This shows our post-learning optimization is very efficient for learning problems with capacity constraints.

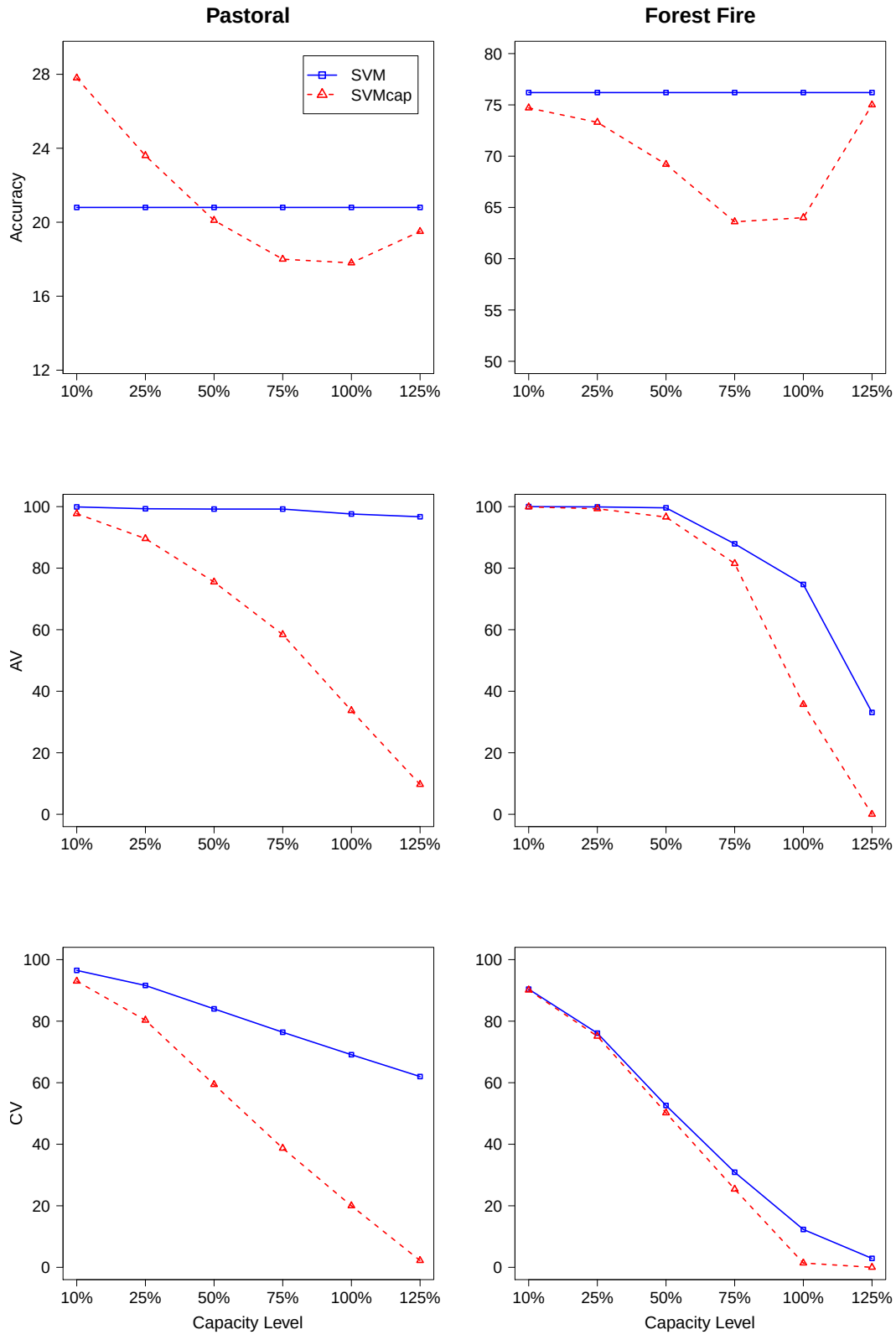

Figure 3.2: The Effect of Varying the Capacity Constraint Tightness

3.8 Discussion

In this chapter we developed a learning approach to the classification problem with locational resource capacity constraints, motivated by a real pastoral herding dataset. With our problem definition each class has a capacity and each instance has a load. The additional constraint that needs to be satisfied, compared with the classical classification problem, is that the sum of load of instances predicted to belong to a class should not exceed the capacity of that class. We formulated the problem as an SVM-based non-convex optimization problem and solved it using a gradient descent method, SVM$_{cap}$. We used two new numeric measures for the new problem, namely the Aggressive Violation (AV) and Conservative Violation (CV), to gauge the level of resource capacity constraint violation in a test dataset. Experimental results comparing SVM$_{cap}$ and other benchmark methods, including classification SVM, on both the real pastoral dataset and a synthetic dataset show that our method is able to maintain a high level of prediction accuracy, while reducing the level of resource capacity violation by as much as 90% measured in terms of AV and CV. When combined with the optimal post-learning optimization, we are able to further reduce the capacity violation and maintain a high level of accuracy relative to SVM. In addition, SVM$_{cap}$ also provides policymakers with reliable evidence for resource point improvement, by predicting relatively accurately which resource point would attract the most individual demand for resources once constructed. When investigating the effect of capacity constraint tightness, we observed the effectiveness of SVM$_{cap}$ in capacity violation reduction under different situations.

In this chapter multiple entities' preferences and choices are considered together to satisfy the resource capacity constraints. These entities have similar and consis-

tent preferences with one another. For example, in the Pastoral dataset, intuitively the herders all want to travel safely to the closest and abundant waterpoint to feed the household animals; in the Forest Fire dataset the fires are assigned to stations according to the distance and fire fighter capabilities. On the other hand, in certain cases multiple entities with conflicting preferences are involved in the same decision making process. Learning the implicit preferences in such a context require a different approach. In the next chapter we will present our structured ranking approach for the patent prior art search problem, in which both the patent examiner and applicant decide what prior patents to cite for a new application. As we will see, their reasons for including a citation often differ with one another and our learning method is capable of providing a list of possible citations for a new patent application more objectively based on importance and relevance.

CHAPTER 4

RANKING STRUCTURED DOCUMENTS WITH LEARNED PREFERENCES

Patents protect intellectual property rights by granting the invention's owner (or his/her assignee) the right to exclude others from making, using, selling, or importing the patented invention for a certain number of years, usually twenty years, in exchange for the public disclosure of the details of the invention. Like any property right, patents may be transacted (e.g., sold, licensed, mortgaged, assigned or transferred) and therefore they have economic value. For a patent to be granted, the patent application must meet the legal requirements related to patentability. In particular, in order for an invention to be patentable, it must be *novel*. Patent applications are subject to official examination performed by a patent examiner to judge its patentability, and in particular its novelty. In most patent systems *prior art* constitutes information that has been made available to the public in any form before the filing date that might be relevant to a patent's claims of originality. Previously patented inventions are to a great extent the most important form of prior art, given that they have already been through the patent application scrutiny.

The decision by an applicant to cite a prior patent is arguably "tricky" [63]: on one hand, citing a patent can increase the chance of a patent being accepted by the United States Patent and Trademark Office (USPTO), if it shows a "spillover" from the previously granted patent; on the other hand, citing a prior patent may invalidate the new patent if it shows evidence of intellectual property infringement. As a result, some applicants choose to cite only remotely related patents, even if they are aware of other more similar patents. It is thus part of the responsibil-

ity of the USPTO examiner to add patent citations, beyond those provided by the applicant. This is important since more than 30% of all patent applications granted from 2001 to 2007 actually have zero applicant citations. However, this is also a tedious procedure often resulting in human error, considering the huge amount of patents to be possibly cited and the fact that different examiners with different expertise backgrounds are likely to cite different sets of patents for the same application. Furthermore, while applicants tend to cite patents that facilitate their invention, denoting to some extent *spillovers* from a patent invention, examiners tend to cite patents that block or limit the claims of later inventions, better reflecting the patent *scope* and therefore representing a better measure of the private value of a patent.

We focus on the problem of automatically ranking patent citations to previously granted patents: given a new patent application, we are interested in identifying an ordered list of previously granted patents that examiners and applicants would consider useful to include as prior art patent citations.

In recent years several margin based methods for document ranking have been proposed, e.g., [51, 82, 19, 57, 23, 109]. For example, in the approach by [51, 82] the ranking problem is transformed into a binary classification problem and solved by SVM. [19] considers different loss functions for misclassification of the input instances, applied to the binary classification problem; [57] uses click through data from search engines to construct pair-wise training examples. In addition, many other methods learn to optimize for a particular performance measure such as accuracy [78], ROC area [52, 21], NDCG [17, 18, 23] and MAP [109].

Note that the patent prior art search problem differs from the traditional learning to rank task in at least two important ways. Firstly, in learning to rank we

measure the relevance of query-document pairs, where a query is usually a short sentence-like structure. In patent ranking, we are interested in patent-patent relevance, where a patent is a full document much longer in length than a normal query. We will also see in the experiment section that treating a document as a long query often results in inferior performance. Secondly, in the case of patent prior art search, we have two entities, namely the examiner and applicant, who decide what patents to cite for a new application. As we mentioned above, the differences in their citation behaviors are not only a matter of level of relevance, but also of strategy.

To address the various issues concerning patent prior art search, we propose a large-margin based method, SVM Patent Ranking (SVM_{PR}), with constraint set directly capturing the different importance between citations made by examiners and citations made by applicants. Our approach combines patent-based knowledge features with meta-score features, based on several different ad-hoc Information Retrieval methods. Our experiments on a real USPTO dataset show that SVM_{PR} performs on average 30%-40% better than other state-of-the-art ad-hoc Information Retrieval methods on a wireless technology patent dataset, in terms of the NDCG measure. To the best of our knowledge, this is the first time a machine learning approach is used for automatically generating patent prior art citations, which is traditionally handled by human experts.

4.1 Numerical Properties of Patent Citations

From January 1, 2001 to the end of 2007, the USPTO has granted 1,263,232 patents. The number of granted patents (per year) is displayed in the left plot of Figure 4.1. We observe a steady and slow increase in the number of patents

granted per year from 2001 (∼180,300 patents) to 2003 (∼187,132 patents), and then a steep decrease until the all-time low of ∼157,866 in 2005. Interestingly there is a sharp increase in the number of patents in 2006, achieving the highest number to date with ∼196,593 patents.

There are obvious differences between examiner and applicant citations. (See e.g., [5, 92].) The average number of patent citations added by the examiner and applicant is presented in the middle plot of Figure 4.1. The average number of patent citations added by the examiner is relatively stable, ranging from 5.49 to 6.79 , while the average number of applicant citations increases significantly from 8.69 in 2001 to 15.11 in 2007. In addition, the distribution of patent citations by the applicant is extremely uneven: ∼372,372 (29.5%) patents have no applicant citations, and ∼19,388 (1.5%) patents have more than 100 applicant citations, while the number of patents with more than 100 examiner citations is only 40. The rightmost plot of Figure 4.1 compares the number of patents with no more than 20 citations made by the examiner, versus that made by the applicant. As clearly displayed, a large portion of the patents have 0 applicant citation, and the mode number of examiner citation is 3, with ∼134,323 patents.

4.2 SVM$_{PR}$ for Patent Ranking

4.2.1 Some Notations

We denote the set of patents whose citations are of interest by μ; and the set of patents that patents in μ cite, by ν.

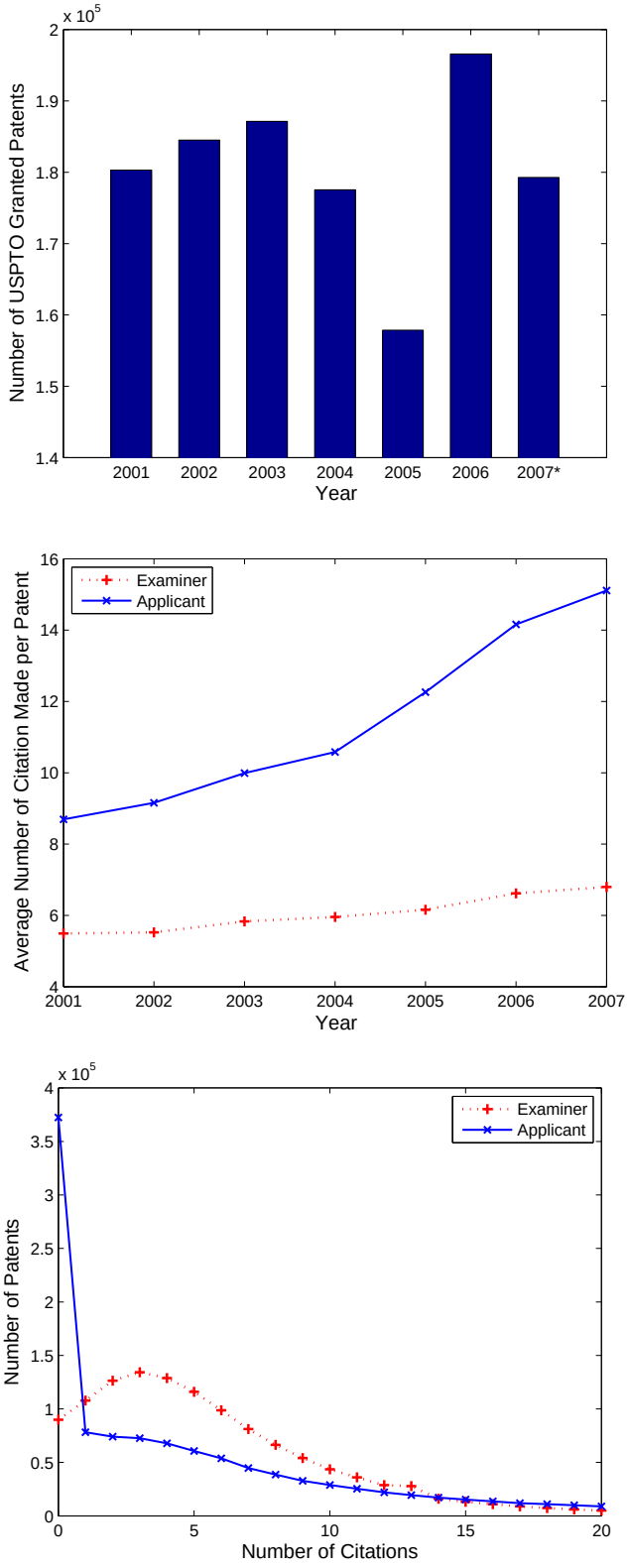

Figure 4.1: USPTO Patent Data. Top panel: patents per year; Middle panel: yearly citations by examiner and applicant; Bottom panel: frequency of examiner citations and applicant citations.

In addition, $\forall p_i \in \mu$ and $\forall p_j \in \nu$, we define the citation matrix $\mathbf{C}$ with entries C_{ij} to be

$$C_{ij} = \begin{cases} 2 & \text{if patent } p_i \text{ cites patent } p_j \text{ by examiner} \\ 1 & \text{if patent } p_i \text{ cites patent } p_j \text{ by applicant} \\ 0 & \text{otherwise} \end{cases}$$

The numerical values seem to be arbitrary here but the reason for such choices will be clear in the next section. Moreover, we denote $\Phi(p_i, p_j)$ as the feature map vector constructed from patents p_i and p_j. Details of Φ are presented in section 4.2.3.

4.2.2 SVM$_{PR}$ Formulation

Our model is called SVM Patent Ranking (SVM$_{PR}$). It is formulated as the following large margin quadratic optimization problem with the constraint set directly capturing the specificities of patent ranking:

Optimization Problem I

$$\min_{\mathbf{w},\xi \geq 0} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{|\mu||\nu|^2} \sum_{p_i \in \mu} \sum_{p_j \in \nu} \sum_{p_k \in \nu} \xi_{ijk} \tag{4.1}$$

subject to:

$\forall p_i \in \mu, \forall p_j \in \nu, \forall p_k \in \nu,$

$$\mathbf{w}^T \Phi(p_i, p_j) - \mathbf{w}^T \Phi(p_i, p_k) \geq \Delta(p_i, p_j, p_k) - \xi_{ijk}$$

where

$$\Delta(p_i, p_j, p_k) = \begin{cases} C_{ij} - C_{ik}, & \text{if } C_{ij} - C_{ik} > 0 \\ -\infty, & \text{otherwise} \end{cases}$$

Here $\Delta(p_i, p_j, p_k)$ is the loss function when $C_{ij} - C_{ik} > 0$ and the learned $\mathbf{w}$ parameter scores less with $\Phi(p_i, p_j)$ than $\Phi(p_i, p_k)$. The magnitude of $C_{ij} - C_{ik}$ represents the severity of the loss: a mistake of ranking an uncited document higher than a document cited by an examiner is more serious than ranking the same uncited document higher than some applicant cited document. If $C_{ij} - C_{ik} \leq 0$, no loss will be incurred, and we set $\Delta(p_i, p_j, p_k)$ to $-\infty$ to invalidate the constraint.

The objective function (4.1) is the usual minimization that trades off between $\|\mathbf{w}\|^2$, the complexity of the model, and $\sum \xi_{ijk}$, the upper bound of the total training loss defined by Δ. λ is the tradeoff constant.

The above formulation considers all tuples (i,j,k) to ensure that examiner cited patents are ranked higher than applicant cited patents, which are again ranked higher than uncited patents, by an absolute margin "1" using the linear discriminant score $\mathbf{w}^T\Phi$. The drawback is that it contains $\mathcal{O}(|\mu||\nu|^2)$ constraints, which poses a challenge for any reasonable training process. In fact, we do not need to include all tuples as constraints, but rather only consider the extreme values of the ranking scores of the three different citation groups (by examiner, by applicant and not cited). Therefore, we can construct the following alternative formulation:

Optimization Problem II (SVM_{PR})

$$\min_{\mathbf{w},\xi\geq 0} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{2|\mu|}\sum_{p_i\in\mu}(\xi_i^1 + \xi_i^2) \tag{4.2}$$

subject to: $\forall p_i \in \mu$

$$\begin{aligned} \min_{p_j\in\{p:p\in\nu\wedge C_{ip}=2\}} \mathbf{w}^T\Phi(p_i, p_j) \quad & - \\ \max_{p_k\in\{p:p\in\nu\wedge C_{ip}=1\}} \mathbf{w}^T\Phi(p_i, p_k) \quad & \geq \quad 1 - \xi_i^1 \end{aligned}$$

$$\begin{aligned} \min_{p_j \in \{p : p \in \nu \wedge C_{ip}=1\}} \mathbf{w}^T \Phi(p_i, p_j) \quad & - \\ \max_{p_k \in \{p : p \in \nu \wedge C_{ip}=0\}} \mathbf{w}^T \Phi(p_i, p_k) \quad & \geq \quad 1 - \xi_i^2 \end{aligned}$$

It is not difficult to see that the constraint set of Optimization Problem 1 implies the constraint set of Optimization Problem 2, and vice versa, since in the latter case we made changes so that the constraints only need to be satisfied at the extreme values. The advantage is obvious because this leads to a convex quadratic optimization problem with only $\mathcal{O}(|\mu|)$ constraints. The objective functions in the two formulations are not equivalent though, as the new formulation only penalizes constraint violations made by the extreme cases. Our training algorithm will be based on Optimization Problem 2.

We note that if the distinction between examiner citations and applicant citations is ignored, i.e, C_{ij}=1 iff patent p_i cites p_j, Optimization Problem I can be transformed into the ordinal regression formulation in [51] in a straightforward way. However, the ordinal regression formulation would require $\mathcal{O}(|\mu|^2|\nu|^2)$ constraints, treating each $\Phi(p_i, p_j)$ as an individual example; or $\mathcal{O}(|\mu||\nu|^2)$ constraints if the examples $\Phi(p_i, p_j)$ are first grouped by p_i. In order to differentiate examiner and applicant citations we applied the SVM multi-rank ordinal regression in [27] to our transformed problem with $\mathcal{O}(|\mu||\nu|)$ examples. Unfortunately the ordinal regression method could not handle the scale of the problem, as it fails to complete the training process within 48 hours of runtime. In contrast, our approach, SVM_{PR}, exploits the important distinction between examiner and applicant citations, leading to the efficient formulation and training process.

In addition, our formulation is also related to the label ranking problem, in which each instance is associated with a preference graph over the possible class labels. A preference graph essentially defines a partial order over the labels and

the goal of label ranking is to learn a ranking function f such that given labels y and y' for an instance x, $f(x, y) > f(x, y')$ whenever $y \succ y'$ defined by the preference graph [32]. In our patent ranking problem setting, for each patent pair we have three "labels": *cited by examiner*, *cited by applicant* and *not cited*. We can transform our formulation to a label ranking problem as follows: for each patent pair, we define the preference graph over the three labels to be a directed complete binary tree of two levels, where the root node is the label with the correct relation between the two patents, and the two leaf nodes are the two other labels. It is straightforward to check that once this transformed label ranking instance is solved our original patent ranking solution can be constructed by checking the relative ranking values of the three labels for each patent pair. However, this transformation requires $\mathcal{O}(|\mu||\nu|)$ examples and preference graphs, therefore it is inefficient for optimization.

After the optimization phase, we use the linear discriminant function $\mathbf{w}^T\Phi(p, q)$ to rank a new patent p against any patent q to be considered for the prior art citation. Our target ranking for p is that all examiner-cited patents are ranked higher than all applicant-cited patents, and all cited patents are ranked higher than all patents not cited. We evaluate our learned patent ranking with the target ranking by NDCG.

It is worthwhile to note that our approach is also related to the margin based methods for label ranking [93], in which the goal is to learn an ordered preference list of labels for each instance by learning local weight vectors for each class label. In contrast, our approach is to learn a global weight vector and differentiate class labels (cited by examiner, or applicant, or not cited) by the linear discriminant score, using the patent specific feature map vectors.

4.2.3 Feature Map Construction

One major difference between assessing patent similarity and query-document similarity, a traditional task in IR, is that patents are full documents, which are usually significantly longer than an average query. Of course we can treat one patent as a long query, but this will not result in good performance for our task as we will see in the experiment section.

A key step in our approach for the training and testing procedure is the construction of a feature map for pairs of patents p_i and p_j. Intuitively, $\Phi(p_i, p_j)$ represents information we know about how similar p_i and p_j are. In our approach, $\Phi(p_i, p_j)$ is a column vector composed of two parts: the domain knowledge patent pair features and the meta score features.

Domain Knowledge Patent Pair Features

The domain knowledge features between patents p_i and p_j come from the study of patent structure without any citation information of either p_i or p_j. Here applicants and inventors are used interchangeably. We have 12 binary features in $\Phi(p_i, p_j)$ as follows:

1. Both patents are in the same US patent class; *2.* Both patents have at least one common inventor (defined by same surname and first name); *3.* Both patents are from the same assignee company (e.g., Microsoft, HP etc.); *4.* Both patents are proposed by independent applicants (i.e, no assignee company); *5-7.* First inventors of the two patents come from the same city, state or country; *8.* p_i's patent date is later than p_j's patent date (so p_i can possibly cite p_j); *9-12.* Both patents made 0 claims, 1-5 claims, 6-10 claims or more than 10 claims.

Meta Score Features

We also make use of ad-hoc IR methods to score the patent pairs. The scores are then changed into feature vector representation in Φ.

We used the Lemur information retrieval package[1] to obtain the scores on patent pairs by six ad-hoc IR methods: TF-IDF, Okapai, Cosine Similarity and three variations of the KL-Divergence ranking method. Additional descriptions of these methods can be found in section 4.3.3. Each method is essentially a scoring function $\psi(q, d) \in Q \times D \mapsto \mathbb{R}$ where Q is a set of queries and D is a set of documents whose relevance to the queries is ranked by the real-valued scores. We use the patents' titles and abstracts available from the USPTO patent database to obtain the meta score features.

We can view each sentence of a patent $p_i \in \mu$ as a query whose relevance with respect to a patent $p_j \in \nu$ is to be evaluated. Let S be the set of all sentences from all patents in μ. We associate a binary vector t^s of length 50 with each $s \in S$ according to $\psi(s, p_j)$, following the representation in [109]:

$$\forall i \in \{1, \ldots, 50\},\ t_i^s = \begin{cases} 1, & \psi(s, p_j) \geq Pt(2(i-1)) \\ 0, & otherwise \end{cases}$$

where $Pt(x)$ is the x^{th} percentile of $\{\psi(q, p_j) : q \in S\}$. The t^s vector is a binary representation of how relevant sentence s is with respect to p_j, compared to all other sentences in S. We repeat this procedure for all six ad-hoc IR methods and concatenate the results to obtain the vector t^s of length 300.

Let $(s_1, s_2, \ldots, s_{m_i})$ be the m_i sentences of p_i sorted in non-increasing order

[1]Lemur Toolkit v4.5, copyright (c) 2008 University of Massachusetts and Carnegie Mellon University

according to $\psi_{TF-IDF}(s, p_j)$. The meta score feature vector between p_i and p_j is defined as

$$\Psi(p_i, p_j) = \sum_{l=1}^{m_i} \frac{t^{s_l}}{2^{(l-1)}} \tag{4.3}$$

In other words, Equation (4.3) is a weighted sum of t^s from each sentence s, discounting sentences that are ranked as less relevant exponentially by ψ_{TF-IDF}. We also tried other alternative weighting schemes, such as harmonic discounting, but none of them performed empirically as well as the exponential weight discount.

Hence, the feature map $\Phi(p_i, p_j)$ for any $p_i \in \mu$ and $p_j \in \nu$ is the concatenation of the 12 knowledge domain features and $\Psi(p_i, p_j)$, with a total of 312 features.

4.2.4 The Training Algorithm

The training algorithm of SVM_{PR} that optimizes (4.2) with respect to $\mathbf{w}$ is an extension of Pegasos [94]. Pegasos is an SVM training algorithm that alternates between a gradient descent step and a projection step. It operates solely in the primal space, and has proven error bounds. Our training algorithm is presented in Algorithm 5.

Among the four input parameters, μ and ν have the same meaning as in the previous sections; T is the total number of iterations; λ is the constant in the SVM_{PR} formulation. In our experiments T and λ are fixed at 200 and 0.1. In general the performance is not sensitive to any reasonable λ setting. Lines 6-9 calculate the extreme values of the discriminant function $\mathbf{w}^T\Phi(p_i, p_j)$ for patents p_j grouped by their citation relation with p_i in order $\mathcal{O}(|\nu|)$. The set A is the set of violated constraints with respect to Optimization Problem II, or equivalently

Algorithm 5: SVM$_{PR}$ Training Algorithm

Input: μ, ν, T, λ

w_0=**0**

for t = 1, 2, ..., T **do**

  A=$\emptyset$

  for i = 1, 2, ..., $|\mu|$ **do**

    p^e_{min}=arg$min_{p_j \in \nu \wedge C_{ij}=2} w^T_{t-1}\Phi(p_i, p_j)$

    p^a_{max}=arg$max_{p_j \in \nu \wedge C_{ij}=1} w^T_{t-1}\Phi(p_i, p_j)$

    p^a_{min}=arg$min_{p_j \in \nu \wedge C_{ij}=1} w^T_{t-1}\Phi(p_i, p_j)$

    p^n_{max}=arg$max_{p_j \in \nu \wedge C_{ij}=0} w^T_{t-1}\Phi(p_i, p_j)$

    if $w^T_{t-1}(\Phi(p_i, p^e_{min}) - \Phi(p_i, p^a_{max})) < 1$ **then**

      A=$A \cup (p^e_{min}, p^a_{max})$

    end if

    if $w^T_{t-1}(\Phi(p_i, p^a_{min}) - \Phi(p_i, p^n_{max})) < 1$ **then**

      A=$A \cup (p^a_{min}, p^n_{max})$

    end if

  end for

  w_t=(1-$\frac{1}{t}$)w_{t-1}+$\frac{1}{\lambda t|\mu|} \sum_{(p_j,p_k)\in A}(\Phi(p_i, p_j) - \Phi(p_i, p_k))$

  if $\|w_t\| > \frac{1}{\sqrt{\lambda}}$ **then**

    w_t=$\frac{1}{\sqrt{\lambda}} \frac{\mathbf{w}_t}{\|\mathbf{w}_t\|}$

  end if

end for

Output: $\mathbf{w}_t$ with the minimum validation error.

the most violated constraints of Optimization Problem I. Line 17 updates $\mathbf{w}$ using the violated constraints in A. This step is in order $\mathcal{O}(|\mu|)$ as $|A| \leq 2|\mu|$. Lines 18-20 ensure the 2-norm of $\mathbf{w}$ is not too big, which is a condition used in [94] to prove the existence of an optimal solution. Line 22 outputs the final $\mathbf{w}$ parameter with the best validation set performance. The performance measure is described in section 4.3.2. In summary, the runtime for each iteration is $\mathcal{O}(|\mu||\nu|)$, so the runtime for SVM$_{PR}$ training is $\mathcal{O}(T|\mu||\nu|)$, given precalculated mapping Φ.

Theoretically, we can show that the number of iterations needed for SVM$_{PR}$ to converge to a solution of accuracy ϵ from an optimal solution is $\tilde{\mathcal{O}}(\frac{R^2}{\lambda\epsilon})$ where R=max$_{p_i\in\mu,p_j\in\nu}$ $2\|\Phi(p_i,p_j)\|$. This result follows from Corollary 1 of [94]. In practice, our training algorithm always completes within five hours of runtime in the experiments.

4.3 Empirical Results

4.3.1 Dataset

For our experiments we focused on Wireless patents granted by the USPTO. We started with data from 2001 since this is the first year USPTO released data differentiating examiner and applicant citations. We used a list of Essential Wireless Patents (EWP), a set of patents that are considered essential for the wireless telecommunication standard specifications being developed in 3GPP – Third Generation Partnership Project – and declared in the associated ETSI (European Telecommunications Standards Institute) database. We considered three versions of the dataset: the original patent files, the patent files after applying a Porter

stemmer [87], and the patent files after applying a Porter stemmer and common stopwords removal. The Porter stemmer reduces different forms of the same word to its original "root form", and the stopword removal eliminates the influence of common but non-informative words, such as "a" and "maybe", in the ranking algorithms.

In our experiment, μ is the set of essential wireless patents (2001-2007) that made at least one citation to any other patent in 2001-2007, and ν is the set of patents from 2001-2007 that has been cited by any patent in μ. This dataset currently contains $\sim$197,000 patent-pair citation judgments. This is significantly larger in scale than the OHSUMED dataset widely used as a benchmark dataset for information retrieval tasks, which contains 16,140 query-document pairs with relevance judgement [53]. Our goal is to learn a good discriminant function $\mathbf{w}^T\Phi(p_i, p_j)$ for $p_i \in \mu$ and $p_j \in \nu$. We randomly split the patents in μ into 70% training, 10% validation and 20% test set in 10 independent trials to assess the performance of SVM_{PR} and other benchmark methods.

4.3.2 Performance Measure

Given a patent in μ, we rank all patents in ν by the score of $\mathbf{w}^T\Phi$, and evaluate the performance using the Normalized Discounted Cumulative Gain (NDCG) [56].

NDCG is a widely used performance measure for multilevel relevance ranking problems. NDCG incorporates not only the position but also the relevance level of a document in a ranked list. In our problem, there are three levels of relevance between any two patents p_i and p_j, as defined by C_{ij}, with 2 the most relevant and 0 the least. In other words, if p_j is cited by an examiner in p_i, it has a relevance

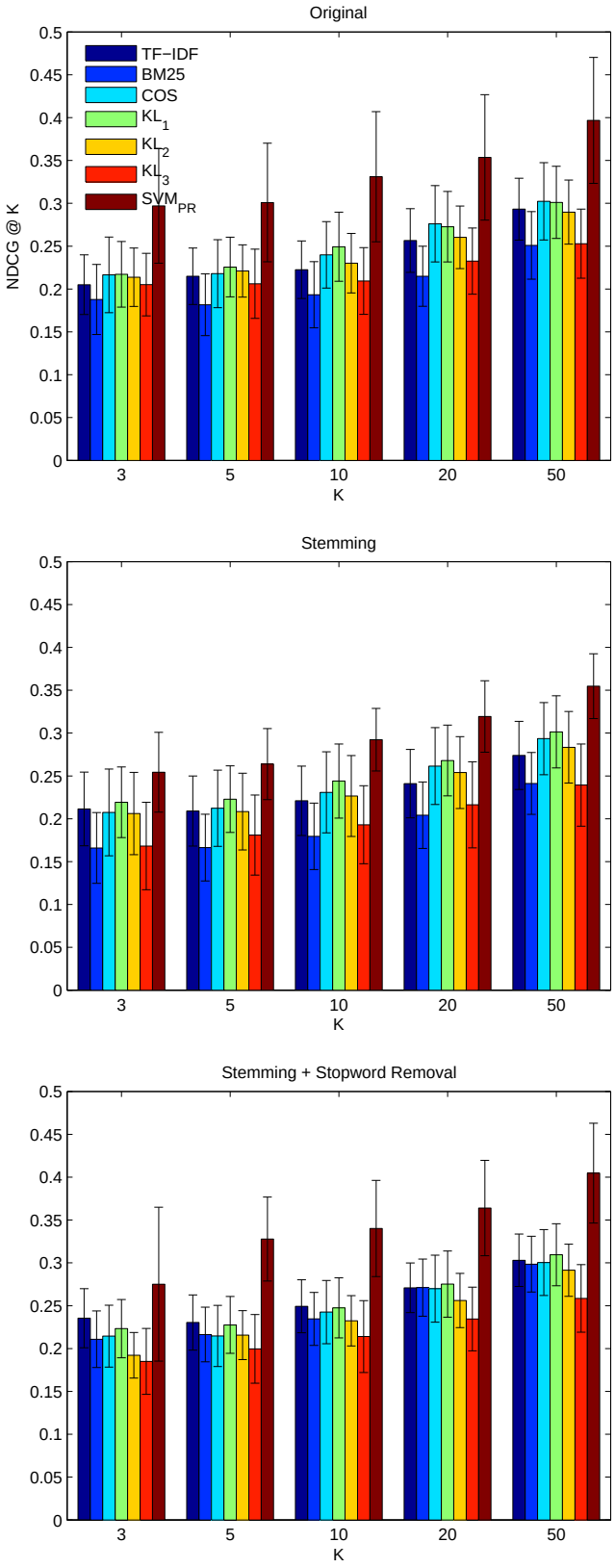

Figure 4.2: NDCG Scores of SVM_{PR} and Benchmark Methods

Table 4.1: Ad-hoc IR Methods as Benchmark

Method	$\psi(q, d)$
TFIDF	Term freq(q,d)*Inv. doc freq(q)
Okapi	The BM25 method in [90]
Cossim	Cosine similarity in vector space
KL_1	KL-Divergence with a Dirichlet prior
KL_2	KL-Divergence with a Jelinek-Mercer prior
KL_3	KL-Divergence with an absolute discount prior

value of 2, and so on. Given an essential wireless patent p_i, and a list of patents π from ν, sorted by the relevance scoring function, the NDCG score for p_i at position k ($k \leq |\nu|$) is:

$$NDCG_{p_i}@k = N_{p_i} \sum_{j=1}^{k} \frac{2^{C_{i\pi_j}} - 1}{log(j+1)} \tag{4.4}$$

N_{p_i} is the normalization factor so that the perfect ranking function, where patents with higher relevance values are always ranked higher, will have a NDCG of 1. The final NDCG performance score is averaged over all essential patents in the test set.

4.3.3 Benchmark Methods

In this section, we briefly describe the six ad-hoc IR methods implemented by the Lemur IR package (with default parameters) that we used as benchmarks, and how they are used to rank the patent citations. Given a query q and a document d, each of the six methods can be described as a function $\psi(q, d)$ whose value,

a real number, is often regarded as the measure of relevance. The six methods are presented in Table 4.1. Details of the last three KL-Divergence methods with different smoothing priors can be found in [111].

For each of the six ranking methods, given a wireless essential patent $p_i \in \mu$, we score it with all patents in ν, by treating p_i as the query and ν as the pool of documents. The methods are evaluated using NDCG with the ranked patent lists. Since we used the patent date feature in SVM_{PR} which effectively indicates that certain citations are impossible, to be fair for the benchmark methods, we set all returned scores $\psi(p_i, p_j)$ from the benchmark methods to -∞, if patent p_i is an earlier patent than p_j.

4.3.4 NDCG Results

We evaluate the NDCG at positions 3, 5, 10, 20, and 50. The NDCG score is averaged using 10 independent trials. For SVM_{PR}, the maximum number of iterations is 200, and the test performance is evaluated when the best validation performance is achieved within the iteration limit. For the benchmark methods, the performance is reported on the same test sets as SVM_{PR}. The results are presented in Figure 4.2. First of all, SVM_{PR} outperforms the benchmark methods by a significant margin for all five positions. Referring to Table 4.2 for the numerical comparison with the best performance of any benchmark method, SVM_{PR} outperforms the best result of the benchmark methods by 16% to 42%. Among all benchmark methods, the KL-Divergence with Dirichlet prior scored the highest, with more than 60% of all tests. Comparing the different document pre-processing procedures, we found that applying the Porter stemmer alone actually hurts the performance of SVM_{PR} by a significant 10% to 17% in comparison to using the original patent documents, while

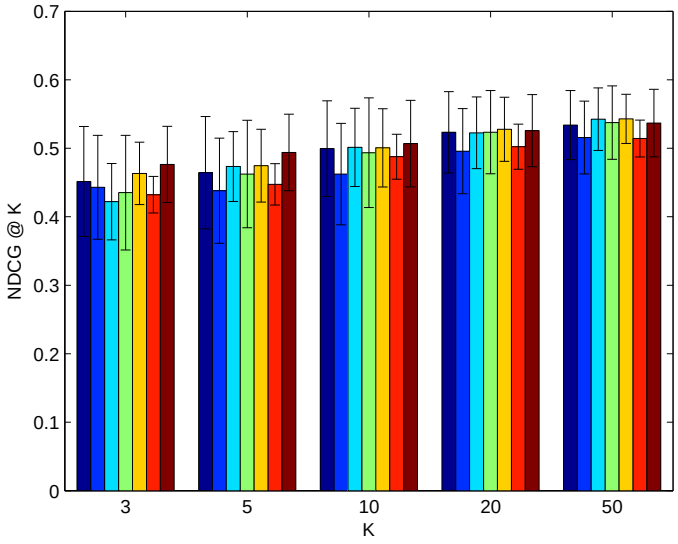

Figure 4.3: NDCG Scores on A Random Dataset

the influence on the benchmark methods is marginal. The overall best performance is achieved with SVM_{PR} when applying the stemming and stopword removal, as highlighted in Table 4.2. All the performance differences between SVM_{PR} and the best alternative benchmark are significant by a signed rank test at the 0.01 level.

4.3.5 Results on A Random Dataset

We repeated the experiments on a non-homogeneous random dataset to understand better whether SVM_{PR} learns and benefits from the internal structure of the patent set μ, and justify our decision to group homogeneous patents together during training.

We randomly sampled the set μ, and among the patents cited by patents in μ, we randomly selected the set ν, while keeping the same number of patent-pair citation judgments as before. We then repeated the experiments described

Table 4.2: SVM_{PR} and Benchmark Performance Comparison

	Original			Stemming			Stemming and Stopword		
K	ψ_{best}	SVM_{PR}	Impr.(%)	ψ_{best}	SVM_{PR}	Impr.(%)	ψ_{best}	SVM_{PR}	Impr.(%)
3	0.217	**0.297**	36.7	0.219	0.254	16.0	0.235	0.275	16.9
5	0.226	0.301	33.4	0.223	0.264	18.3	0.230	**0.328**	42.3
10	0.249	0.331	32.8	0.244	0.292	19.8	0.249	**0.340**	36.4
20	0.276	0.354	28.1	0.268	0.319	19.1	0.275	**0.364**	32.3
50	0.302	0.397	31.2	0.301	0.355	17.7	0.310	**0.405**	30.8

above with the Porter stemmer and stopword removal applied. Now instead of a structured essential patent set, μ is an arbitrary set with little internal similarities among its members. Because the patents in μ are quite unrelated, the patents they cite in ν are non-homogeneous too. In other words, this is an easier task than the previous one since we are learning to rank patents in ν that are more distinguishable than before.

The results are presented in Figure 4.3. We observed that the new performance differences among SVM_{PR} and other benchmark methods are largely indistinguishable (best alternative method performance is within 5% of SVM_{PR}). This follows from our intuition that the random dataset lacks a homogeneous citation structure to be learned, and the reasonable methods would perform comparably well, although the learned ranking is less informative as it only differentiates irrelevant patents.

4.4 Discussion

This chapter focused on the problem of patent prior art search which is traditionally a tedious task requiring significant expert involvement because the preferences of the patent applicant and patent examiner, although implicit, differ substantially. Our proposed approach based on large margin optimization incorporates constraints that directly capture patent ranking specificities and ranks patent citations to previously granted patents by a linear discriminant function $\mathbf{w}^T\Phi$, where $\mathbf{w}$ is the learned parameter and Φ is the feature map vector consisting of patent domain knowledge features and meta score features. Experiments on a wireless technology patent set show that SVM_{PR} consistently outperforms other state-of-the-art general IR methods, based on the NDCG performance measure.

In this and the previous two chapters we discussed the problems of learning with implicit preferences without modeling their exact forms. If one wishes to automatically formulate such preferences in a more concrete manner, a model to "explain" the learning models is needed. To achieve this goal, we propose an Inductive Logic Programming (ILP) based method to extract the prediction decisions of a learning model in terms of Horn clauses in the next chapter.

CHAPTER 5

LEARNING TO EXPLAIN THE IMPLICIT PREFERENCES FROM OPAQUE MODELS VIA INDUCTIVE LOGIC PROGRAMMING

As the field of machine learning prevails, tens of new models are being proposed each year with increasing complexity and improved performance. However, before we can automate some critical decision processes previously determined by human experts using a machine learning model, the experts need to understand what the model is doing even if the model has shown good performance in objective tests such as cross validation. For example, in order to diagnose cancer for patients, the doctor can never completely rely on any machine learning model; rather, he may prefer to be provided with an "explanation" of a high accuracy machine learning model on the diagnosis task to aid his decision makings. Unfortunately, most of the machine learning models being proposed are not "transparent", i.e, they are very hard for a human expert to understand by looking at the trained model: Artificial Neural Net (ANN) consists of a layered network configuration and activation threshold values, and a normal soft-margin Support Vector Machine (SVM) consists of kernel mappings and quadratic optimization formulations with weight and slack-variable values [29], let alone other meta-level models such as AdaBoost[42] and Bagging[14]. Among the more traditional machine learning models, decision tree is a nice example of an "understandable" model, as a human expert can easily interpret a decision tree's classification behavior by looking at the split nodes. People have worked at extracting a tree structured model from trained ANN [30] or SVM [37] with varying degrees of success. Methods that try to learn a simpler model to mimic the behavior of a complex meta-level model have also been proposed [16, 35]. We will discuss more of these models in the related work section.

However, we feel that a framework which is applicable to general machine learning models and capable of explaining the behavior of a trained model for human understandability with objective evaluations is more than needed.

Fortunately, understandability is the strength of rule-based learning applications, such as in [106, 55]. Inductive Logic Programming (ILP) [80], as one rule-based learning model, has shown its efficiency in inducing new hypothesis in various applications, especially in the field of molecular biology [98, 100]. In this chapter, we will develop a framework that explains the behavior of a "black box" machine learning model using Horn clauses [54] induced by an ILP system for interpretability.

More excitingly, we devised a method to overcome ILP's dependence on background knowledge to produce accurate predictions on datasets where pre-defined background knowledge is hard to obtain. We will also show that using our method the ILP predicting accuracy would converge to that of highly accurate opaque machine learning models such as ANN and SVM.

5.1 Inductive Logic Programming Preliminaries

Inductive Logic Programming (ILP) is the study that combines inductive machine learning and logic programming. The general problem an ILP system tries to solve is briefly described as follows:

Given a set of examples E consisting of positive example set E^+ and negative example set E^-, and background knowledge B, such that

1. $\forall\ e \in E^-,\ \neg(B \vdash e)$

2. $\exists\ e \in E^+,\ B \vdash e$

The goal is to learn a hypothesis H, such that

1. $\forall\ e \in E^-,\ \neg(B \wedge H \vdash e)$,
2. $\forall\ e \in E^+,\ B \wedge H \vdash e$.

In other words, ILP's task is to find a new hypothesis H, together with the background knowledge B, entails all the positive examples but not any negative example. In the process of inducing new rules to include in the current hypothesis, 4 of Duce's inductive inference rules presented below are often considered, where lower case letters represent single propositional variable and capital letters represent conjunctions of such variables.

$$\textbf{Absorption}\frac{p \leftarrow A, B \quad q \leftarrow A}{p \leftarrow q, B \quad q \leftarrow A}$$

$$\textbf{Identification}\frac{p \leftarrow A, B \quad q \leftarrow A, q}{q \leftarrow B \quad p \leftarrow A, q}$$

$$\textbf{Intra-construction}\frac{p \leftarrow A, B \qquad\qquad p \leftarrow A, C}{q \leftarrow B \quad p \leftarrow A, q \quad q \leftarrow C}$$

$$\textbf{Inter-construction}\frac{p \leftarrow A, B \qquad\qquad q \leftarrow A, C}{p \leftarrow r, B \quad r \leftarrow A \quad q \leftarrow r, C}$$

In practical ILP systems such as Progol [79], new clauses in the hypothesis are generated from the search in the lattice defined by the θ-subsumption relationship on the clauses, where the bottom of the lattice is the most specific clause generated from a ground example. [10] provides more details of ILP theories and techniques.

5.2 Synthetic Dataset Illustration

In order to demonstrate the main idea and feasibility of using ILP to extract a set of Horn clauses that closely describes the behavior of an opaque machine learning model for human interpretation, we will use a toy example of learning the concept of "2-disks". In this example, we are trying to learn whether it is stable to put one disk on the other, given the sizes of the 2 disks. It is stable only when the size of the first (lower) disk is no smaller than the second. Suppose each disk can take size 1, 2 or 3. We generated 50 random instances of this task, and used SVM to learn the concept. SVM gives 94% accuracy in 10-fold cross validation. Then we apply ExOpaque to learn from the instances together with SVM's predictions. And the 6 learned Horn rules are as follows, which are the underlining concept of stability for 2-disk:

```
1. stable :- disk1(3)
2. unstable :- disk1(1), disk2(3)
3. unstable :- disk1(1), disk2(2)
4. stable :- disk1(2), disk2(2)
5. unstable :- disk1(2), disk(3)
6. stable :- disk2(1)
```

This toy example shows the idea of how we may use ExOpaque to explain an opaque model (SVM) given its behaviors (predictions) on original input instances. We rely on the fact that the opaque model is relatively accurate, otherwise what we can learn by ExOpaque is still reasonably close to the model but can be far away from the true concept.

5.3 ExOpaque Algorithm

ExOpaque is presented in Algorithm 6. The different ways we construct training examples in step * for an ILP system in ExOpaque will be described and validated in the experiment section. The ILP system we used in ExOpaque is Progol version 4.4 [79]. Readers who are interested in more of ILP's fundamental theories and techniques can refer to [10]. We use Progol to only find Horn clauses where the body elements are conjunctions of attribute values, and the head element is the class label. The Horn clause format is similar as in the synthetic example, which makes the learned set of rules easy to understand. When using ExOpaque returned rules to classify examples, the example's attribute values are matched sequentially from the first rule, and the first matching Horn clause's head part specifies the class label of the example. We also supply Progol all attribute values of constructed training examples as mentioned above with their class labels.

The algorithm starts with the constructed supervised example set E, and applies ILP model to learn a set of Horn clauses to describe the set E. However, the set of Horn clauses is not guaranteed to classify all examples in E, since we do not include any default rule such as *class(A, 1) :-* , which in general does not offer more understanding on the behavior of the opaque model. Instead, we iteratively update E to be the examples left unclassified in the while loop. And k is the ratio of newly correctly classified examples to the increase of the size of the rule set. We stop the process and return the set of Horn clauses sequentially added in each iteration when either there is no more example to apply ILP to (E=$\emptyset$) or the gain of classifying more examples correctly is not worth the increase in rule size compared to the previous iteration ($c \times k < k_0$), where c is a constant representing our willingness to trade-off rule size with fidelity in general.

Algorithm 6: ExOpaque Framework

Input:fully trained opaque model M, M's training set (optional), test set (optional)

Output: a set of Horn clauses to explain the behavior of M for interpretability

Construct the set of training examples T without class labels, according to the availability of M's train/test set.*

$\mathbf{y} \leftarrow M(\mathrm{T})$

$E \leftarrow (T, \mathbf{y})$

$k_0 \leftarrow 0$, $k \leftarrow 0$, $R \leftarrow \emptyset$

while $E \neq \emptyset$ and $c \times k \geq k_0$ **do**

$k_0 \leftarrow k$

$R \leftarrow R \bigcup$ Horn clauses learned by ILP on E, let S be the size of the newly learned clauses

E_+, $E_- \leftarrow$ examples in E that are correctly and incorrectly classified by R in terms of label $\mathbf{y}$.

$k \leftarrow \frac{|E_+|}{S}$

$E \leftarrow E \backslash E_+ \backslash E_-$

end while

Return R

5.4 Experiment Design and Results

For the task of explaining a fully trained machine learning model, depending on the availability of the training and testing examples, we need to be able to handle 4 different situations: only the training examples are available, only test examples are available, both are unavailable and both are available. Throughout this section, in order to obtain representative results, we will use 5 different opaque machine learning models, including Artificial Neural Networks (ANN), Soft-Margin

Table 5.1: Dataset Size

	Iris	Wine	Lung	Car	Balance
Train size	100	120	22	150	200
Test size	50	58	10	50	100

Support Vector Machine (SVM), AdaBoost with Decision Stumps (AdaStump), Bagging with Random Forest [15] (BagRF) and Naive Bayes (NB). We used the implementations of all the above models from WEKA [107], with WEKA default parameters. As ExOpaque is a general framework, we can choose to apply it to arbitrary models for human interpretability. The datasets are all from UCI machine learning repository [6], including Iris, Wine, Lung Cancer, Car and Balance datasets, which have a good mix of both discrete and continuous attribute values. [22] provides a recent comparison of supervised learning algorithms on various UCI datasets. Since most rule-based learning models require the attributes to be discrete, we discretize all continuous attributes using the standard Minimum Description Length Principle (MDLP) on supervised data [40] before applying ExOpaque. In this section, when referred to, the size of training/testing set of each dataset is given in Table 5.1.

We compare our results with decision tree J4.8 in WEKA, which is a java implementation of the C4.5 decision tree algorithm, in turns in both fidelity and learned rule size. Fidelity is a measure of how similar a learned model is to the original one. Since ExOpaque is trying to explain an arbitrary opaque model, fidelity is a very important measure of ExOpaque's performance. Formally, given a set of examples $\mathbf{x}=(\mathrm{x}^1,\ldots,\mathrm{x}^T)$, if the predictions of model M_1 and M_2 on $\mathbf{x}$ are $\mathbf{y}_1=(\mathrm{y}_1^1,\ldots,\mathrm{y}_1^T)$ and $\mathbf{y}_2=(\mathrm{y}_2^1,\ldots,\mathrm{y}_2^T)$, the fidelity between M_1 and M_2 on dataset

$\mathbf{x}$ is defined as $\frac{1}{T}\sum_{i=1}^{T}\Delta(y_1^i, y_2^i)$ where Δ is the normal 0-1 loss function. As a decision tree size is not directly comparable to a set of Horn clauses, we translate each decision tree leaf node into a single Horn clause (body elements represent tree splits, and head element is the classification at leaf node), and the size of a decision tree is the sum of all such Horn clauses. The size of a Horn clause is defined as the number of body elements plus one, for example, A :- B, C is a Horn clause of size 3.

In addition, although both ExOpaque and J4.8 provide explanations for opaque models, there are a couple of differences to note. Firstly, the rules provided by J4.8 are of equal importance, as no rule can be a generalization of another. While ExOpaque rules are ordered, as the later rules can be generalizations of earlier rules, which provides another level of freedom of manipulating the final rule set: since the earlier rules are more important (examples that can be classified by them would not be matched against later rules), we could prune part of the rules at the end without significantly damaging fidelity. Secondly, although in theory, given an example set with no contradictions (i.e, examples with identical attribute values but different class labels), a decision tree with exponential size can always achieve 100% fidelity, it is not clear how we can control the tradeoff between tree size and fidelity explicitly, while in ExOpaque this is controlled by the parameter c.

5.4.1 Only Training Set Is Available

When we are trying to understand the behavior of a model handed to us with only the examples that the model was trained on, ExOpaque uses the training set with the predicted class labels by the opaque model. In other words, if (x,y) is the training set with y being the true class labels and $\overline{y}$ being the predicted labels,

Table 5.2: Train Set Available Only (ExOpaque; J4.8)

Fidelity	ANN	SVM	AdaStump	BagRF	NB
Iris	100; 99	100; 100	100; 100	100; 99	100; 99
Wine	100; 99.17	100; 100	100; 99.17	100; 99.17	100; 100
Lung	100; 90.91	100; 90.91	100; 100	100; 90.91	100; 90.91
Car	100; 95	100; 93.33	100; 100	100; 94	100; 98
Bal	98.5; 90	100; 94	100; 100	96.5; 89.5	100; 97

Rule Size	ANN	SVM	AdaStump	BagRF	NB
Iris	12; 8	6; 8	6; 8	12; 8	12; 8
Wine	25; 22	32; 22	25; 17	25; 22	31; 22
Lung	27; 12	27; 12	17; 8	27; 12	28; 8
Car	92; 71	78;30	1; 1	96; 66	65; 33
Bal	227; 83	78; 63	10; 4	231; 66	196; 91

we apply ExOpaque on (x,$\overline{y}$) to obtain the set of Horn clauses **H**. The fidelity in percentage and rule size comparisons of **H** with J4.8 is presented in Table 5.2. Each entry in the table consists a pair of numbers, such as (100;99). The first number in the pair is the statistic of ExOpaque and the second number is of J4.8.

From the table we see that ExOpaque is always able to provide high fidelity (above 96%) interpretations of the opaque models, while J4.8 has higher efficiency in providing more compact interpretations with lower fidelity.

5.4.2 Only Test Set Is Available

When we only have the fully trained model and test examples, due to the fact that the test set size is often much smaller than that of the training set, we may not be able to learn a good explanation of the trained model's behaviors from only the predictions on the test set. As a result, we need to first obtain additional examples of similar distributions as the test set. MUNGE [16] is a nice fit for this task, as it takes into a small set of examples and multiply the size of example set by generating additional examples based on the original ones.

We set the multiplier parameter k of MUNGE to be the smallest integer that the resulted example set size is at least the size of original training set, assuming we know the size of the train set. We use Euclidean distances as the distances among examples in MUNGE. Then we apply ExOpaque on the newly generated example set with predictions by opaque models. The results are presented in Table 5.3, which are similar to when only training set is available as ExOpaque has high fidelity (100%) in every case, and the rule size difference between J4.8 and ExOpaque is not as large.

5.4.3 Both Training And Test Sets Are Unavailable

When both training and test sets are missing, we have to generate random instances. Each random instance is generated independently, and each attribute value is selected uniformly at random among all feasible discrete values. And the size of random example set is equal to the original training set. ExOpaque is used to explain the random examples with predictions from various machine learning models. Results are presented in Table 5.4.

Table 5.3: Test Set Available Only (ExOpaque; J4.8)

Fidelity	ANN	SVM	AdaStump	BagRF	NB
Iris	100; 100	100; 100	100; 100	100; 100	100; 100
Wine	100; 99.43	100; 98.28	100; 100	100; 99.43	100; 99.43
Lung	100; 93.33	100; 93.33	100; 100	100; 93.33	100; 93.33
Car	100; 97.33	100; 99.33	100; 100	100; 98.67	100; 98
Bal	100; 99	100; 97	100; 100	100; 98.5	100; 99

Rule Size	ANN	SVM	AdaStump	BagRF	NB
Iris	12; 19	6; 8	6; 8	12; 19	12; 19
Wine	26; 28	26; 28	21; 17	20; 28	24; 33
Lung	23; 13	24; 13	24; 17	36; 24	25; 13
Car	80; 53	49; 47	1; 1	67; 66	51; 54
Bal	164; 178	115; 64	10; 4	130; 86	127; 118

In cases where both train set and test set are available, we can either choose to study the behavior of models on the training or test sets as described in the previous sections.

5.5 ILP Accuracy Improvement without Using Background Knowledge

The background knowledge to an ILP system is often provided in logic programs, describing what we know as truth *a priori* before we see any example. Fort in-

Table 5.4: Both Train and Test Set Unavailable (ExOpaque; J4.8)

Fidelity	ANN	SVM	AdaStump	BagRF	NB
Iris	100; 92	100; 93	100; 100	100; 100	100; 98
Wine	100; 91.67	100; 91.67	100; 99.17	100; 95.83	100; 93.33
Lung	100; 100	100; 100	100; 95.45	100; 90.91	100; 100
Car	100; 96	100; 96	100; 100	100; 94	100; 98
Bal	99; 94.5	100; 96.5	100; 100	98.5; 94	100; 97

Rule Size	ANN	SVM	AdaStump	BagRF	NB
Iris	82; 49	56; 23	20; 19	24; 19	81; 73
Wine	99; 97	75; 51	61; 48	71; 75	37; 1
Lung	19; 13	21; 13	19; 13	19; 8	16; 8
Car	113; 64	52; 25	1; 1	82; 34	61; 52
Bal	195; 103	168; 65	10; 4	182; 83	128; 81

stance, in the task of grammar parsing, a piece of background knowledge can be given as $NP(S_1, S_2)$:- $det(S_1, S_3), noun(S_3, S_3)$.

One well-known common problem of an ILP system is the dependance on supplied background knowledge, which makes it less desirable in situations where such knowledge is hard to obtain. For example, in most of the UCI datasets, only the attribute values are available but not any pre-defined background knowledge. As a result, ILP is not often used for general prediction purposes on such datasets. However, as ILP possesses the nice property of understandability, finding a way to get around the problem can be very useful. During our development of ExOpaque, we have developed a method to address this problem by making use of

high-accuracy opaque machine learning models to boost the prediction accuracy of an ILP system without using background knowledge. The idea is to generate additional training cases, and use the opaque models trained on original training examples to predict them. Then we take those predictions as class labels for the additional examples, and train ILP using these new examples together with the original training set. To the best of our knowledge, we are the first to propose combining the predictions of high-accuracy models with artificial instances to get rid of the background knowledge required by most ILP systems to make accurate predictions on test data. More generally, in situations where a model has some certain desirable properties (as understandability for ILP), but suffers from low predicting accuracy, we propose to use a high-accuracy model to construct random instances to improve the accuracy while maintaining the desired properties.

We will show that by using additional artificial instances we indeed can increase the accuracy of an ILP system. In addition, experiment results also suggest that the accuracy of ILP converges to the accuracy of the opaque model given enough additional instances. The ILP model we used here is the same as in ExOpaque described in section 5.3, and we also use a training and a testing set to evaluate the prediction accuracy. We will use ANN and SVM as 2 representative relatively high accuracy opaque models and try to improve ILP accuracy on the Balance and Car datasets from UCI repository. We start with the full Balance and Car datasets and use ANN, SVM and ILP for the prediction task. The accuracies in percentage of the 3 models are presented in Table 5.5.

We notice that ILP performance in prediction tasks without background knowledge is unstable, as it gives only 52% accuracy on Balance dataset but higher accuracy than SVM on Car dataset. For both datasets, we use random and MUNGE

Table 5.5: Model Performance Using Original Training Set

	Train Size	Test Size	ILP	ANN	SVM
Bal.	400	225	52.00	92.89	90.22
Car	1000	728	87.07	91.21	84.34

algorithm to generate additional instances. The results of ILP accuracy with these additional training examples can be found in Figure 5.1.

In the first 2 plots (Bal.+ANN and Bal.+SVM), we see that by introducing additional random instances with high accuracy model's predictions, ILP's performance actually converges to the high-accuracy model (from 52% to 92.44% for Bal.+ANN and to 90.22% for Bal.+SVM), without the need of background knowledge while maintaining the comprehensibility property. On the other hand, Munge-generated instances are not as good as random ones in this task as the performance converges to some lower accuracy very fast. This is due to Munge's characteristic that it generates new instances based on the original training set, which limits its ability to provide instances scattered in a larger feasible attribute space. As a result, when the multiplier parameter k is large (greater than 3 in our experiments), the new instances are repetitive and condensed which will not provide additional information for ILP even if we further increase the additional training set size. It is surely interesting to take a close look at the other 2 plots (Car+ANN, Car+SVM). In the first one, we can still boost ILP's accuracy by random additional instances, but the converged accuracy is still lower than that of the high-accuracy model (88.87% and 91.21%), while the Munge data is behaving strangely. We believe this is partially due to the fact that the base ILP accuracy is considerably high already, thus the margin for improvement is much

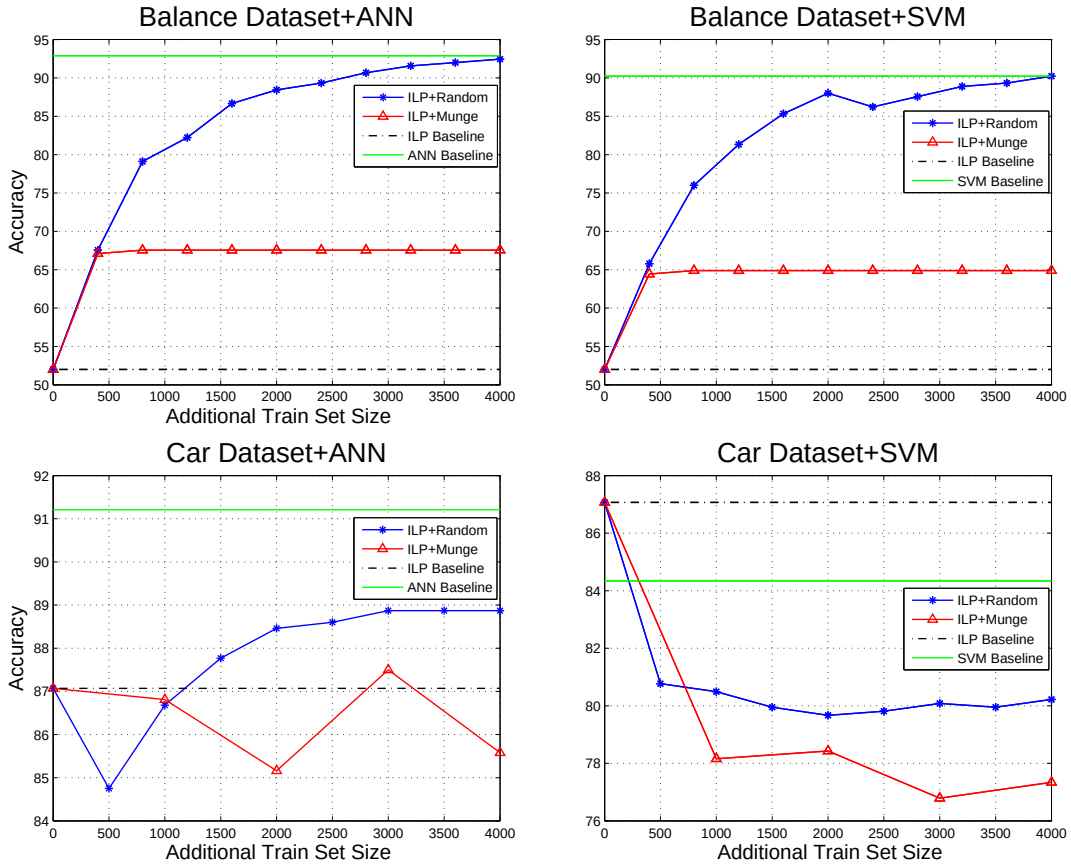

Figure 5.1: Using Additional Training Set to Improve ILP Accuracy w/o Background Knowledge

smaller if compared with the Balance dataset. So within the small margin random fluctuations would appear to be more significant than before. While for the Car+SVM plot, since ILP base line is clearly higher than SVM base line, it is as expected that learning from a worse model would deteriorate accuracy. Again, Munge instances seem to be worse than random instances for the current task. In summary, from our experimental results, we propose to use additional random instances with predictions from models with significantly higher accuracy than ILP system, to train the ILP system in order to achieve comparable accuracy as the other model without using the background knowledge otherwise required.

5.6 Related Work

Trepan [30] is a model used to extract a decision-tree like structure from a trained neural network. While Trepan is also a model aimed at interpretability, its generalization power depends on a special *m-of-n* node which is a tree node that says "if at least m of the following n conditions are satisfied, take the left otherwise right branch". Although this kind of explanation is normal in diagnostic decision criteria according to the authors, we feel such "m-of-n" conditions can be also confusing to people who are looking for more deterministic explanation of a model. In addition, it is not always possible to express one "m-of-n" condition as a simple Horn clause of polynomial size in n, so the total size of the structure extracted by Trepan is indeed more than the tree nodes have shown. Nevertheless, from the experiment results in [30], decision tree model size is better than Trepan model in half of the size comparisons. As a result, we based our comparison with J4.8.

Trepan also makes use of artificially generated instances to compensate the decrease of training data to select splitting tests with the depth of the constructed tree structure.

Bucilia et al [16] used a small size neural network to mimic the behavior of a complex ensemble model. They generated artificial data by random, Munge and NBE (Naive Bayes Estimation) methods. The artificial data is labeled by the ensemble model and used to train a neural network model. Experiment results show that the obtained neural network has similar performance as the ensemble model when Munge is used to generate the additional instances. While they also utilize the idea of using artificial data and predictions of a complex model to train another model, their goal is a smaller model rather than human comprehensibility in this chapter.

CMM (Combined Multiple Models) [35] uses C4.5Rules [89] to learn from a bagged ensemble of the same C4.5Rules base models, which are trained by bootstrap re-sampling of the original training set. In learning from the bagged ensemble CMM apply C4.5Rules on the original training set as well as instances constructed randomly with predictions of the bagged model. Experiments showed that the resulted model can retain 60% of the accuracy gain by bagged model relative to a single run of C4.5Rules. While CMM also returns a comprehensible model, it is not used to explain the behavior of the bagged model, or any other general opaque model; rather, it uses the information from a complex model to extract a simpler one and tries to maintain the accuracy benefit at the same time.

5.7 Discussion

In this chapter we introduced a new ILP-based algorithm ExOpaque to explain the behavior of a trained opaque general machine learning model using Horn clauses. Depending on the availability of train/test data of the opaque model, ExOpaque may need to artificially construct additional random or Munge instances. Empirical evidence shows that ExOpaque is able to describe various trained models with high fidelity but the total rule size is larger than decision tree. There are at least two well-known difficulties faced by most ILP systems: the dependence on background knowledge and scalability. We have proposed a way to use artificial instances and predictions from high-accuracy models to improve ILP system's predicting performances without using pre-defined background knowledge. Empirical evidence suggests that given enough such additional instances, the performance of ILP converges to that of the opaque model. On the other hand, we still suffer from the scalability problem at an acceptable level as most ILP systems do,

as the run-time of our ILP model varies from a few seconds to less than 2 hours. Fortunately, theoretical and practical system implementation progresses have been made in dealing with this problem [108].

CHAPTER 6

CONCLUSION AND FUTURE RESEARCH DIRECTIONS

In many practical optimization applications, it is much cheaper and easier to obtain observations on one's past decisions than the exact preferences which govern those decisions. Therefore, we propose to model these applications as machine learning problems with implicit preferences. In order to achieve high prediction accuracy for new instances, we circumvent the step of explicit preference modeling and focus on correctly inferring the individual's choice if faced with a new instance according to information gathered from historical training data. We discussed several topics in learning with implicit preferences, including structural learning, resource capacity constraint satisfaction, document citation ranking and the interpretation of "black-box" machine learning models.

In chapter 2 we considered structural learning with implicit preferences. Experimental results on both a synthetic problem domain and a real-world face image subset selection problem show that SVM_{OS}, the subset learning algorithm we proposed, significantly outperforms previous learning approaches for such problems. However, SVM_{OS} uses a simple feature map for the purpose of exact inference during the training phase. Consequently, the method is not general enough to model the dependencies among items in the optimal subset. Exploring the use of a more powerful feature map while maintaining the training and testing complexity is part of the future work we would pursue.

In chapter 3 we developed a new SVM based learning model with additional non-convex capacity resource constraints. This model aims to provide high accuracy classification results while enforcing the new global resource capacity constraints. Experimental results on both the pastoral dataset and a synthetic forest

fire dataset show that our method can provide results as accurate as the benchmark SVM classification method, and reduce the amount of resource capacity violation by as much as 90%. When incorporated with our optimal post learning optimization, the violation measures are further reduced while the accuracy is kept competitive with SVM. Furthermore, we identified two natural extensions to this problem. First, by allowing each class to be associated with multiple resource types, each instance can be associated with multiple resource loads. The second potential extension allows an instance to be assigned to multiple classes. For example, in order to satisfy their herding needs African pastoral households can move to multiple water points. Structural learning approaches may suit the new problem better than a simpler classification approach.

Patent prior art search as a learning to rank problem is a challenging task. The intriguing differences of the citation incentive and preference between the patent applicants and patent examiners are reflected in their chosen set of cited patents. With the assumption that an examiner citation is a more objective indication of a patent's relevance to the current application, we learned to predict an ordered list of all possible patent citations that rank the examiner's citations higher than the applicant's in chapter 4. Our algorithm SVM_{PR} serves as an automatic citation recommendation system for a new patent application. The experiments on a real USPTO patent dataset show that SVM_{PR} performs on average 30%-40% better than other state-of-the-art ad-hoc Information Retrieval methods on a wireless technology patent dataset, in terms of the NDCG measure. To the best of our knowledge, this is the first time a machine learning approach is used for automatically generating patent prior art citations, which is traditionally handled by human experts. We plan to incorporate human experts' (a.k.a, the examiners') feedback in the learning loop on why they add new citations on top of applicants'

original ones. This will help both in generating additional informative patent-pair domain knowledge features, and understanding how we can improve our current assumption that examiner citations are more important than applicant citations for constructing our learning to rank algorithm.

ExOpaque, the "interpreter" of opaque machine learning models based on Inductive Logic Programming (ILP) was introduced in chapter 5. We also devised a method to overcome ILP's dependence on background knowledge to produce accurate predictions on datasets where pre-defined background knowledge is hard to obtain. Using our method the ILP prediction accuracy would converge to that of highly accurate opaque machine learning models such as ANN and SVM.

In order to complement the research on problems in the domain of *learning with implicit preference*, we included in the appendix our study on the multi-stage production sourcing problem under international Free Trade Agreements (FTAs) using heuristic methods. We show that for the NP-Complete problem our heuristic outperforms the CPLEX exact solver over a spectrum of dataset instances. Although simplified international trade assumptions are made in this work, the future research goal is to optimize the firm's sourcing when a larger range of implications of FTAs is included. For example, the effects of different FTAs that govern different countries can be factored into the same sourcing problem.

To conclude, this dissertation presents our new approaches in the machine learning domain where user preferences are hard to formulate explicitly but are a determining factor of the observed output. The proposed learning algorithms with implicit preferences combine techniques from various fields such as structural machine learning, data mining, combinatorial optimization, statistics and logic programming. Based on extensive experimental comparisons, our new models often

outperform other contemporary ones in providing accurate and fast predictions.

APPENDIX A

A DIFFERENT PERSPECTIVE: KNOWN PREFERENCES IN A COMBINATORIAL OPTIMIZATION PROBLEM

We explained in the introduction of this dissertation that depending on the availability of explicit preference and constraints, solution approaches for challenging problems differ significantly. For the major part of the earlier chapters we described several machine learning problems with implicit preferences. In this chapter, we demonstrate the effective use of heuristics for an involved optimization problem with known preferences to complement our prior research focus.

There has been a proliferation of preferential and free trade agreements (FTAs) recently [59] adding to those already in place - for example, the European Union (EU), the North American Free Trade Agreement (NAFTA), the Central European Free Trade Agreement (CEFTA), the Australia-United States Free Trade Agreement (AUSFTA), the Japan and Singapore New Age Economic Partnership Agreement (JSEPA), and the China-ASEAN Free Trade Agreement (CAFTA). Many more continue to be shaped.

As firms evolve strategies to compete in international tariff concession environments, "tariff engineering" [66] is beginning to play a larger role in regional and global manufacturing. Companies, such as Steve & Barry's [64], have grown their businesses successfully by exploiting tariff agreements to lower costs. Global sourcing solutions providers, such as Li & Fung (Hong Kong), help customers take advantage of tariff preferences wherever possible [68]. The following is a simple illustration. To satisfy demand from Europe for apparel, Li & Fung procures yarn from a South Korea producer and has it woven and dyed in Taiwan. Zippers and buttons are purchased from Japanese companies located in China. All semi-

finished components are then shipped to Thailand, where production is completed. In this example, other than a preferential trade agreement between South Korea and Taiwan, tariff concessions between China and Thailand (as part of ASEAN, the Association of Southeast Asian Nations), and Thailand (ASEAN) with the EU, impact outsourcing recommendations offered by Li & Fung. For more complex products, e.g., electronic toys, the number of manufacturing stages, which are dispersed regionally and globally, can escalate making outsourcing choices more difficult for the firm.

In this work, we develop a model which allows the firm to make sourcing and plant location decisions to take advantage of tariff concessions in a multi-country environment where FTAs come into play. We study how the firm can leverage its sourcing network using tariff concessions to lower production costs.

A.1 Related Work

Trade agreements have been studied extensively from national, welfare and economic perspectives [62, 69, 24] together with their impact at the macro level on industries [62, 7, 97]. However, in the operations management literature there has been little work on the influence of trade agreements at the firm level. In [81], the authors study models where local content rules force firms to buy components from suppliers in a single country of manufacture. Here, the classical plant location model is extended to factor in local content requirements. [65] extended this by incorporating supplier capacity constraints. [61] provided a mixed integer programming model to design global networks, which incorporated government subsidies, trade tariffs and taxation issues. Their work focuses on special cases and provides useful insights and analysis.

In related operations management literature, early empirical work on international procurement is found in [31], [75], [103], [20], [73] and [76]. The general supplier selection problem has been studied by [77], [43], [9], [60], [101], [25], [105]. Other studies on sourcing include [74], where inventory models in the global environment are provided, and [28].

A.2 The n-Stage m-Country Production Line Design Problem

The problem we study can be described as a multi-stage production line design problem (PLD, in short) in which the firm makes decisions on where to outsource from and/or locate its manufacturing plants taking into account production costs as well as tariff concessions arising from FTAs. Assume that the firm produces a single product to sell in a market located in country D and that the product is manufactured in n stages where one or more stages can occur in any of m countries for which FTA concessions apply; see Figure A.1. Assume, for simplicity, that the firm incurs a fixed production cost for any stage in a given country. Depending on FTA tariff concessions available in its sourcing network located in these countries, the firm has to decide where each stage should be carried out to minimize the total production costs, including tariff costs.

In its prevalent form, an FTA between country i and j allows goods exported from country i into j to be tariff exempt if they originate in i, and vice versa. Tariff exemption or a lower tariff can be claimed if products satisfy rules of origin (ROO), and can qualify as originating from the exporting country i. ROO stipulate a local content rule, which requires that value added (local content) to the product in

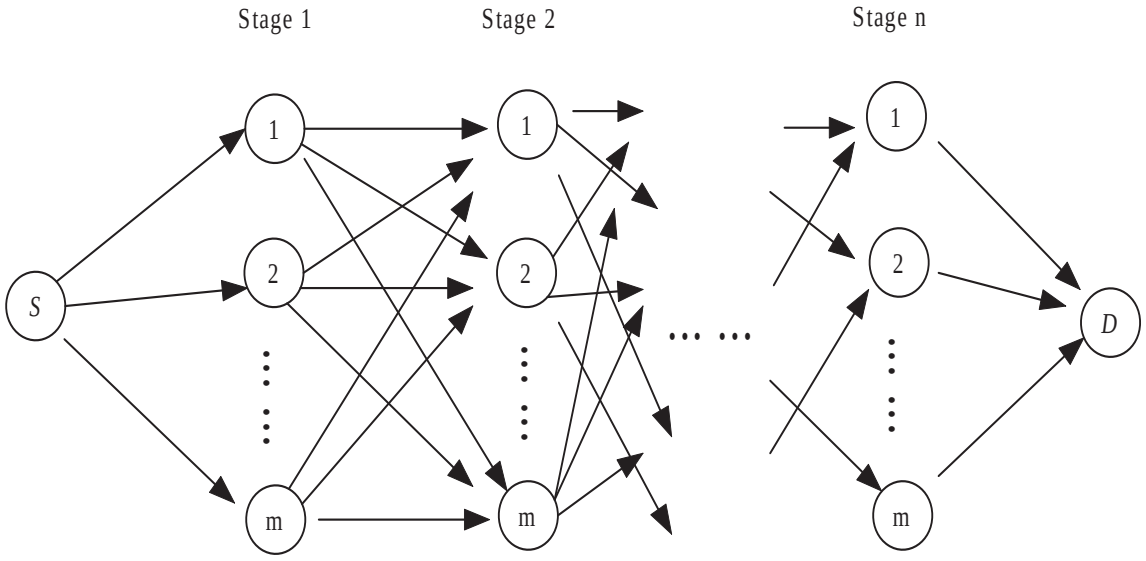

Figure A.1: The n-stage Production Line Design Problem

exporting country i must be no less than a specified percentage of its final total production value.

In order to calculate the value added as production moves from one stage to the next, costs are based on one unit of the product, where a "unit" is a generic term, and can mean a piece, carton etc. Denote the aggregate value of a unit of product up to stage k by V_k, for $k = 1, ..., n$, which includes all costs, including production and transportation costs, and the profit margin, up to stage k. Here, "production cost" is a collective term, and includes raw material cost, labor cost, local production tax, facility cost, factory rental cost, etc. In this model, production costs are taken as the price paid to an outsourced plant by the firm. V_k is commonly referred to as the "free on board" (FOB) price. In this case, FOB is determined by the firm and has a fixed value, similar to the so-called "transfer price" in [104], where products are transported in an internal network.

To calculate value added to a product in a particular country, we use the so-

called "outward processing" method which is used in many FTAs (see, e.g., the JSEPA: http://app.fta. gov.sg/asp/goods/guides.asp, the Singapore- Australian FTA: http://www.fta.gov.au), which takes value added as the cumulative value in a country for all stages of production. To be specific, let A_{ki} be the sum of the value added in country i up to stage k. If stage k occurs in country i and stage $k+1$ in country j, then the value added is used to calculate tariff as follows: if the value added in country i, A_{ki}, taken as a percentage of V_k, is less than a specified value β_{kij}, then tariff equal to α_{kij} (tariff rate) times the product value is incurred. Here, β_{kij} is a value added minimum threshold, i.e. the local content threshold, required for tariff elimination from country i to country j following production stage k. Both are specified in tariff rules in the applicable FTA. Otherwise, if the value added is higher than the threshold, the product is tariff free.

More formally, to describe the problem, the following parameters and decision variables are used:

- m = the number of countries in production network
- n = the number of production stages
- t_{kij} = the unit transportation cost from country i to j, following stage k, where $k = 1, ..., n;\ i, j = 1, ..., m, m+1$, where $m+1 = D$ is the market country
- P_{kj} = the production cost incurred in stage k in country j, i.e., price charged for stage k by the outsourced plant in country j, where $k = 1, ..., n; j = 1, ..., m$.
- $I_{kj} = 1$ if stage k occurs in country j; 0 otherwise, for $k = 1, ..., n;\ j = 1, ..., m$

- $J_{kij} = 1$ if output of stage k is shipped from country i to country j; 0 otherwise, for $k = 1, ..., n;\ i, j = 1, ..., m$:
- T_{kij} = tariff paid to country j if stage k occurs in country i and stage $(k+1)$ occurs in country j, $i \neq j$; $T_{kii} = 0$, for $k = 1, ..., n;\ i,\ j = 1, ..., m+1$

We can now formulate the PLD as an integer program. In the program, the objective is to find an assignment of production stages to countries to minimize the total cost, including production and transportation costs, and tariff costs taking into consideration FTA tariff exemptions that apply between countries.

$$\min \sum_{k=1}^{n}\sum_{j=1}^{m} I_{kj}P_{kj} + \sum_{k=1}^{n-1}\sum_{i=1}^{m}\sum_{j=1}^{m} J_{kij}t_{kij} + \sum_{k=1}^{n-1}\sum_{i=1}^{m}\sum_{j=1}^{m} T_{kij} + \sum_{i=1}^{m}(t_{ni(m+1)}I_{ni} + T_{ni(m+1)}) \tag{A.1}$$

s.t.

$$\sum_{j=1}^{m} I_{kj} = 1, \qquad k = 1, ..., n \tag{A.2}$$

$$I_{ki} = I_{k+1,j} = 1 \Longleftrightarrow J_{kij} = 1$$

$$\Longleftrightarrow J_{kij} + 1 \geq I_{ki} + I_{k+1,j}, \quad k = 1, ..., n-1;\ i, j = 1, ..., m,\ i \neq j \tag{A.3}$$

$$\frac{\sum_{\kappa=1}^{k} I_{\kappa i}P_{\kappa i}}{V_k} < \beta_{kij}\ \&\ J_{kij} = 1 \Longrightarrow T_{kij} = \alpha_{kij}V_k \ \text{ for } k = 1, ..., n,\ i, j = 1, ..., m; \tag{A.4}$$

$$\Longleftrightarrow$$

$$h_{kij}.M \geq \beta_{kij}V_k - \sum_{\kappa=1}^{k} I_{\kappa i}P_{\alpha i} \tag{A.5}$$

$$T_{kij} - \alpha_{kij}.V_k \geq G.(h_{kij} + J_{kij} - 2) \tag{A.6}$$

The equations (2) ensure that each stage is assigned to exactly one country, while (3) ensure that I_{ki} and J_{kij} are consistent. (4) represent tariff threshold constraints, and (5) and (6) result from (4) by introducing binary variables $h_{kij} \in \{0, 1\}$ to transform the implication in (4) [95]. Here, M is a suitably large number. In (6), G is a suitably large number, and both (5) and (6) hold for $k = 1, ..., n-1$, $i, j = 1, ..., m$ and if $k = n$, $j = m + 1$.

Before solving the problem, we show it to be **NP**-Complete.

Lemma. *The PLD problem with a cumulative value add rule is **NP**-Complete.*

Proof. See Appendix B.2. □

A.3 A Multi-Exchange Heuristic Embedded in Simulated Annealing

A solution approach for the PLD which uses a multi-exchange heuristic embedded in a simulated annealing algorithm can be developed. Here, the multi-exchange neighborhood local search is a variant of a very large-scale neighborhood (VLSN) search, which is suitable for this type of problem and motivated by Ahuja et al. [3]. The use of simulated annealing with an adapted VLSN search is new in two aspects: (1) neighborhoods are searched with a heuristic using a constructed estimated improvement graph, whereas in traditional VSLN search, exact improvement graphs are required [4], and (2) VLSN search is embedded into a simulated annealing metaheuristic framework.

Simulated annealing (SA) differs from standard hill-climbing search since it is

Algorithm 7: SAVLSN Framework

read input: $n, m, V_k, t_{kij}, P_{kj}, \alpha_{kij}, \beta_{kij}$

$S \leftarrow Weighted\ Probablistic\ Initial\ Solution\ Generation$

$Temperature \leftarrow T_{max}$; $Iter \leftarrow 0$

while $Iter < Max\_Iter$ and

$Temperature > T\_Terminate$ **do**

with probability 0.5

$Stemp \leftarrow VLSN\_Cycle(S,\ random(2,\ K_{max}))$

with probability 0.5

$Stemp \leftarrow VLSN\_Path(S,\ random(2,\ K_{max}))$

$\delta = value(Stemp) - value(S)$

if $\delta \leq 0$ **then**

$S \leftarrow Stemp$

else

$p = e^{-\delta/Temperature}$

with probability p

$S \leftarrow Stemp$

with probability $1 - p$

reheat()

end if

if $value(S) > best\_value$ **then**

$best\_value \leftarrow value(S)$;

end if

$iter \leftarrow iter + 1$

$Temperature \leftarrow Temperature * C_0$

end while

able to accept down-hill moves which can decrease the quality of the objective function with a probability related to a temperature variable [38]. In Algorithm 7, the framework of the algorithm (called SAVSLN) which uses a SA framework with a multi-exchange heuristic is provided. In this algorithm, a geometric annealing scheme is used, with the constant C_0 taken to be 0.995, where a reheating mechanism is employed whenever an iteration cannot yield a new current solution. This mechanism counters the effect of annealing to allow for a higher chance of diversifying local moves in later iterations. The reheating is geometrically defined by $Temperature = Temperature * (1 + \frac{(1-C_0)}{5})$. From experiments, it was found that once reheating is used, solution quality can be improved by between 1% and 1.5%, on average.

A.3.1 Generating Initial Solutions

Let the array S of length n represent a solution where $S[i]$ is the index of country which stage i is assigned to, $1 \leq S[i] \leq m, 1 \leq i \leq n$. Two methods were used to generate initial solutions. The first is to randomly choose a country for a stage to be processed in, which serves as a comparison for the second method. The second is to use a weighted probability to assign a country index to every stage, by considering the stages 1 to n sequentially. Since there is no tariff cost or transportation cost involved in stage 1 of production, the total cost of stage 1, if assigned to country j, $1 \leq j \leq m$, can be estimated to be the production cost P_{1j}. This is an estimation since the effect of assigning a country index to stage 1 on later decisions for stage 2 to stage n is not known. Define $Q_{1j} = \frac{1}{P_{1j}}$ and $Q_{total} = \Sigma_j Q_{1j}$ and assign j to stage 1 with probability $\frac{Q_{1j}}{Q_{total}}$. This is to increase the chance that stage 1 is processed in countries that have a smaller production

cost. After stage 1 is assigned to a country, continue to decide country indices for stage 2 to n in a similar way, sequentially, except that the estimated cost for assigning country index j to stage k would, in addition, include transportation cost and tariff (if incurred) from the country where stage $(k-1)$ is processed. To decide a country index for the last stage, tariff and transportation cost to the destination is used.

A.3.2 Very Large-Scale Neighborhood Search

Given a solution $\mathbf{S}$, the neighborhood $\mathcal{N}(\mathbf{S})$ is defined as the set of all feasible solutions $\mathbf{S}'$ which are achievable from $\mathbf{S}$ by a single neighborhood move. In general, the larger the neighborhood size $|\mathcal{N}(\mathbf{S})|$ is, the better the solution quality will be after a local move. However, it is often the case that due to a very large number of neighborhood solutions, the running time for a neighborhood move is high. The idea of a VLSN search is based on maintaining a large set of neighborhood solutions while exploring these efficiently. For this, cyclic and path neighborhood exchange moves are used as the local moves.

Neighborhood Structure

For a solution $\mathbf{S}$, define C_j, $1 \leq j \leq m$ by $C_j = \{i \mid S[i] = j, 1 \leq i \leq n\}$, which is the set of indexes of stages processed in country j. A cyclic exchange neighborhood move first selects K different countries $i_1, i_2, ..., i_K$ such that $C_{i_j} \neq \emptyset$, for $j \in \{1, 2, ..., K\}$. In each selected country j, choose stage $t_j \in C_{i_j}$ and reassign stages t_1 to t_K to country C_{i_j}, $j = 1, ..., K$ in a cyclic manner: $S[t_i] := S[t_{i+1}]$ for $i = 1, ..., K-1$, and $S[t_K] := S[t_1]$. Consequently, C_j, $1 \leq j \leq m$ is changed

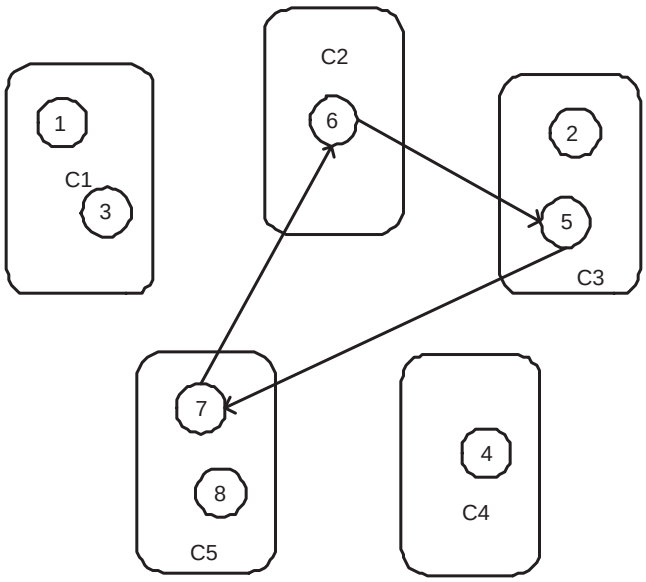

Figure A.2: Example of A VLSN Search Cyclic Exchange with $n = 8,\ m = 5,\ K = 3$

accordingly and the changes take place simultaneously. For the simple example illustrated in Figure A.2, after the local move, the three sets C_2, C_3 and C_5 are changed to $C_2 = \{7\}$, $C_3 = \{2, 6\}$, $C_5 = \{5, 8\}$, while C_1 and C_4 remain unchanged. It is clear that by the K^{th}-cyclic change, the number of neighborhood solutions is $(n/K)^K K!$ assuming the n stages are uniformly allocated in m countries, and in general, the number of neighborhood solutions $\mathcal{N}(\mathbf{S}) = \Omega(n^K)$. When K is allowed to vary linearly with n, the neighborhood size increases exponentially with n. In the algorithm developed here, K_{max} is fixed to be approximately 10% as large as n, and in each iteration of a cyclic local move, K is selected randomly in the range [2, K_{max}]. A neighborhood in path exchange is very similar to a cyclic one although path exchange does not select any stage in C_{i_K} to move to C_{i_1}.

In order to choose K proper stages, the estimated total cost change must be specified when stages are chosen in the local move. The notion of an improvement graph [4] can be used for this. The estimated improvement graph developed here

differs from that used in Ahuja et al. (2004) where the arc weights actually reflect the exact cost change of stages. Since cost calculations in the PLD problem are impossible with only partial information, arc weights in the improvement graph can only be estimates. This is discussed in the next section.

Estimated Improvement Graph

Given a solution $\mathbf{S}$ and C_j, $1 \leq j \leq m$ defined in the previous section, an *estimated improvement graph* is a directed graph $G(\mathbf{S}) = (V, E)$ in which the set of vertices V contains n nodes: v_q, $q = 1, ..., n$ each representing a stage q in the solution $\mathbf{S}$. The arc set E represents the relationship between different stages, where there is a directed arc (q, l) from v_q to v_l if and only if $S[q] \neq S[l]$. The weight of each arc (p, l) is taken to be E_{pl} where:

$$E_{ql} = \begin{cases} P_{q,S[l]} - P_{l,S[l]}, \text{ if } q = 1 \texttt{ or } l = 1 \\ P_{q,S[l]} - P_{l,S[l]} + t_{q,S[q-1],S[l]} - t_{l,S[l-1],S[l]} \\ +T_{q,S[q-1],[S[l]} - T_{l,S[l-1],S[l]}, \quad \text{otherwise} \end{cases}$$

This weight is designed to reflect the total cost change if stage q is reassigned to country $S[l]$ and stage l reassigned to some other country. The above function E_{kl} can only be an estimation because the exact transportation cost and tariff cost for stage q in country $S[j]$ are not available until all stages $1, ..., q-1$ are fixed. However, these $q-1$ stages may be used in the current cyclic/path exchange, and therefore cannot be fixed yet.

Probabilistic Selection of Cycles and Paths

Once the estimated improvement graph has been constructed, the algorithm first randomly chooses K countries $C_{i_j},\ j = 1, 2, ..., K$. If it is a cyclic neighborhood move, $C_{i_j} \neq \emptyset$, while for a path exchange, $C_{i_K} = \emptyset$ is possible. In the neighborhood search, a stage in C_{i_1} is chosen to be included as the first stage in the cycle/path as follows: Let the production cost P_{ji_1} be the indicator of the preference to choose stage j originally in C_{i_1} for all j such that $j \in C_{i_1}$. Define $P_{total} = \Sigma_j P_{ji_1}$. Then stage $j \in C_{i_1}$is selected by the cyclic neighborhood change with probability $\frac{P_{ji_1}}{P_{total}}$. This procedure is similar to when the country index is generated for the first stage in the weighted probability initial solution generation described in section 3.1. However, there are two differences. The first is the fact that since we are selecting stages to "move away" from the currently assigned countries, assigning a larger probability to stages that have a large production costs rather than those which have a small production costs is preferred. The second is that probabilities are used to select a stage when the country index C_{i_1} is fixed, unlike in initial solution generation when the stage is fixed and the country index is selected.

When a stage from country C_{i_1} is selected, one stage for each of the remaining $K-1$ countries is selected sequentially to be used in the cyclic exchange move. Let the index of the selected stage from country C_{i_j} be l_j for $j = 1, 2, ..., K$. The selection of l_j is based on the value of l_{j-1} for $2 \leq j \leq K$. In the estimated improvement graph, there should be an arc from the node representing l_{j-1} to every node in C_{i_j} by definition. A negative arc weight indicates a potential improvement in solution quality if the exchange local move is made to contain the stages associated with this arc. Arc weights are modified in the following way: first, multiply these by -1 and then add a minimum positive number to the arcs to make all weights positive.

For example, weights $\{1, 2, 3, -4, -5\}$ are changed to $\{-1, -2, -3, 4, 5\}$ and then to $\{3, 2, 1, 8, 9\}$. This is to facilitate later calculations of probabilities used for selecting each stage in C_{i_j}. Let E'_{pq} be the modified arc weight from stage p to stage q and $Arc_{total} = \Sigma_{q \in C_{i_j}} E'_{l_{i-1}q}$ and then select stage $q \in C_{i_j}$ in the cyclic exchange neighborhood move with probability $E'_{l_{i-1}q}/Arc_{total}$. If there is a cyclic exchange, the cost from C_{i_K} to C_{i_1} is included using the arc weight E_{ql_1} to determine the probability of selecting the stage $q \in C_{i_K}$. If the local move is a path exchange, it is not necessary to select a stage from C_{i_K}. When l_i for $i \in \{1, 2, ..., K\}$ are fixed, a cyclic/path exchange is performed to complete an iteration of the neighborhood search.

A.4 Numerical Experiments

A.4.1 Test Instances Generation

As the problem is new, no benchmark test sets have been established. In order to determine the effectiveness of the solution approach, test instances were generated to represent realistic situations as far as possible. These include using the following attributes for the test instances:

- Product value increases as more stages are completed. Specify the following parameters: S_i is the base product value for the i^{th} one third stages, $i = 1, 2, 3$; inc_i, the exponential increment of product value compared with the value in the previous stage; and d_i and u_i, is the variance in product value. For stages 1 to $\lfloor n/3 \rfloor$ which belong to the first one third stages, the FOB

values of output of stage j are defined by:

$$V_j = S_1 * (1 + inc_1)^{(j-1)} \times \texttt{Unif}\ [1 - d_1, 1 + u_1],\ \text{for } j \in \{1, 2, ..., \lfloor n/3 \rfloor\}$$

where Unif $[x, y]$ generates a real number uniformly in $[x, y]$. The FOB values for the remaining stages are calculated in the same way with the respective parameters.

- Some countries have a relatively lower production cost (e.g. for labor intensive work in Asia) for some stages of production. This is addressed by introducing parameters d_{ij} and u_{ij}, where $i \in \{1, ..., n\}$ and $j \in \{1, ..., m\}$, which specify how much change the production cost of stage i in country j can have compared with V_i:

  $$P_{ij} = V_i \times \texttt{Unif}\ [1 - d_{ij}, 1 + u_{ij}],\ \text{for } i \in \{1, ..., n\},\ j \in \{1, ..., m\}$$

  The $n \times m$ matrices $\mathbf{d} = [d_{ij}]$ and $\mathbf{u} = [u_{ij}]$ are assigned values to represent country preferences for different stages. For example, if country j is preferred in the first third of the stages, higher d_{ij} and lower u_{ij} values can be used for stages 1 to $\lfloor n/3 \rfloor$.

- The transportation cost t_{kij} is taken to be proportional to the product of the distance between points and the value V_i, with a 10% variance limit.

- Tariff rates and value add threshold values are specified by the parameters *tar* and *lcr* which range between 20% and 50% in value. Both are subject to a 40% variation limit.

A.4.2 Experiments on the Parameter K_{max}

In order to evaluate the performance of the SAVLSN algorithm, we investigated the effect of the most important parameter of the VLSN local search method, i.e.,

Table A.1: Comparing $SAVLSN_{0.1}$ and $SAVLSN_{0.5}$

n	m	$\mu_{SA_{0.1}}$	$\sigma_{0.1}$	t_1	$\mu_{SA_{0.5}}$	$\sigma_{0.5}$	t_2	δ
60	30	8.02	0.34	1.21	8.13	0.35	3.16	1.37%
80	30	114.04	5.86	1.25	117.55	7.85	4.29	3.04%
100	50	1740.83	48.61	2.25	1840.11	78.05	5.85	5.70%
100	80	1765.77	82.17	2.49	1924.26	90.09	22.34	8.98%
120	50	2489.85	698.92	14.39	27446.21	1147.35	35.11	10.23%

K_{max}, on performance and compared the SAVLSN algorithm between $K_{max} = 0.1n$ and $K_{max} = 0.5n$. Since there are no established benchmarks, 5 groups of test sets were generated using different scales. Each test set group consists of 20 test instances generated according to the categories described in the previous section. To ascertain the effectiveness of the VLSN search used in the simulated annealing framework, the acceptance rate of the local move in the framework for different iteration ranges was compared. All the experiments were conducted on P4, 1.4Ghz CPU with 256Mb of memory.

The experimental results are provided in Table A.1, where $\mu_{SA_{0.1}}$ and $\mu_{SA_{0.5}}$ is the average costs obtained by the SAVLSN algorithm using $0.1n$ and $0.5n$ for K_{max}, respectively; σ_{SA} measures the standard deviation of the results for the 20 test instances in each group, and t_1 and t_2 is the average running time in seconds for each instance; δ measures the difference between solutions of $SAVLSN_{0.1}$ over $SAVLSN_{0.5}$, as a percentage. From the table, it can be seen that setting $K_{max} = 0.1n$ provides 1% to 10% better results compared with $K_{max} = 0.5n$. This results from the fact that a large estimated cycle length will easily lose the advantage for small adjustments. In addition, the standard deviation of the results was higher

than those for $SAVLSN_{0.1}$. Both parameter settings result in fast running times of less than 40 seconds, where the $SAVLSN_{0.1}$ algorithm obtains results within 15 seconds. The longer running time for $SAVLSN_{0.5}$ is expected in view of the longer cycle local search. From the experiments, for different problems, setting K_{max} to equal $0.1n$ was found to be a good choice.

To further analyze the use of simulated annealing for the $SAVLSN_{0.1}$ and $SAVLSN_{0.5}$ algorithms, we recorded the acceptance rate of the VLSN local move in simulated annealing framework for iterations ranging from 1 to 1000 without reheating since this affects the natural acceptance rate of the algorithm. The acceptance rate was recorded for every 100 iterations performed. The results for the 20 test instances for $n = 100$ and $m = 50$ can be found in Figure A.3. Each point with x-axis value i represents the acceptance rate for iteration $100(i-1)$ to $100i$. Other groups of test instances had similar acceptance rates. The initial acceptance rate for simulated annealing should ideally be approximately 60% [72], which explains why $SAVLSN_{0.1}$ with an initial acceptance rate near to 65% performed better than $SAVLSN_{0.5}$ which had an acceptance rate of about 28%.

A.4.3 Comparing Initial Solution Generation

A weighted probability method was used to generate the initial solution, which was compared with a random generation method. Instances with $n = 80$ and $m = 30$ were used to compare performance, and all parameters were assigned the same values. These recorded close running times of 19.23 and 17.38 seconds, for the weighted and random probability method, respectively. The results using the weighted probability method were 11.3% better than those from the random method. Although solution quality can be improved by multi-restarts, each re-

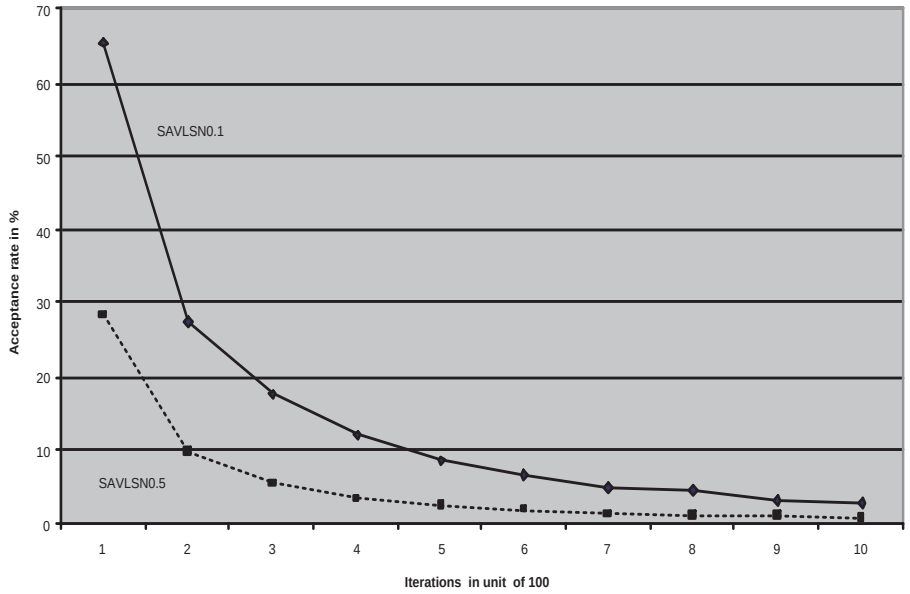

Figure A.3: Acceptance Rate for the First 1000 Iterations

quiring only a few seconds, the focus is on solution convergence; hence, an initial solution was fixed in each run. Experimental results are presented in Figure A.4.

The x-axis represents the running time consumed by each method at that point, and the y-axis gives the difference of the current solution to the best solution obtained by the weighted probability method, as a percentage. The value at $x = 0$ is the initial solution. It is not surprising that a weighted probability led to better starting solutions since this method takes the various costs into consideration. Both methods converge very quickly, to within 10% of the final value within 3 seconds. From this, we conclude that the weighted probability method provides better performance in both solution quality and convergence rate.

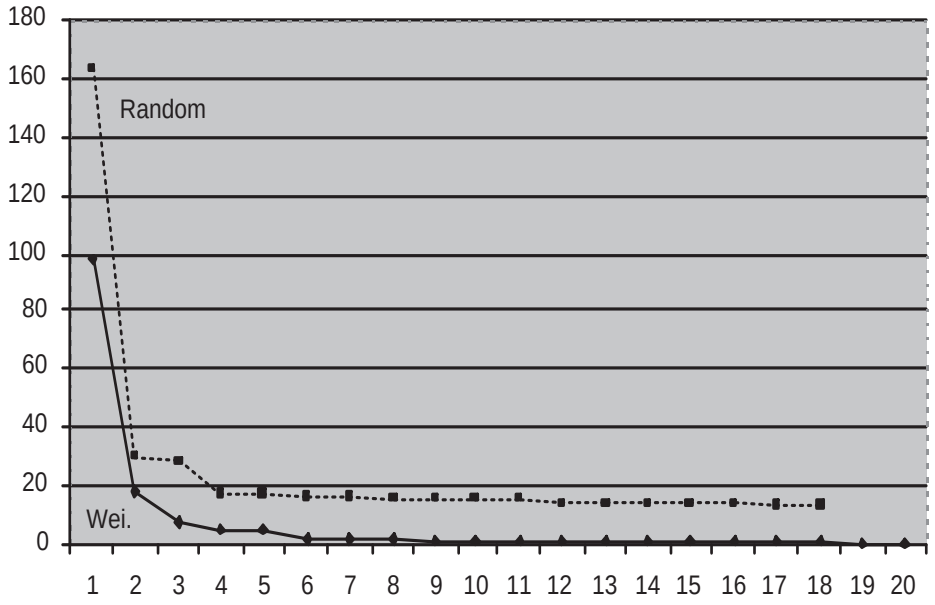

Figure A.4: Convergence of Initial Solution Generation Methods

A.4.4 CPLEX and the SAVLSN

Comparisons were made between the SAVLSN and ILOG CPLEX 9.0. Here, K_{max} for the SAVLSN algorithm is taken to be $0.1n$. From the experiments, it was determined that even when both m and n are of moderate size - 40 to 60 -, CPLEX was not able to obtain a feasible solution within arbitrary time and terminated with an "out of memory" error. Hence, for comparisons, test instances were limited to the largest range for which CPLEX was able to obtain feasible solutions. Two groups of small test instances were generated, each with 21 test cases starting from $n = 5$, $m = 5$, and increasing to $n = 30$ and $m = 30$. A time limit 10,000 seconds was set for CPLEX, and if after this time an optimal solution was not found, the best solution obtained is reported. Instances in the first group had a fixed tariff rate of 30% and value added threshold of 30%, both with up to 20%

variation, while instances of the second group had a tariff rate 50% and value added threshold of 50%, both with up to 20% variation. The results for the first 21 instances is provided in Table A.2, where t_1 and t_2 is the time used by CPLEX and SAVLSN, respectively. If CPLEX did not complete within 10,000 seconds, the t_1 column shows a hyphen - ; the lower bound obtained by CPLEX at 10,000 seconds is presented in the "LB" column. Ratio_1 is the ratio of solution value from the SAVLSN algorithm over the solution from CPLEX, and Ratio_2 is the ratio of the solution from the SAVLSN algorithm over LB.

Among these instances, 9 (42.9%) were optimal; 3 (14.3%) were better than CPLEX solutions by a percentage of 0.3% to 7%, obtained within 10,000 seconds; 6 (28.6%) were within 1% of CPLEX's solutions, and only 3 instances were worse than CPLEX's results by 1% to 5%. Nevertheless, CPLEX obtained 15 optimal results out of the 21 instances. For instances where optimal solutions were not obtained, SAVLSN was always within 10% of the lower bound. The SAVLSN algorithm was more stable providing the solutions within 10 to 25 seconds, while CPLEX's time requirement was approximately exponential to the size of instance. For the 9 instances where CPLEX gave better results by 0.11% to 4.08%, CPLEX used 8 to 545 times the running time required by the SAVLSN algorithm. In addition, considering the heuristic nature of the SAVLSN algorithm and the fact that we limited the instance size when using CPLEX, we can say that the SAVLSN algorithm has an advantage over CPLEX.

The results of the second group of experiments are given in Table A.3. For this group, for two instances (25_15, 30_20), the SAVLSN algorithm's results were more than 5% worse than those from CPLEX. This can be explained by the fact that the SAVLSN algorithm uses a cyclic/path exchange local move, and although the path

Table A.2: Experimental Results for Instances with A 30% Tariff and Value-added Threshold

Size	CPLEX	t_1	LB	SAVLSN	t_2	Ratio_1	Ratio_2
5_5	224.772	1.00	-	224.772	10.98	100.00%	-
10_5	440.257	1.00	-	440.257	26.21	100.00%	-
10_10	455.606	6.00	-	455.606	19.93	100.00%	-
15_5	692.283	1.00	-	693.864	15.15	100.23%	-
15_10	711.747	6.00	-	711.747	14.89	100.00%	-
15_15	580.890	17.00	-	580.890	16.22	100.00%	-
20_5	1481.703	1.00	-	1481.703	17.89	100.00%	-
20_10	1527.546	105.00	-	1527.546	16.55	100.00%	-
20_15	1394.503	461.00	-	1394.503	17.48	100.00%	-
20_20	1078.740	1353.00	-	1122.738	17.48	104.08%	-
25_5	2000.440	1.00	-	2003.950	21.74	100.18%	-
25_10	2203.780	157.00	-	2206.119	20.35	100.11%	-
25_15	1768.260	1345.00	-	1822.545	19.96	103.07%	-
25_20	1686.560	-	1683.990	1726.465	18.36	102.37%	102.52%
25_25	1800.390	-	1699.850	1794.241	19.12	99.66%	105.55%
30_5	2577.530	5.00	-	2577.534	23.40	100.00%	-
30_10	2620.060	599.00	-	2631.067	23.52	100.42%	-
30_15	2229.780	6936.00	-	2247.729	22.30	100.80%	-
30_20	2474.420	-	2301.620	2477.636	21.34	100.13%	107.65%
30_25	2652.290	-	2394.690	2600.158	21.61	98.03%	108.58%
30_30	2423.700	-	2117.080	2258.655	22.60	93.19%	106.69%

exchange local move can possibly decrease the number of countries involved in the n stages of production, the cyclic exchange move never decreases this number. This is a disadvantage of the SAVLSN algorithm, which is not as efficient as CPLEX in exploring search regions with a very small number of countries when compared with the number of stages.

However, CPLEX failed to generate satisfactory solutions for instances with slightly larger size. In the experiments, 3 additional instances (40_40, 50_50 and 60_60) were used with tariff rate and value added threshold both fixed at 30%. CPLEX was allowed to run for 10,000 seconds for both the first two instances without providing optimal solutions. For the 60_60 test instance, CPLEX was not able to find a feasible solution while the SAVLSN algorithm obtained a solution within 70 seconds. For the 40_40 instance CPLEX's solution was 4.6% worse than that from the SAVLSN algorithm, obtained in 40 seconds; for the 50_50 instance CPLEX's solution was 22.8% worse than the solution from the SAVLSN algorithm, obtained in 48 seconds. Moreover, there was a 27% and 32% difference between CPLEX's solution and the lower bound for the 40_40 and 50_50 instances, respectively. The SAVLSN algorithm was able to handle larger test instances more efficiently than CPLEX, and is better suited for problems with a large number of stages.

To compare the speed of the SAVLSN algorithm and CPLEX, an important factor in practical implementation, the SAVLSN algorithm solutions were found following which solutions were obtained by CPLEX which were no worse than those from the SAVLSN algorithm, and the time consumed by CPLEX recorded. The results are presented in Table A.4 and Table A.5 for the cases with tariff rate and value added both equal to 30% and both equal to 50%. *Ratio* is the time

required by CPLEX divided by time used by the SAVLSN algorithm. The tables do not include cases where both methods obtained optimal solutions; here time performance can be directly compared from Table A.2 and A.3. From the two tables we see that, to achieve the same performance as the SAVLSN algorithm, CPLEX required significantly more time, ranging from several times to hundreds of times more. The only exceptions were when the instance sizes are small, in which case a branch-and-cut exact method such as CPLEX is expected outperform a heuristic.

A.4.5 Comparison with Greedy Heuristics

Having examined the efficiency of the algorithm against CPLEX, we examined how tariff concessions can benefit companies which are able to exploit it, and how much the heuristic approach to the problem can reduce costs when compared with intuitive and widely-used greedy production strategies. In particular, we consider the following intuitive greedy strategies adopted in practice [66]:

- *Material Oriented Strategy* (MATS)*:* Choose the country for a particular production stage with the smallest production cost.
- *Market Oriented Strategy* (MORS): Choose production locations nearest to the market and process all stages in this country.

To better understand the impact of tariff concessions on planning decisions, we compared the performance of the SAVLSN method with tariff benefits (SALVSN), the SAVLSN without tariff benefits (SAVLSN w/o FTA), and the greedy MATS and MORS approaches.

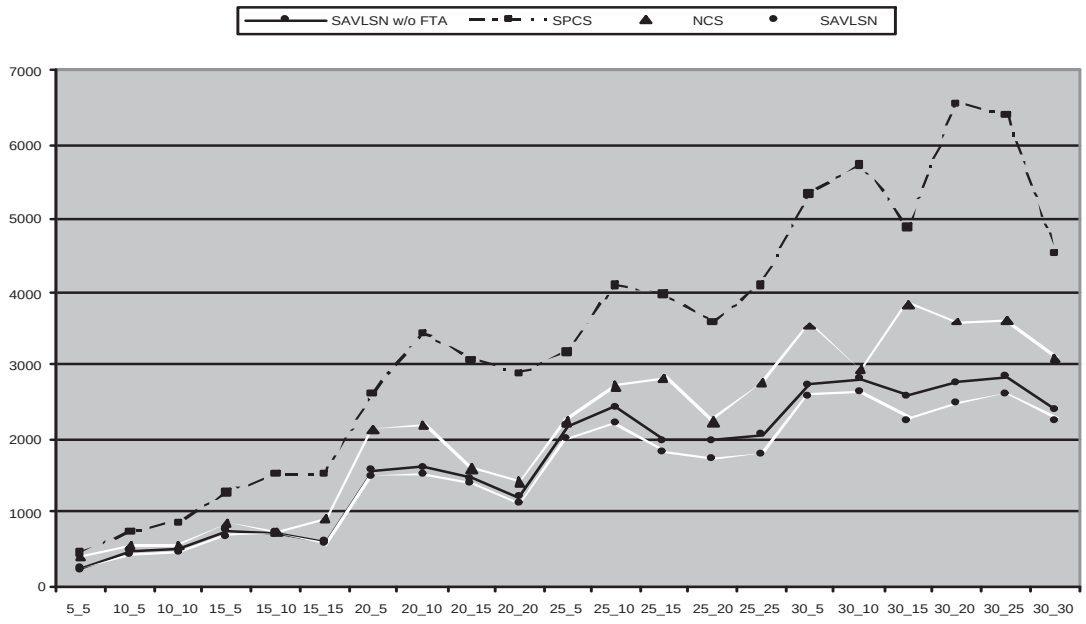

Figure A.5: Comparisons with Greedy Approaches

Experimental results are presented in Figure A.5. The x-axis represents the size of the test instances in the form of n_m, where n and m represent the number of stages and number of countries, respectively. All test instances use a fixed 30% tariff rate and 30% value added threshold, with a 20% variation limit. The y-axis represents the best total cost obtained. From the figure, the SAVLSN algorithm with tariff benefits results in the least cost when compared with the other approaches. The gap between the SAVLSN algorithm and MORS approach varies from 20% to 50% as test sizes increase. Moreover, the difference between the MATS approach and the SAVLSN algorithm can be as much as 100% for large instances. SAVLSN performs better than the greedy algorithms. As instance size increases, the advantage from tariff concessions increases since the larger number of stages and countries allow for greater choice. The SAVLSN using tariff concessions provided a 10% cost reduction when compared with the SAVLSN without

concessions.

A.5 Discussion

In this chapter, we studied a multi-stage production sourcing problem where free trade tariff concessions can be exploited to lower costs. The model developed is new, and adds to the existing - but sparse - literature in operations management. It can be used at the firm level to design a supply chain which spans a number of countries where free trade agreements come into play. To solve the problem, we used a multi-exchange heuristic which employs a VSLN search based on an estimated improvement graph. This method of embedding a VSLN neighborhood search in a simulated annealing framework is new and shown to be effective in the numerical study conducted. In particular, the multi-exchange heuristic was found to be superior in speed to a commercially-available solver, and is able to provide solutions for larger, more realistic problem sizes.

Now we have completed our discussion on combinatorial optimization problems where exact preferences are known, to complement our earlier study with implicit preference learning.

Table A.3: Experimental Results for Instances with A 50% Tariff and Value-added Threshold

Size	CPLEX	t_1	LB	SAVLSN	t_2	Ratio_1	Ratio_2
5_5	293.538	1.00	-	293.538	37.99	100.00%	-
10_5	402.631	1.00	-	402.631	24.11	100.00%	-
10_10	424.003	3.00	-	424.003	24.48	100.00%	-
15_5	770.252	1.00	-	770.252	30.87	100.00%	-
15_10	749.333	63.00	-	749.333	55.62	100.00%	-
15_15	629.977	47.00	-	629.977	28.77	100.00%	-
20_5	1291.777	2.00	-	1291.777	35.06	100.00%	-
20_10	1282.450	14.00	-	1295.130	34.32	100.99%	-
20_15	1323.949	299.00	-	1323.949	34.16	100.00%	-
20_20	1287.680	1947.00	-	1308.052	33.51	101.58%	-
25_5	2149.300	4.00	-	2200.713	39.16	102.39%	-
25_10	1957.030	407.00	-	1963.642	38.31	100.34%	-
25_15	1886.950	744.00	-	2041.687	37.58	108.20%	-
25_20	1850.320	4807.00	-	1890.099	45.74	102.15%	-
25_25	1729.000	-	1698.430	1800.747	43.82	104.15%	106.02%
30_5	2542.350	14.00	-	2547.072	42.63	100.19%	-
30_10	2856.920	1163.00	-	2871.363	44.22	100.51%	-
30_15	2430.150	-	2353.270	2440.994	44.62	100.45%	103.73%
30_20	2288.420	-	2144.240	2452.440	42.01	107.17%	114.37%
30_25	2465.600	-	2218.340	2518.320	41.62	102.14%	113.52%
30_30	2618.950	-	2346.150	2573.357	45.27	98.26%	109.68%

Table A.4: Time Comparisons with Tariff Rate and Value-added Threshold Equal 30%

Size	CPLEX time	SAVLSN time	Ratio
20_20	491	17.48	28.9
25_5	2	21.74	0.09
25_10	124	20.35	6.09
25_15	30	19.96	1.50
25_20	182	18.36	9.91
25_25	12718	19.12	665.17
30_10	507	23.52	21.56
30_15	4619	22.30	207.13
30_20	8935	21.34	418.70
30_25	38855*	21.61	1798
30_30	44104*	22.60	1951

*: CPLEX ran out of memory at the time result was recorded.

Table A.5: Time Comparisons with Tariff Rate and Value-added Threshold Equal 50%

Size	CPLEX time	SAVLSN time	Ratio
20_10	10	34.32	0.29
20_20	108	33.51	3.22
25_5	2	39.16	0.05
25_10	370	38.31	9.66
25_15	38	37.58	1.01
25_20	1013	45.74	22.15
25_25	1498	43.82	34.18
30_5	11	42.63	0.26
30_10	597	44.22	13.50
30_15	6730	44.62	150.83
30_20	95	42.01	2.26
30_25	351	41.62	8.43
30_30	39364.19*	45.27	869.54

*: CPLEX ran out of memory at the time result was recorded.

APPENDIX B

NP-COMPLETE PROOFS

B.1 The SCP Problem from Chapter 2

Definition. Subset Computation Problem (SCP)

Input: ground set x, preference $\mathfrak{p}$, an item $I \in x$

Decision Question: Is $I \in \mathfrak{p}(x)$?

Lemma SCP is **NP**-Complete.

Proof: This can be shown by reducing the binary knapsack problem to SCP. SCP is in **NP**, because if we can non-deterministically put the other $|x| - 1$ items into the subset, we can check if any such subset with item I equals $\mathfrak{p}(x)$.

To prove SCP is **NP**-hard, we reduce the binary knapsack problem to it. Given a knapsack instance, with n items, each with value v_i and weight w_i ($1 \leq i \leq n$), and the maximum weight W and threshold value K, we can construct an instance of SCP as follows: let x be the set of all n items, $\mathfrak{p}$ be the preference that given x, select the subset with the largest sum of values not exceeding the weight constraint W. Then it is clear that the knapsack problem has a solution with value at least K, if and only if among the n items, $\sum_{I \in \mathfrak{p}(x)} v_I \geq K$, which can be obtained by inquiring SCP n times. ■

e

B.2 The PLD Problem from Appendix A

Lemma *The PLD problem with a cumulative value add rule is* **NP**-*Complete.*

Proof: It is easy to see that PLD is in **NP**. **NP**-hardness is shown by reduction from the **NP**-hard 2-PARTITION problem [44]: Given an integer set $\{a_1, a_2, ..., a_n\}$ with $\sum_{i=1}^{n} a_i = 2M$, can we find a subset S with $\sum_{a_i \in S} a_i = M$?

An instance of PLD problem can be constructed as follows. Suppose only two countries are available, the production line consists of $(n+2)$ stages, transportation costs is negligible and production costs are given as in Figure B.1:

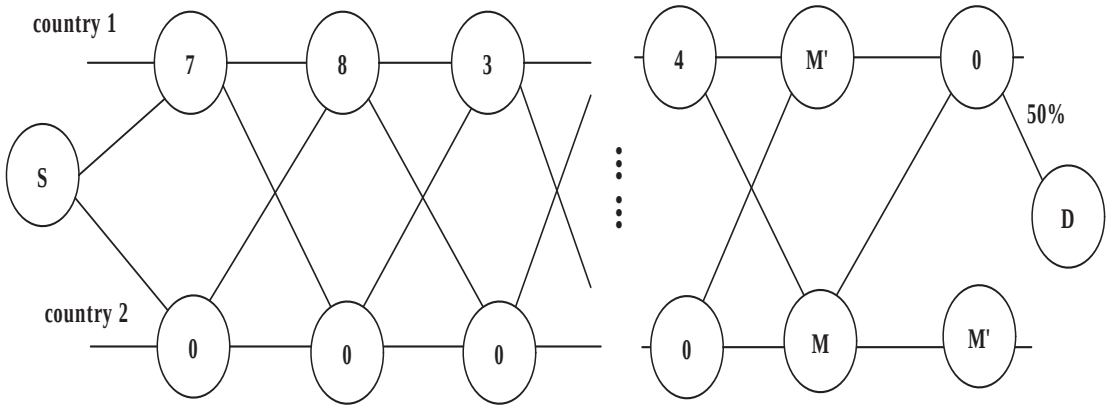

Figure B.1: 2-country Multi-stage Sequencing

From stages 1 to n in country 1, take these to be $a_1, a_2, ..., a_n$, and in country 2 take these to be 0 (negligible). Stage $(n+1)$ can only occur in country 2 with production cost M and stage $(n+2)$ can only occur in country 1 if we assume $M' \gg M$. The threshold value is specified as follows: it is zero between country 1 and 2 (i.e., there is no tariff between them), and 50% for final export from country 1 to the market D.

Suppose tariff imposed from country 1 to country D is high in the absence of tariff concession. The local value for the final product must be satisfied since final stage is carried out in country 1, which is at least 50% of the total value. We know stage $(n+1)$ occurs in country 2 with value added M, so the value added in country 1 cannot be less than M, i.e., $V_{n+2} = 2M$ is the minimum possible cost. This requires that value added in country 1 from stages 1 to n to be exactly M. Hence, once the problem is solved, we know if a feasible solution to 2-PARTITION problem can be found, i.e., if the total cost is $2M$, then the answer is "yes"; otherwise it is "no". ■

BIBLIOGRAPHY

[1] Victor Aguirregabiria and Pedro Mira. Dynamic discrete choice structural models: A survey. *Journal of Econometrics*, Forthcoming.

[2] Ravindra K. Ahuja and James B. Orlin. Inverse optimization. *Operations Research*, 49:771–783, 2001.

[3] R.K. Ahuja, O. Ergun, J.B. Orlin, and A.P. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123:75–102, 2002.

[4] R.K. Ahuja, J.B. Orlin, S. Pallottino, M.P. Scaparra, and M.G. Scutella. A multi-exchange heuristic for the single source capacitated facility location. *Management Science*, 50(6):749–760, 2004.

[5] Juan Alcacer and Michelle Gittelman. How do i know what you know? patent examiners and the generation of patent citations. *Review of Economics and Statistics*, 88(4):774–779, 2006.

[6] A. Asuncion and D.J. Newman. UCI machine learning repository, www.ics.uci.edu/∼mlearn/MLRepository.html, 2007.

[7] J. Bair and G. Gereffi. Upgrading, uneven development, and jobs in the North American apparel industry. *Global Networks - A Journal of Transnational Affairs*, 3(2):143–169, 2003.

[8] Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explorations*, 6(1):20–29, 2004.

[9] P. S. Bender, R. W. Brown, M. H. Isaac, and J. F. Shapiro. Improving purchasing productivity at ibm with a normative decision support system. *Interfaces*, 15(3):106–115, 1985.

[10] Francesco Bergadano and Daniele Gunetti. *Inductive Logic Programming: From Machine Learning to Software Engineering*. MIT Press, Cambridge, MA, USA, 1995.

[11] Maxim Binshtok, Ronen I. Brafman, Solomon E. Shimony, Ajay Martin, and Craig Boutilier. Computing optimal subsets. In *Conference of the Association for the Advancement of Artificial Intelligence (AAAI)*, 2007.

[12] R. I. Brafman, C. Domshlak, S. E. Shimony, and Y. Silver. Preferences over sets. In *Conference of the Association for the Advancement of Artificial Intelligence (AAAI)*, 2006.

[13] Ronen I. Brafman, Carmel Domshlak, and Solomon E. Shimony. On graphical modeling of preference and importance. *Journal of AI Research*, 25, 2006.

[14] Leo Breiman. Bagging predictors. *Journal of Machine Learning*, 24(2):123–140, 1996.

[15] Leo Breiman. Random forests. *Journal of Machine Learning*, 45(1):5–32, 2001.

[16] Cristian Bucilu, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006.

[17] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the International Conference on Machine Learning*, 2005.

[18] C. J. C. Burges, R. Ragno, and Q.V. Le. Learning to rank with non-smooth cost functions. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*, 2006.

[19] Yunbo Cao, Jun Xu, Tie Yan Liu, Hang Li, Yalou Huang, and Hsiao Wuen Hon. Adapting ranking svm to document retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference*, 2006.

[20] J. R. Carter and R. Narasimhan. Purchasing in the international marketplace: Implications for operations. *Journal of Purchasing and Materials Management*, 26(3):2–11, 1990.

[21] Ben Carterette and Desislava Petkova. Learning a ranking from pairwise preferences. In *Proceedings of the annual international ACM SIGIR conference*, 2006.

[22] Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, 2006.

[23] Olivier Chapelle, Quoc Le, and Alex Smola. Large margin optimization of ranking measures. In *NIPS Workshop on Machine Learning for Web Search*, 2007.

[24] Kerry A. Chase. Economic interests and regional trading arrangements: The case of NAFTA. *International Organization*, 57(1):137– 174, Win 2003.

[25] S. S. Chaudhry, F. G. Forst, and J. L. Zydiak. Vendor selection with price breaks. *European Journal of Operational Research*, 70(1):52–66, 1993.

[26] Nitesh V. Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explorations*, 6(1):1–6, 2004.

[27] Wei Chu and S. Sathiya Keerthi. New approaches to support vector ordinal regression. In *Proceedings of the International Conference on Machine Learning*, 2005.

[28] W. Chung, A. Yam, and M. Chan. Networked enterprise: A new business model for global sourcing. *International Journal of Production Economics*, 87(3):267–280, 2004.

[29] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Joural of Machine Learning Research*, 2:265–292, 2001.

[30] Mark W. Craven and Jude W. Shavlik. Extracting tree-structured representations of trained networks. In *Advances in Neural Information Processing Systems*, 1996.

[31] H. L. Davis, G. D. Eppen, and L. Mattsson. Critical factors on worldwide purchasing. *Harvard Business Review*, 52(6):81–90, 1974.

[32] Ofer Dekel, Christopher D. Manning, and Yoram Singer. Log-linear models for label ranking. In *Advances in Neural Information Processing Systems (NIPS)*, 2003.

[33] Marie desJardins, Eric Eaton, and Kiri L. Wagstaff. Learning user preferences for sets of objects. In *Proceedings of the International Conference on Machine Learning*, 2006.

[34] Marie desJardins and Kiri L. Wagstaff. Dd-pref: A language for expressing

preferences over sets. In *Conference of the Association for the Advancement of Artificial Intelligence (AAAI)*, 2005.

[35] Pedro Domingos. Knowledge acquisition from examples via multiple models. In *Proc. 14th International Conference on Machine Learning*, 1997.

[36] C. Domshlak and T. Joachims. Unstructuring user preferences: Efficient non-parametric utility revelation. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2005.

[37] E. Douglas, D. Torres, M. Claudio, and S. Rocco. Extracting trees from trained svm models using a trepan based approach. In *Proceedings of the Fifth International Conference on Hybrid Intelligent Systems*, 2005.

[38] K. Dowsland. *Simulated annealing*. Blackwell Scientific Publications, Oxford, 1993.

[39] Charles Elkan. The foundations of cost-sensitive learning. In *Proceedings of the 17th international joint conference on Artificial intelligence*, 2001.

[40] Usama M. Fayyad and Keki B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of International Joint Conferences on Artificial Intelligence*, 1993.

[41] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

[42] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, 1996.

[43] A. A. Gaballa. Minimum cost allocation of tenders. *Operational Research Quarterly*, 25(3):389–398, 1974.

[44] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[45] Yunsong Guo and Carla Gomes. Learning optimal subsets with implicit user preferences. In *Proceedings of the 21st International Joint Conference on Artifical Intelligence*, 2009.

[46] Yunsong Guo and Carla Gomes. Ranking structured documents: a large margin based approach for patent prior art search. In *Proceedings of the 21st International Joint Conference on Artifical Intelligence*, 2009.

[47] Yunsong Guo, Yanzhi Li, Andrew Lim, and Brian Rodrigues. Tariff concessions in production sourcing. *European Journal of Operational Research*, 187(2), 2008.

[48] Yunsong Guo, Andrew Lim, Brian Rodrigues, and Shanhu Yu. Machine scheduling performance with maintenance and failure. *Mathematical and Computer Modelling*, 45(9-10), 2007.

[49] Yunsong Guo, Andrew Lim, Brian Rodrigues, and Yi Zhu. Carrier assignment models in transportation procurement. *Journal of the Operational Research Society*, 57, 2006.

[50] Yunsong Guo and Bart Selman. Exopaque: A framework to explain opaque machine learning models using inductive logic programming. In *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence*, 2007.

[51] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers*. MIT Press, Cambridge, MA, 2000.

[52] A. Herschtal and B. Raskutti. Optimising area under the ROC curve using gradient descent. In *Proceedings of the International Conference on Machine Learning*, 2004.

[53] William R. Hersh, Chris Buckley, T. J. Leone, and David H. Hickam. Ohsumed: An interactive retrieval evaluation and new large test collection for research. In *Proceedings of the 17th annual international ACM SIGIR Conference*, 1994.

[54] Alfred Horn. On sentences which are true of direct unions of algebras. *Journal of Symbolic Logic*, 16:14–21, 1951.

[55] Nitin Indurkhya and Sholom M. Weiss. Solving regression problems with rule-based ensemble classifiers. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 2001.

[56] Kalervo Järvelin and Jaana Kekäläinen. Ir evaluation methods for retrieving

highly relevant documents. In *Proceedings of the 23rd annual international ACM SIGIR conference*, 2000.

[57] T. Joachims. Optimizing search engines using clickthrough data. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2002.

[58] T. Joachims. A support vector method for multivariate performance measures. In *International Conference on Machine Learning*, 2005.

[59] J. Ju and K. Krishna. Firm behaviours and market access in a free trade area with rules of origin. *National Bureau of Economics Report,Working paper 6857*, 1998.

[60] B. G. Kingsman. Purchasing raw materials with uncertain fluctuating prices. *European Journal of Operational Research*, 25(3):358–372, 1986.

[61] Panos Kouvelis, Meir J. Rosenblatt, and Charles L. Munson. A mathemtatical programming model for global plant location problems: analysis and insights. *IIE Transactions*, 36:127–144, 2004.

[62] Anne O. Krueger. Free trade agreements as protectionist devices: Rules of origin. *National Bureau of Economics Report, working paper*, 2003.

[63] Ryan Lampe. *Strategic Citation.* working paper, 2007.

[64] P. Lattman. Cheapskates. *Forbes*, 176(1):76–77, 2005.

[65] Yanzhi Li, Andrew Lim, and Brian Rodrigues. Global sourcing using local content tariff rules. *IIE Transactions*, 39(5):425–437, 2007.

[66] Douglas Long. *International Logistics: Global Supply Chain Management.* Kluwer Academic Publishers, 2003.

[67] Travis J. Lybbert, Christopher B. Barrett, Solomon Desta, and D. Layne Coppock. Stochastic wealth dynamics and risk management among a poor population. *Economic Journal*, 114(498):750–777, 2004.

[68] Joan Magretta. Fast, global, and entrepreneurial: Supply chain management, Hong Kong style: An interview with Victor Fung. *Harvard Business Review, enhanced edition, online #2020*, Oct. 1st 2002.

[69] KM Mann. Recent developments in government operations and liability - United Mexican States v. Metalclad Corporation: The North American Free Trade Agreement provides powerful private right of action to foreign investors. *Urban Lawyer*, 35(4):697–702, 2003.

[70] Daniel L. McFadden. Econometric analysis of qualitative response models. *Handbook of Econometrics*, pages 1395–1457, 1984.

[71] John G. McPeak and Christopher B. Barrett. Differential risk exposure and stochastic poverty traps among east african pastoralists. *American Journal of Agricultural Economics*, 83(3):674–679, 2001.

[72] Zbigniew Michalewicz and David B. Fogel. *How to Solve It: Modern Heuristics*. Springer-Verlag, 2000.

[73] H. Min and W. P. Galle. International purchasing strategies of multinational U.S. firms. *Journal of Purchasing and Materials Management*, 27(3):9–18, 1991.

[74] S. Minner. Multiple-supplier inventory models in supply chain management: A review. *International Journal of Production Economics*, 81(2):265–279, 2002.

[75] R. M. Monczka and L. C. Giunipero. International purchasing: Charateristics and implementation. *Journal of Purchasing and Materials Management*, 20(4):2–9, 1984.

[76] R. M. Monczka and R. J. Trent. Global sourcing: A development approach. *Journal of Purchasing and Materials Management*, 27(2):2–8, 1991.

[77] D. L. Moore and H. E. Fearon. Computer-assited decision-making in purchasing. *Journal of Purchasing*, 9(4):5–25, 1973.

[78] K. Morik, P. Brockhausen, and T. Joachims. Combining statistical learning with a knowledge-based approach. In *Proceedings of the International Conference on Machine Learning*, 1999.

[79] S. Muggleton. Inverse entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4):245–286, 1995.

[80] Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19/20:629–679, 1994.

[81] Charles L. Munson and Meir J. Rosenblatt. The impact of local content rules on global sourcing decisions. *Production and Operations Management*, 6(3):277–290, 1997.

[82] Ramesh Nallapati. Discriminative models for information retrieval. In *Proceedings of the 27th annual international ACM SIGIR Conference*, 2004.

[83] D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. UCI repository of machine learning databases, http://www.ics.uci.edu/∼mlearn/mlrepository.html, 1998.

[84] Nam Nguyen and Yunsong Guo. Comparisons of sequence labeling algorithms and extensions. In *Proceedings of the 24th International Conference on Machine Learning*, 2007.

[85] Nam Nguyen and Yunsong Guo. Metric learning: A support vector approach. In *Proceedings of the 18th European Conference on Machine Learning and Practice of Knowledge Discovery in Databases*, 2008.

[86] Emanuel Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33:1065–1076, 1962.

[87] M.F. Porter. An algorithm for suffix stripping. In *Program 14(3)*, pages 130–137, 1980.

[88] Bob Price and P. R. Messinger. Optimal recommendation sets: Covering uncertainty over user preferences. In *In Proceedings of the Twentieth National Conference on Artificial Intelligence*, 2005.

[89] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[90] S. Robertson, S. Walker, S. Jones, M.M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3 (1996). In *Text REtrieval Conference*, 1996.

[91] David Ruppert, M.P. Wand, and R.J. Carroll. *Semiparametric Regression*. Cambridge University Press, New York, NY, USA, 2003.

[92] B. Sampat. Examining patent examination: an analysis of examiner and applicant generated prior art. In *National Bureau of Economics, 2004 Summer Institute, Cambridge, MA*, 2004.

[93] Shai Shalev-Shwartz and Yoram Singer. Efficient learning of label ranking by soft projections onto polyhedra. *J. Mach. Learn. Res.*, 7:1567–1599, 2006.

[94] Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th international conference on Machine learning*, 2007.

[95] Gerard Sierksma. *Linear and integer programming:theory and practice*. Marcel Dekker: New York, 2nd edition, 2002.

[96] Martin Smith. Spatial search and fishing location choice: Methodological challenges of empirical modeling. *American Journal of Agricultural Economics*, 82(5):1198–1206, 2000.

[97] S. Stordal. Impacts of the European Economic Area Agreement on the structure and concentration of roundwood sales in norway. *Forest Policy and Economics*, 6(1):49–62, Jan 2004.

[98] Alireza Tamaddoni-Nezhad, Raphael Chaleil, Antonis Kakas, and Stephen Muggleton. Application of abductive ilp to learning metabolic network inhibition from temporal data. *Journal of Machine Learning*, 64:209–230, 2006.

[99] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research (JMLR)*, 6:1453–1484, 2005.

[100] M. Turcotte, S.H. Muggleton, and M.J.E. Sternberg. The effect of relational background knowledge on learning of protein three-dimensional fold signatures. *Journal of Machine Learning*, 1,2:81–96, 2001.

[101] I. Turner. An independent system for the evaluation of contract tenders. *Journal of the Operational Research Society*, 39(6):551–561, 1988.

[102] Vladimir N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

[103] S. K. Vickery. International sourcing: Implications for just-in-time manufacturing. *Production and Inventory Management Journal*, 30(3):66–72, 1989.

[104] C. J. Vidal and M. Goetschalckx. A global supply chain model with transfer pricing and transportation cost allocation. *European Journal of Operational Research*, 129:134–158, 2001.

[105] C. A. Weber and J. R. Current. A multiobjective approach to vendor selection. *European Journal of Operational Research*, 68(2):173–184, 1993.

[106] Sholom M. Weiss and Nitin Indurkhya. Rule-based machine learning methods for functional prediction. *Journal of Artificial Intelligence Research*, 3:383–403, 1995.

[107] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[108] Stefan Wrobel. Scalability issues in inductive logic programming. In *Proceedings of the 9th International Conference on Algorithmic Learning Theory*, 1998.

[109] Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. A support vector method for optimizing average precision. In *Proceedings of the 30th annual international ACM SIGIR conference*, 2007.

[110] Yisong Yue and Thorsten Joachims. Predicting diverse subsets using structural svms. In *International Conference on Machine Learning*, 2008.

[111] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Research and Development in Information Retrieval*, 2001.