Theory and Algorithms for Modern Problems in Machine Learning and an Analysis of Markets

by

Ashish Rastogi

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

Courant Institute of Mathematical Sciences

New York University

May 2008

Richard Cole—Advisor

Mehryar Mohri—Advisor

To the most wonderful parents in the whole world,

Mrs. Asha Rastogi and Mr. Shyam Lal Rastogi

Acknowledgements

First and foremost, I would like to thank my advisors, Professor Richard Cole and Professor Mehryar Mohri, for their unwavering support, guidance and constant encouragement. They have been inspiring mentors and much of what lies in the following pages can be credited to them. Working under their supervision has been one of the most enriching experiences of my life.

I would also like to thank Professor Joel Spencer, Professor Arun Sundararajan, Professor Subhash Khot and Dr. Corinna Cortes for agreeing to serve as members on my thesis committee.

Professor Spencer's class on Random Graphs remains one of the most stimulating courses I undertook as a graduate student. Internships at Google through the summers of 2005, 2006 and 2007 were some of the most enjoyable periods of my graduate school life. Many thanks are due to Dr. Corinna Cortes for providing me with the opportunity to work on several challenging problems at Google. Research initiated during these internships culminated in the development of ideas that form the bulk of this thesis.

I would also like to thank my peers from the graduate school. Spirited discussions in the reading group meetings and during various seminars were directly responsible for the development of a "research attitude", a critical ingredient for the successful completion of graduate studies. For all this, I thank Eugene Weinstein and Tyler Neylon.

Pursuing a Ph.D. can be an arduous affair. It entails everything from having to cope with failure to the exhilarating feeling of settling a research problem successfully. I would like to express my gratitude towards my friends, especially Shruti Haldea, Ritu Gupta, Abhimanyu Yadav, Usha Mallya, Hameer Ruparel, Chris Wu and Marjorie Levy for providing an excellent web of support.

Finally, none of this would ever have transpired without the unselfish love and support of my precious parents and my delightful sister Ragini. Their constant support, patient love and care and unswerving faith has gotten me where I am. This thesis is dedicated to them.

Abstract

The unprecedented growth of the Internet over the past decade and of data collection, more generally, has given rise to vast quantities of digital information, ranging from web documents and images, genomic databases to a vast array of business customer information. Consequently, it is of growing importance to develop tools and models that enable us to better understand this data and to design data-driven algorithms that leverage this information. This thesis provides several fundamental theoretical and algorithmic results for tackling such problems with applications to speech recognition, image processing, natural language processing, computational biology and web-based algorithms.

- Probabilistic automata provide an efficient and compact way to model sequence-oriented data such as speech or web documents. Measuring the similarity of such automata provides a way of comparing the objects they model, and is an essential first step in organizing this type of data. We present algorithmic and hardness results for computing various discrepancies (or dissimilarities) between probabilistic automata, including the relative entropy and the L_p distance; we also give an efficient algorithm to determine if two probabilistic automata are equivalent. In addition, we study the complexity of computing the norms of probabilistic automata.

- Widespread success of search engines and information retrieval systems has led to large scale collection of rating information which is being used

to provide personalized rankings. We examine an alternate formulation of the ranking problem for search engines motivated by the requirement that in addition to accurately predicting pairwise ordering, ranking systems must also preserve the magnitude of the preferences or the difference between ratings. We present algorithms with sound theoretical properties, and verify their efficacy through experiments.

- Organizing and querying large amounts of digitized data such as images and videos is a challenging task because little or no label information is available. This motivates transduction, a setting in which the learning algorithm can leverage unlabeled data during training to improve performance. We present novel error bounds for a family of transductive regression algorithms and validate their usefulness through experiments.

- Finally, price discovery in a market setting can be viewed as an (ongoing) learning problem. Specifically, the problem is to find and maintain a set of prices that balance supply and demand, a core topic in economics. This appears to involve complex implicit and possibly large-scale information transfers. We show that finding equilibrium prices, even approximately, in discrete markets is NP-hard and complement the hardness result with a matching polynomial time approximation algorithm. We also give a new way of measuring the quality of an approximation to equilibrium prices that is based on a natural aggregation of the dissatisfaction of individual market participants.

Contents

Chapter 1

Distances Between Probabilistic Automata

1.1 Introduction

A probabilistic automaton is a finite automaton in which each transition carries a non-negative weight. The weight associated with a path π in the automaton is the product of the weights of the transitions that appear on π and the probability associated with a certain string x accepted by the automaton is the sum of the weights of all the paths on which x is accepted. In addition, the sum of the weights of all strings accepted by the automaton is one. Thus, such an automaton represents a probability distribution over a regular set.

Probabilistic automata were introduced in [81] and are extensively used in a variety of areas of computer science including text and speech process-

ing [73], image processing [39], and computational biology [43]. In natural language-processing applications, for example, probabilistic automata are used to describe morphological and phonological rules [61].

Closely related to probabilistic automata are Hidden Markov Models, which are also used extensively in statistical learning [42, 82]. In a Hidden Markov Model, the weight associated with a transition can itself be a random variable whose value is drawn from some distribution. However, when each transition has a fixed constant weight, a Hidden Markov Model is identical to a probabilistic automaton.

The output of a large-vocabulary speech recognition system or that of a complex information extraction system is often represented as a probabilistic automaton compactly representing a large set of alternative sequences [78, 72]. Natural language sequences such as documents or biological sequences can also be modeled by probabilistic automata [62]. Comparing the objects modeled by these automata is necessary when one wishes, for example, to cluster these objects. Thus, the problem of comparing two probabilistic automata by computing the discrepancy between them is central when one wishes to use such automata for learning. This also also arises in several other machine learning problems. When a probabilistic automaton is obtained as a result of training on a large data set, the quality of the learning algorithm can be measured by computing the distance between the automaton inferred and that of the target automaton. Similarly, in many on-line learning algorithms and grammar inference applications, the convergence of an iterative algorithm relies on the

magnitude of the distance between two consecutive probabilistic automata.

This motivates the design of efficient algorithms for the problem of computing the distance or discrepancy between probabilistic automata. There are many standard distances commonly used to compare distributions, such as the relative entropy or Kullback-Leibler divergence, the L_p distance, the Hellinger distance, the Jensen-Shannon distance, the χ^2-distance, and the Triangle distance between two distributions [98, 38]. In Section 1.2.5, we define each one of these distances formally.

Our focus in this chapter will be on the problem of computing various commonly used distances between two probabilistic automata, including the relative entropy and the L_p distance. We will also consider various distribution-related properties of a single probabilistic automaton (such as the entropy, or the L_p norm). The results of this chapter were published in two journal articles [32, 34]. A high-level summary of our results is as follows.

L_p distance. We give efficient exact and approximate algorithms for computing the L_p distance between two probabilistic automata for even-valued p and prove the problem to be NP-hard for all odd values of p using a reduction from the Max-Clique problem. These latter results complete previously known hardness results given by Lyngsø and Pederson [84], who showed the problem to be NP-hard for L_1 and L_∞. We also establish the hardness of additive approximation of the L_p distance of two probabilistic automata for odd values of p.

Relative Entropy. We give efficient exact and approximate algorithms for computing the relative entropy between two unambiguous probabilistic automata. A finite automaton is said to be *unambiguous* if any string is accepted on at most one path in the automaton. We report the results of experiments demonstrating the practicality of our algorithms for very large probabilistic automata. Finally, we prove that the computation of the relative entropy of arbitrary probabilistic automata is (at least) PSPACE-hard.

We also examine the use of the symmetricized relative entropy in machine learning algorithms and show that, contrary to what is suggested by a number of publications (e.g., [65]), the symmetricized relative entropy is neither positive definite symmetric nor negative definite symmetric, which limits its use and application in kernel methods. In particular, the convergence of training for learning algorithms is not guaranteed when the symmetricized relative entropy is used directly as a kernel, or as the operand of an exponential as in the case of Gaussian Kernels [88].

Equivalence of Probabilistic Automata. A problem closely related to that of computing distances between two probabilistic automata is to test for their equivalence. Our algorithm for computing the L_2 distance of two arbitrary probabilistic automata A_1 and A_2 actually provides a polynomial-time method for testing their equivalence, since A_1 and A_2 are equivalent if and only if their L_2 distance is zero. However, we will describe an even more efficient algorithm based on Schützenberger's standardization technique

[90, 12] with a running-time complexity of $O(|\Sigma|\,(|A_1| + |A_2|)^3)$. This is a significant improvement over the previously best algorithm reported for this problem whose complexity is $O(|\Sigma|\,(|A_1| + |A_2|)^4))$ [99].

The intuition behind the design of our algorithms for the computation of the entropy can be explained in terms of traditional shortest-distance algorithms over weighted graphs. Dijkstra's algorithm, for example, computes the shortest distance from a given source vertex to all the other vertices in the graph. The weight of a path is the *sum* of the weights of the individual edges on the path. Among all candidate paths, we choose the one with the *minimum* weight. Roughly speaking, one can keep track of the entropy instead of the shortest distance by redefining the *sum* and the *minimum* operations appropriately. We formalize this notion using the framework of semirings and generalized shortest-distance algorithms that are introduced in Sections 1.2 and 1.2.3.

In Section 1.2, we introduce some basic algebraic definitions (e.g. semirings) and notation related to probabilistic automata needed for the description of our algorithms. We also introduce the notion of generalized shortest-distances on weighted automata that play a central role in all our algorithms. We conclude Section 1.2 with a set of definitions of the various measures of discrepancy between probability distributions that we study and a high-level summary of the basic questions investigated in this chapter.

In Section 1.2.3, we present efficient algorithms for the computation of the relative entropy between two unambiguous probabilistic automata, along with

similar algorithms for computing the L_p distance for even p and the Hellinger distance between two (not necessarily unambiguous) probabilistic automata. In Section 1.4, we present hardness results for the problem of computing the L_p distance for odd p and the L_∞ distance between probabilistic automata. We also show that computing the relative entropy between arbitrary probabilistic automata is P-SPACE complete.

In Section 1.5, we present an efficient algorithm to determine whether two given probabilistic automata are equivalent, that is, if they assign the same probability to each string x. In Section 1.6, we examine the use of (symmetricized) relative entropy as a kernel in machine learning and present negative results showing that it is neither positive definite, nor negative definite. We conclude the chapter with Section 1.7, in which we study the problem of computing distribution-related measures of a single probabilistic automaton (such as the entropy, or the L_p norm).

1.2 Preliminaries

1.2.1 Semirings

Probabilistic automata are automata in which each transition carries some weight in addition to the usual alphabet symbol [44, 85, 12]. For various operations to be well-defined, the weight set must have the algebraic structure of a semiring [63]. A semiring is a ring that may lack negation.

Definition 1.1 (Monoid) *A* monoid *is a system* $(\mathbb{M}, \otimes, \overline{1})$ *such that:*

- $\otimes$ *is associative: for all* a, b, c *in* $\mathbb{M}$*,*

$$(a \otimes b) \otimes c = a \otimes (b \otimes c).$$

- $\overline{1} \in \mathbb{M}$ *is an identity element for* $\otimes$*: for all* a *in* $\mathbb{M}$*,* $a \otimes \overline{1} = \overline{1} \otimes a = a$*.*

A monoid $(\mathbb{M}, \otimes, \overline{1})$ is commutative if the operation $\otimes$ is commutative: that is, for all a, b in $\mathbb{M}$, $a \otimes b = b \otimes a$.

Definition 1.2 (Semiring) *A* semiring *is a system* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *such that:*

- $(\mathbb{K}, \oplus, \overline{0})$ *is a commutative monoid with* $\overline{0}$ *as the identity element for* $\oplus$*,*
- $(\mathbb{K}, \otimes, \overline{1})$ *is a monoid with* $\overline{1}$ *as the identity element for* $\otimes$*,*
- $\otimes$ *distributes over* $\oplus$*: for all* a, b, c *in* $\mathbb{K}$*,*

$$(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c) \quad \text{and} \quad c \otimes (a \oplus b) = (c \otimes a) \oplus (c \otimes b).$$

- $\overline{0}$ *is an annihilator for* $\otimes$*:* $\forall a \in \mathbb{K}, a \otimes \overline{0} = \overline{0} \otimes a = \overline{0}$*.*

Definition 1.3 (Idempotent Semiring) *A semiring* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *is idempotent if the* $\oplus$ *operation is idempotent. That is, for all* $a \in \mathbb{K}$*,*

$$a \oplus a = a.$$

Some familiar semirings are the Boolean semiring $(\{0,1\}, \vee, \wedge, 0, 1)$ and the tropical semiring $(\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$ related to classical shortest-paths problems and algorithms. A semiring is *commutative* when $\otimes$ is commutative.

Definition 1.4 (Closed Weight) *A weight $a \in \mathbb{K}$ is said to be closed if the infinite sum,*

$$\bigoplus_{n=0}^{\infty} a^n \in \mathbb{K},$$

and if associativity, commutativity, and distributivity apply to countable sums.

Definition 1.5 (Closed Semiring [76]) *A semiring $\mathbb{K}$ is said to be closed if all its elements a are closed.*

Definition 1.6 (k-closed Semiring [76]) *A semiring $\mathbb{K}$ is said to be k-closed if for all $a \in \mathbb{K}$,*

$$\bigoplus_{n=0}^{k+1} a^n = \bigoplus_{n=0}^{k} a^n.$$

More generally, we will say that $\mathbb{K}$ is *closed* (*k-closed) for an automaton A*, if the closedness (respectively k-closedness) axioms hold for all finite sums of cycle weights at any state of A [76]. In some semirings, e.g., the probability semiring $(\mathbb{R}_+, +, \times, 0, 1)$, the equality $\bigoplus_{n=0}^{k+1} a^n = \bigoplus_{n=0}^{k} a^n$ may hold for the cycle weights of A only approximately, modulo $\epsilon > 0$, as defined next.

Definition 1.7 (ϵ-k-closed Semiring for Automaton A under Metric d) *A semiring $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ is said to be ϵ-k-closed for an automaton A under*

metric $d : \mathbb{K} \times \mathbb{K} \mapsto \mathbb{R}$ *if for all finite sums* a *of cycle weights at any state in* A,

$$d\left(\bigoplus_{n=0}^{k+1} a^n, \bigoplus_{n=0}^{k} a^n\right) \leq \epsilon.$$

Comment 1.1 *All of the automata considered in this chapter are probabilistic, defined over the probability semiring* $(\mathbb{R}_+, +, \times, 0, 1)$. *Henceforth, we will always use the* L_1 *metric for the notion of* ϵ*-*k*-closedness.*

1.2.2 Probabilistic Automata and Shortest-Distances

Definition 1.8 (Weighted Automaton) A weighted automaton A *is a 7-tuple* $(\Sigma, Q, I, F, E, \lambda, \rho)$ *over a semiring* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$*:*

- Σ *is the finite alphabet of the automaton,*
- Q *is a finite set of states,*
- $I \subseteq Q$ *is the set of initial states,*
- $F \subseteq Q$ *is the set of final states,*
- $E \subseteq Q \times \Sigma \cup \{\epsilon\} \times \mathbb{K} \times Q$ *is a finite set of transitions,*
- $\lambda : I \to \mathbb{K}$ *is the initial weight function mapping* I *to* $\mathbb{K}$*, and*
- $\rho : F \to \mathbb{K}$ *is the final weight function mapping* F *to* $\mathbb{K}$*.*

The weighted automata considered in this chapter are assumed not to contain ϵ-transitions. A pre-processing ϵ-removal algorithm can be used to

remove such transitions for the automata considered here [75]. In the absence of ϵ-cycles, the complexity of that algorithm is $O(|Q|^2 + |Q||E|)$ [75].

We denote by $|A| = |E| + |Q|$ the size of an automaton A, that is the sum of the number of states and transitions of A. Given a transition $e \in E$, we denote by $i[e]$ its input label, $p[e]$ its origin or previous state and $n[e]$ its destination state or next state, $w[e]$ its weight. Given a state $q \in Q$, we denote by $E[q]$ the set of transitions leaving q.

A *path* $\pi = e_1 \cdots e_k$ in A is an element of E^* with consecutive transitions: $n[e_{i-1}] = p[e_i]$, $i = 2, \ldots, k$. We extend n and p to paths by setting: $n[\pi] = n[e_k]$ and $p[\pi] = p[e_1]$. A *cycle* is a path with the same origin and destination states. The labeling functions i and the weight function w can also be extended to paths by defining the label of a path as the concatenation of the labels of its constituent transitions, and the weight of a path as the $\otimes$-product of the weights of its constituent transitions: $i[\pi] = i[e_1] \cdots i[e_k]$, $w[\pi] = w[e_1] \otimes \cdots \otimes w[e_k]$. We denote by $P(q, q')$ the set of paths from q to q' and by $P(q, x, q')$ the set of paths from q to q' with input label $x \in \Sigma^*$.

The output weight associated by an automaton A to an input string $x \in \Sigma^*$ is defined by:

$$[\![A]\!](x) = \bigoplus_{\pi \in P(I,x,F)} \lambda[p[\pi]] \otimes w[\pi] \otimes \rho[n[\pi]].$$

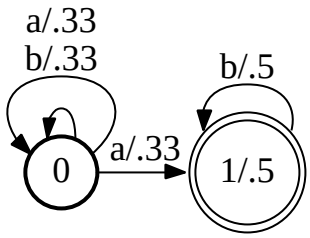


Figure 1.1: An unambiguous weighted finite automaton that admits no equivalent deterministic weighted automaton. 0 is the initial state and 1 the final state. The automaton accepts the set of strings $(a^*b^*)^*ab^*$.

The language accepted by A, denoted by $L(A)$, is defined by:

$$L(A) = \{x : P(I, x, F) \neq \emptyset\}.$$

Definition 1.9 (Trim Automaton) *A state of an automaton A is* accessible *if it can be reached from an initial state. It is said to be* co-accessible *if it lies on a path reaching a final state. An automaton is said to be* trim *if all of its states are both accessible and co-accessible and it admits no zero-weight transitions.*

Without loss of generality, we assume that the automata considered in this chapter are trim.

A weighted automaton A is said to be *deterministic* or *subsequential* if it has a deterministic input, that is if it has a unique initial state and if no two transitions leaving the same state share the same input label.

Definition 1.10 (Unambiguous Weighted Automata) *A weighted automa-*

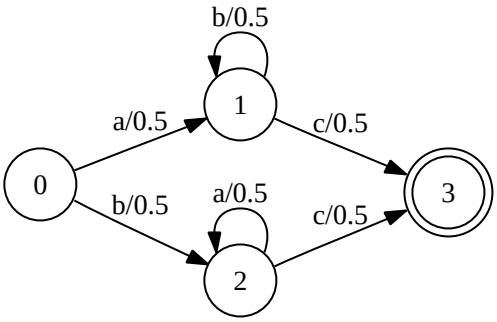


Figure 1.2: An example of a probabilistic automaton. The automaton accepts the set of string $ab^*c \cup ba^*c$. The probability that the automaton accepts the string $abbbc$ is given by $0.5 \times (0.5)^3 \times 0.5 = (0.5)^5$.

ton A is said to be unambiguous *if for any $x \in \Sigma^*$ there is at most one accepting path labeled with x in A.*

Thus, the class of unambiguous weighted automata includes *deterministic* weighted automata.

Definition 1.11 (Probabilistic Automaton) *A weighted automaton A defined over the probability semiring $(\mathbb{R}_+, +, \times, 0, 1)$ is said to be* probabilistic *if the weights it associates to the strings in Σ^* corresponds to a probability distribution. That is, it verifies:*

$$\sum_{x \in \Sigma^*} [\![A]\!](x) = 1 \qquad \textit{and} \qquad \forall x, [\![A]\!](x) \geq 0.$$

Definition 1.12 (Stochastic Automaton) *A probabilistic automaton A is said to be* stochastic *if at each state the weights of the outgoing transitions and the final weight sum to one.*

Observe that our definition of probabilistic automata differs from those of [81] and [80]. *Probabilistic automata* as defined by these authors are weighted automata over $(\mathbb{R}_+, +, \times, 0, 1)$ such that at any state q and for any label $a \in \Sigma$, the weights of the outgoing transitions of q labeled with a sum to one. More generally, with that definition, the weights of the paths leaving state q and labeled with $x \in \Sigma^*$ sum to one. Such automata define a conditional probability distribution $\Pr[q' \mid q, x]$ over all states q' that can be reached from q by reading x.

Instead, with our definition, probabilistic automata represent distributions over Σ^*, $\Pr[x], x \in \Sigma^*$. These are the natural distributions that arise in many applications. They can be inferred from large data sets using statistical learning techniques. We are interested in computing various distances between two such distributions over strings. As we shall see, the problem of computing these distances can be formulated computing "shortest-distances" over an appropriate semiring on a suitably defined automaton. Next, we introduce the notion of shortest-distances and in Section 1.2.3, we discuss both exact and approximate algorithms for their computation, provided the underlying semiring satisfies the technical condition of closedness, specified precisely later.

Shortest-Distances

When the sum of the weights of all paths from any state p to any state q is well-defined and in $\mathbb{K}$, we can define the "shortest-distance" from $p \in Q$ to

$q \in Q$ as:

$$d[p, q] = \bigoplus_{\pi \in P(p,q)} w[\pi], \tag{1.1}$$

where the summation is defined to be $\overline{0}$ when $P(p, q) = \emptyset$.

Let $s[A]$ denote the $\oplus$-sum of the weights of all successful paths of A when it is defined and in $\mathbb{K}$. This sum in an automaton with initial state q_s and the set of final states F is given by:

$$s[A] = \bigoplus_{q_f \in F} d[q_s, q_f]. \tag{1.2}$$

$s[A]$ can be viewed as the shortest-distance from the initial states to the final states. When $\oplus$ is replaced by min and $\otimes$ by $+$, this definition coincides with the classical definition of shortest-distance in the tropical semiring. This motivates the terminology we use.

1.2.3 Algorithms for Computing Shortest-Distances

In this section, we review two algorithms for computing the generalized shortest-distances (as defined in Section 1.2.2) between states in a weighted automaton. The exact algorithm is a generalization of the classical Floyd-Warshall algorithm and computes shortest-distances between all pairs of states, provided the underlying semiring is closed. The approximate algorithm, on the other hand, is a generalization of the single-source shortest-distance algorithm and computes the shortest-distance approximately as long as the underlying semiring

is ϵ-k-closed.

Exact Algorithm

A generalization of the classical Floyd-Warshall algorithm can be used to compute all-pairs shortest distances $d[p, q]$ $(p, q \in Q)$ over a *closed semiring* which need not be idempotent [74, 76] (Definition 1.3). Thus, this algorithm can also be used to compute the shortest-distance $s[A]$ (Equation 1.2) for a weighted automaton A over a non-idempotent semiring, as needed for our purpose.

Note that our definition of closed semirings (Definition 1.5) [64] is more general than the classical one used by Cormen *et al.* [26] as our definition does not assume idempotence. This is because idempotence is not necessary for the proof of the correctness of the generic all-pairs shortest-distance algorithms of Floyd-Warshall and Gauss-Jordan [74, 76].

The following is the pseudocode for the generic Floyd-Warshall algorithm admits. It also admits an in-place implementation as described in [74].

1 **for** $i \leftarrow 1$ **to** $|Q|$
2 **do for** $j \leftarrow 1$ **to** $|Q|$
3 **do** $d[i, j] \leftarrow \bigoplus_{e \in P(i,j)} w[e]$
4 **for** $k \leftarrow 1$ **to** $|Q|$
5 create a copy of matrix d in d'
6 **do for** $i \leftarrow 1$ **to** $|Q|$
7 **do for** $j \leftarrow 1$ **to** $|Q|$
8 **do** $d[i, j] \leftarrow d'[i, j] \oplus (d'[i, k] \otimes d'[k, k]^* \otimes d'[k, j])$

9 **return** d

In all of the semirings we study in this chapter, the $\oplus$, $\otimes$ and closure operations can be carried out in constant time. Thus, the running time complexity of the algorithm is $\Theta(|E| + |Q|^3)$ and its space complexity is $\Omega(|Q|^2)$ when applied to a weighted automaton $A = (Q, I, F, \Sigma, \delta, \sigma, \lambda, \rho)$ over a closed semiring.

The cubic time complexity, together with the quadratic space complexity of this algorithm is unappealing and makes the algorithm especially inapplicable to large automata. In text and speech processing applications, a weighted automaton may have several hundred million states and transitions. In such a case, the algorithm would require maintaining a matrix with ten billion entries.

The next section presents an algorithm that exploits the sparseness of the graph and does not impose as stringent space requirements.

Approximate Algorithm

A generic single-source shortest-distance algorithm for directed graphs defined over a *k-closed* semiring was presented in [76]. The algorithm can be viewed as a generalization to these semirings of classical shortest-paths algorithms. This generalization is not trivial and does not require the semiring to be idempotent. The algorithm is also generic in the sense that it works with any queue discipline.

The approximate version of the generic single-source shortest distance algorithm relies on the ϵ-k-closedness of the underlying semiring (see Defini-

tion 1.7), where the equality test is replaced by an ϵ-equality: $u =_\epsilon v$ if $||u - v||_\infty \leq \epsilon$. Note that under the probability semiring $(\mathbb{R}_+, +, \times, 0, 1)$ (a natural semiring for probabilistic automata), cycle weights from any state in a trim automaton are strictly less than one (otherwise, the sum of the weights of all successful paths would be divergent, and therefore not sum to one). This implies that the closure operation for cycle weights $0 \leq w < 1$ is well defined, and in $(\mathbb{R}_+, +, \times, 0, 1)$:

$$\lim_{k \to \infty} \bigoplus_{i=0}^{k} w^i = \frac{1}{1-w}.$$

Because of this property, a probabilistic automaton defined over the probability semiring $(\mathbb{R}_+, +, \times, 0, 1)$, with maximum cycle weight W is ϵ-k-closed (recall Definition 1.7) when

$$\left| \sum_{n=0}^{k+1} W^n - \sum_{n=0}^{k} W^n \right| \leq \epsilon.$$

Upon simplification, this yields the condition $k \geq \frac{\log(1/\epsilon)}{\log(1/W)} - 1$. Clearly, $k \geq \frac{\log(1/\epsilon)}{\log(1/W)}$ suffices.

The following gives the pseudocode of the modified algorithm.

1 **for** $i \leftarrow 1$ **to** $|Q|$
2 **do** $d[i] \leftarrow r[i] \leftarrow \overline{0}$
3 $d[s] \leftarrow r[s] \leftarrow \overline{1}$
4 $S \leftarrow \{s\}$
5 **while** $S \neq \emptyset$
6 **do** $q \leftarrow head(S)$

```
7        DEQUEUE(S)
8        r' ← r[q]
9        r[q] ← 0̄
10       for each e ∈ E[q]
11       do if d[n[e]] ≠_ε d[n[e]] ⊕ (r' ⊗ w[e])
12             then d[n[e]] ← d[n[e]] ⊕ (r' ⊗ w[e])
13                  r[n[e]] ← r[n[e]] ⊕ (r' ⊗ w[e])
14                  if n[e] ∉ S
15                    then ENQUEUE(S, n[e])
```

$d[q]$ denotes the tentative shortest distance from the source s to q. $r[q]$ keeps track of the sum of the weights added to $d[q]$ since the last queue extraction of q from the queue. The attribute r is needed for the shortest-distance algorithm to work in non-idempotent cases. The algorithm uses a queue S to store the set of states to consider for the relaxation steps of lines 11-15 [76]. Any queue discipline, e.g., FIFO, shortest-first, topological (in the acyclic case), can be used. The test of line 11 is based on an ϵ-equality.

Different queue disciplines yield different running times for our algorithm. The choice of the best queue discipline to use can be based on the structure of the automaton, which can be exploited to obtain a more efficient algorithm.

Let $N(q)$ denote the number of times a state q is inserted in the queue. Then, using the Fibonacci heap with a shortest first queue discipline (as in

Dijkstra's algorithm), the complexity of the algorithm is given by:

$$O(|Q| + |E| \max_{q \in Q} N(q) + \log |Q| \sum_{q \in Q} N(q)). \tag{1.3}$$

If the underlying automaton is acyclic, then using the queue discipline corresponding to the topological order yields the best time complexity, and the problem can be solved in linear time: $O(|Q| + |E|)$.

Using a breadth-first queue discipline (as in the Bellman-Ford shortest distance algorithm), updates to the shortest distance estimates in iteration k can be formulated as $D^k = MD^{k-1}$, where M is the *matrix associated with the automaton*, that is the matrix representing the weighted graph defined by the automaton. Note that the matrix multiplication here is over the $\oplus$ and $\otimes$ operations of the semiring, so that $D^k[i] = \oplus_{j=1}^{|Q|} M[i,j] \otimes D^{k-1}[j]$.

1.2.4 Composition of Weighted Automata

In many cases, including the problem of computing the relative entropy of two probabilistic automata, we will need to carry out the composition of two probabilistic automata under an appropriate semiring. We review the composition principle here.

Let A_1 and A_2 be two weighted automata over the same semiring, with $A_i = (\Sigma, Q_i, I_i, F_i, E_i, \lambda_i, \rho_i)$ for $i \in \{1, 2\}$. The intersection A of A_1 and A_2 is denoted by $A = A_1 \cap A_2$. It is a weighted automaton accepting the language

$L(A_1) \cap L(A_2)$ and defined by the tuple A:

$$A = (\Sigma, Q_1 \times Q_2, I_1 \times I_2, F_1 \times F_2, E, (\lambda_1, \lambda_2), (\rho_1, \rho_2)),$$

where the transitions E are defined according to the following rule:

$$(q_1, a, w_1, q_2) \in E_1 \text{ and } (q_1', a, w_1', q_2') \in E_2 \Rightarrow ((q_1, q_1'), a, (w_1 \otimes w_1'), (q_2, q_2')) \in E.$$

There exists a general algorithm for the computation of the intersection over an arbitrary semiring, even in the presence of ϵ-transitions [77]. The time complexity of the algorithm is quadratic $O(|A_1||A_2|)$ since in the worst case the outgoing transitions of each state of A_1 match all those of each state of A_2.

1.2.5 Distances Between Distributions And Norms

There are many standard distances or discrepancies used to compare distributions which can also serve to compare probabilistic automata. Some of the most commonly used ones are: the *relative entropy* or *Kullback-Leibler divergence* D, the L_p *distance*, the *Hellinger distance*, the *Jensen-Shannon distance* JS, the χ^2*-distance*, and the *triangle distance* Δ. For distributions q_1 and q_2

over a discrete set $\mathcal{X}$, their definitions are as follows:

$$\begin{aligned}
D(q_1\|q_2) &= \sum_{x\in\mathcal{X}} q_1(x)\log\frac{q_1(x)}{q_2(x)} \\
L_p(q_1,q_2) &= \Big(\sum_{x\in\mathcal{X}} |q_1(x)-q_2(x)|^p\Big)^{1/p} \\
L_\infty(q_1,q_2) &= \max_{x\in\mathcal{X}} |q_1(x)-q_2(x)| \\
\text{Hellinger}(q_1,q_2) &= \Big(\sum_{x\in\mathcal{X}} \Big(\sqrt{q_1(x)}-\sqrt{q_2(x)}\Big)^2\Big)^{1/2} \\
\text{JS}(q_1,q_2) &= \sum_{x\in\mathcal{X}} \left(q_1(x)\log\frac{2q_1(x)}{q_1(x)+q_2(x)} + q_2(x)\log\frac{2q_2(x)}{q_1(x)+q_2(x)}\right) \\
\chi^2(q_1,q_2) &= \sum_{x\in\mathcal{X}} \frac{(q_1(x)-q_2(x))^2}{q_2(x)} \\
\Delta(q_1,q_2) &= \sum_{x\in\mathcal{X}} \frac{(q_1(x)-q_2(x))^2}{q_2(x)+q_2(x)}.
\end{aligned} \tag{1.4}$$

In this chapter, we will also be interested in computing the entropy of the L_p norm of a single probability distribution. For a distribution q over a discrete set $\mathcal{X}$, its entropy $H(q)$ and L_p norm $L_p(q)$ is defined as:

$$\begin{aligned}
H(q) &= -\sum_{x\in\mathcal{X}} q(x)\log q(x) \\
L_p(q) &= \left[\sum_{x\in\mathcal{X}} [q(x)]^p\right]^{1/p}
\end{aligned} \tag{1.5}$$

Several general inequalities relate these distances [98, 38] including the following ones (the last one holds when the set $\mathcal{X}$ is finite and of size n):

$$[L_1(q_1,q_2)]^2/2 \le D(q_1\|q_2)$$

$$\text{Hellinger}(q_1, q_2) \leq \Delta(q_1, q_2) \leq 2 \cdot \text{JS}(q_1, q_2)$$

$$\frac{L_2(q_1, q_2)}{L_\infty(q_1) + L_\infty(q_2)} \leq \Delta(q_1, q_2) \leq L_1(q_1, q_2) \leq \sqrt{n}\, L_2(q_1, q_2).$$

Now that we have presented the basic concepts of semirings, probabilistic automata and provided the definitions of the various distance-measures between probability distributions, we are in a position to state precisely the problems we study in this chapter.

1.2.6 Problems Studied

Problem 1.1 *Given two probabilistic automata A_1 and A_2, what is the complexity of computing various information theoretic distances between the distributions represented by A_1 and A_2, including relative entropy, L_p distance and Hellinger distance?*

Problem 1.2 *Given a probabilistic automaton A, what is the complexity of computing its entropy and its L_p norm?*

Problem 1.3 *Given two probabilistic automata A_1 and A_2, what is the complexity of deciding whether or not they are equivalent? That is, whether for all $x \in \Sigma^*$, $[\![A_1]\!](x) = [\![A_2]\!](x)$.*

1.3 Algorithms for Computation of Distances

In this section, we review the computation of various information-theoretic distances between automata. We begin with the problem of computing the rel-

ative entropy between two unambiguous probabilistic automata. We then generalize the semiring-based shortest-distance formulation, and apply the ideas used in the computation of relative entropy to the problem of computing various other distances including the L_p distance for even p and the Hellinger distance.

1.3.1 Relative Entropy of Unambiguous Automata

The relative entropy $D(q_1 \| q_2)$ (Equation 1.4), or Kullback-Leibler divergence, is one of the most commonly used measures of the discrepancy of two distributions p and q [36]. It is an asymmetric difference that admits the following information-theoretical interpretation: it measures the number of additional bits needed to encode distribution p when using an optimal code for q in place of an optimal code for p. It is always non-negative, and is zero if and only if the two distributions q_1 and q_2 are identical. It also does not obey the triangle inequality, it is not symmetric, and is therefore not a metric.

One approximate solution for the computation of the relative entropy would consist of sampling sequences from the distributions represented by each of the automata and of using those to compute the KL-divergence by simply summing their contributions. But, sample sizes guaranteeing a small approximation error could be very large, with prohibitive computational cost, while still providing only an approximate solution.

A procedure for the approximate computation of the relative entropy was given by Calera-Rubio and Carrasco [16]. The procedure applies to determin-

istic weighted automata and cannot be generalized to the case of unambiguous weighted automata because it is based on a specific sum decomposition (the partitioning assumed in [16] [Equations 15 and 16, page 6], that does not hold for unambiguous automata).

Our algorithms apply to the larger class of unambiguous weighted automata. For some unambiguous weighted automata, the size of any equivalent deterministic weighted automaton is exponentially larger (Figure 1.1). Since the size of the machine directly affects the complexity of the computation, it is important to be able to compute the entropy directly from the unambiguous automaton. To the best of our knowledge, the algorithm presented in this chapter is the first algorithm for the *exact* computation of the relative entropy. We also describe an approximate algorithm that is conceptually simpler than the procedure of [16] and has a better time and space complexity.

In this section, we introduce the *entropy semiring*, which helps formulate the computation of the relative entropy of unambiguous probabilistic automata as a shortest-distance problem. We then use the exact and approximate algorithms for the computation of shortest-distances already described in Section 1.2.3 to compute the relative entropy of unambiguous probabilistic automata.

The relative entropy between two probabilistic automata A and B can be written as the sum of two terms:

$$D(A_1 \| A_2) = \sum_x [\![A_1]\!](x) \log [\![A_1]\!](x) - \sum_x [\![A_1]\!](x) \log [\![A_2]\!](x). \tag{1.6}$$

Note that the first term is simply $-H(A_1)$, where $H(A_1)$ is the entropy of A_1.

Entropy Semiring

This section introduces a semiring that will be used later to formulate the problem of computing the relative entropy of two unambiguous automata as a single-source shortest-distance problem.

Let $\mathbb{K}$ denote $(\mathbb{R} \cup \{+\infty, -\infty\}) \times (\mathbb{R} \cup \{+\infty, -\infty\})$. For pairs (x_1, y_1) and (x_2, y_2) in $\mathbb{K}$, define the following :

$$(x_1, y_1) \oplus (x_2, y_2) = (x_1 + x_2, y_1 + y_2) \tag{1.7}$$

$$(x_1, y_1) \otimes (x_2, y_2) = (x_1 x_2, x_1 y_2 + x_2 y_1) \tag{1.8}$$

Lemma 1.1 *The system* $(\mathbb{K}, \oplus, \otimes, (0,0), (1,0))$ *defines a commutative semiring.*

Proof. It is known that $(\mathbb{K}, \oplus, (0,0))$ is a commutative monoid with $(0,0)$ as the identity element for $\oplus$. Furthermore, it is clear that $(\mathbb{K}, \otimes, (1,0))$ is a commutative monoid with $(1,0)$ as the identity element for $\otimes$. Also, $(0,0)$ is an annihilator for $\otimes$. Thus, all that remains to be shown is that $\otimes$ distributes over $\oplus$. Since both operations are commutative, we need to verify that for all $z_1, z_2, z_3 \in \mathbb{K}$,

$$(z_1 \oplus z_2) \otimes z_3 = (z_1 \otimes z_3) \oplus (z_2 \otimes z_3)$$

Let $z_i = (x_i, y_i)$ for $i \in \{1, 2, 3\}$. Consider $(z_1 \oplus z_2) \otimes z_3$. We have

$$\begin{aligned}
(z_1 \oplus z_2) \otimes z_3 &= ((x_1, y_1) \oplus (x_2, y_2)) \otimes (x_3, y_3) \\
&= (x_1 + x_2, y_1 + y_2) \otimes (x_3, y_3) \\
&= ((x_1 + x_2)x_3, (x_1 + x_2)y_3 + x_3(y_1 + y_2)) \\
&= (x_1x_3, x_1y_3 + x_3y_1) \oplus (x_2x_3, x_2y_3 + x_3y_2) \\
&= ((x_1, y_1) \otimes (x_3, y_3)) \oplus ((x_2, y_2) \otimes (x_3, y_3)) \\
&= (z_1 \otimes z_3) \oplus (z_2 \otimes z_3).
\end{aligned}$$

□

This definition of the entropy semiring is motivated by the following observation:

Observation 1.1 *Let A be an unambiguous probabilistic automaton under the probability semiring $(\mathbb{R}_+, +, \times, 0, 1)$. Let H_A denote the automaton obtained from A by replacing the weight $w[e]$ of each transition by the pair $(w[e], w[e] \log w[e])$ over the entropy semiring. Then, for all strings x,*

$$[\![H_A]\!](x) = ([\![A]\!](x), [\![A]\!](x) \log [\![A]\!](x)).$$

Note that the second term in the pair can be thought of as the negation of the contribution of x to the entropy of automaton A.

Comment 1.2 *Observation 1.1 reveals why the shortest-distance based approach with the entropy semiring only works when the underlying automata are unambiguous. Indeed, if a string x is accepted on two paths π_1 and π_2 with weights (probabilities) w_1 and w_2, then:*

$$[\![H_A]\!](x) = (w_1 + w_2, w_1 \log w_1 + w_2 \log w_2).$$

Clearly, the second term above is not *the negation of the contribution of x to the entropy, which is $-(w_1 + w_2) \log(w_1 + w_2)$.*

The generic Floyd-Warshall algorithm can be applied to any automaton A for which the semiring considered is closed. Recall that a semiring is closed for A if the infinite $\oplus$-sums for all cycle weights are well-defined and in $\mathbb{R}$. The following lemma shows that the entropy semiring has the desired property.

Lemma 1.2 *Let A be a weighted automaton over the entropy semiring such that for any cycle weight $w = (x, y)$, x less than one $(0 \leq x < 1)$. Then, the entropy semiring is closed for A.*

Proof. For any $(x, y) \in \mathbb{K}$ and $k \geq 0$, define R_k as:

$$R_k = \overbrace{(x, y) \otimes \ldots \otimes (x, y)}^{k \text{ times}},$$

with $R_0 = (1, 0)$. It is straightforward to show by induction that $R_k = (x^k, kyx^{k-1}) = (x^k, y\frac{d(x^k)}{dx})$.

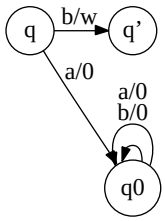


Figure 1.3: Illustration of the completion operation.

For $N \geq 0$, define S_N by:

$$S_N = \bigoplus_{i=0}^{N} R_i = \left(\frac{1 - x^{N+1}}{1 - x}, y \cdot \left[\frac{1 - x^N}{(1-x)^2} - \frac{N x^N}{1 - x} \right] \right). \tag{1.9}$$

Thus, for $0 \leq x < 1$, the closure of (x, y) is well-defined and in $\mathbb{K}$:[1]

$$(x, y)^* = \lim_{N \to \infty} S_N = \left(\frac{1}{1-x}, \frac{y}{(1-x)^2} \right) = \left(\frac{1}{1-x}, y \frac{d}{dx} \left(\frac{1}{1-x} \right) \right).$$

□

Semiring Formulation for Computing Relative Entropy

Suppose we wish the compute the relative entropy between two unambiguous probabilistic automata A_1 and A_2. A_1 and A_2 are not necessarily *complete*: at some states, there may be no outgoing transition labeled with a given element of the alphabet $a \in \Sigma$. We can however make them complete in a way similar to the standard construction in the unweighted case. We introduce a new state q_0 with final weight 0, add self-loops with weight 0 at that state labeled with

[1]The right-hand side can also be written as: $(x^*, y(x^*)^2)$, if we denote by $x^* = \sum_{n=0}^{\infty} x^n$.

all elements of the alphabet, and for any $a \in \Sigma$ and $q \in Q$, add a transition from state q to q_0 labeled with a with weight 0 when q does not have an outgoing transition labeled with a (see Figure 1.3). This construction leads to a complete and unambiguous weighted automaton equivalent to the original one since the transitions added all have weight 0. The completion operation is only applied to handle the boundary case when there exists a string $x \in \Sigma^*$ such that $[\![A_2]\!](x) = 0$ and $[\![A_1]\!](x) \neq 0$. In this case, the completion operation ensures that the future computation of the relative entropy would correctly lead to ∞. Note that the completion operation can be done on-demand. States and transitions can be created only when needed for the application of other operations. Thus, we can assume that A_1 and A_2 are unambiguous and complete. At the cost of introducing a super-initial and a super-final state, we can also assume in the following, without loss of generality, that the initial weight λ and the final weights $\rho(q)$ are all equal to 1 in A_1 and A_2.

Let $\log A$ denote the weighted automaton derived from A by replacing each weight $w \in \mathbb{R}_+$ by $\log w$ and let $\Phi_1(A)$ ($\Phi_2(A)$) denote the weighted automaton over the entropy semiring derived from A by replacing each weight w by the pair $(w, 0)$ (resp. $(1, w)$). The construction of $\log A$, $\Phi_1(A)$, and $\Phi_2(A)$ from A is straightforward and can be done in linear time.

For the following lemma, recall that $s[A]$ is the generalized shortest-distance in A, as defined in Equation 1.2.

Lemma 1.3 *The relative entropy of A_1 and A_2 satisfies the following identity*

in the entropy semiring:

$$(0, D(A_1 \| A_2)) = s[\Phi_1(A_1) \cap \Phi_2(\log A_1)] - s[\Phi_1(A_1) \cap \Phi_2(\log A_2)].$$

Thus, the relative entropy is expressed in terms of single-source shortest-distance computations over the entropy semiring.

Proof. Since A_1 is unambiguous and complete, both $\Phi_1(A_1)$ and $\Phi_2(\log A_1)$ are also unambiguous and complete. Thus, for a given string x, there is at most one accepting path in $\Phi_1(A_1)$ or $\Phi_2(\log A_1)$ labeled with x. Then, by definition of intersection, the weight associated by $\Phi_1(A_1) \cap \Phi_2(\log A_1)$ to a string x is

$$([\![A_1]\!](x), 0) \otimes (1, \log[\![A_1]\!](x)) = ([\![A_1]\!](x), [\![A_1]\!](x) \log[\![A_1]\!](x)). \tag{1.10}$$

Thus, the shortest-distance from the initial states to the final states in $\Phi_1(A_1) \cap \Phi_2(\log A_1)$ is

$$\begin{aligned} s[\Phi_1(A_1) \cap \Phi_2(\log A_1)] &= \bigoplus_x ([\![A_1]\!](x), [\![A_1]\!](x) \log[\![A_1]\!](x)) && (1.11) \\ &= (\sum_x [\![A_1]\!](x), \sum_x [\![A_1]\!](x) \log[\![A_1]\!](x)) && (1.12) \\ &= (1, \sum_x [\![A_1]\!](x) \log[\![A_1]\!](x)). && (1.13) \end{aligned}$$

Similarly, we can show that[2]

$$s[\Phi_1(A_1) \cap \Phi_2(\log A_2)] = (1, \sum_x [\![A_1]\!](x) \log [\![A_2]\!](x)). \tag{1.14}$$

The statement of the lemma follows directly from the identities 1.13 and 1.14 and Equation 1.6. □

Thus, the computation of the relative entropy is reduced to two single-source shortest-distance computations over the entropy semiring. The next section applies the shortest-distance algorithms reviewed in Section 1.2.3 to compute these two quantities in the sum in Equation 1.14. Since the first term yields the entropy of a single unambiguous probabilistic automaton, our results clearly also apply to the computation of the entropy.

Exact Computation

Theorem 1.1 *The relative entropy of two unambiguous probabilistic automata A_1 and A_2 can be computed exactly in time $\Theta(|A_1 \cap A_2|^3)$ and space $\Theta(|A_1 \cap A_2|^2)$.*

Proof. Since the probabilistic automaton A_1 is assumed to be trim and complete (Definition 1.9 and Figure 1.3), the weight u of any cycle must satisfy $0 \leq u < 1$, otherwise the automaton would not be closed. The weight of a cycle of $\Phi_1(A_1) \cap \Phi_2(\log A_1)$ is of the form $(u, u \log u)$ (see Equation 1.10),

[2]Given a string $x = x_1 x_2$ whose respective transitions have weights u_1 and u_2 in A and v_1 and v_2 in B, the weight in $\Phi_1(A_1) \cap \Phi_2(\log A_2)$ becomes $(u_1, u_1 \log v_1) \otimes (u_2, u_2 \log v_2) = (u_1 u_2, u_1 u_2 \log(v_1 v_2))$, that is $(\|A_1\|(x_1 x_2), \|A_1\|(x_1 x_2) \log \|A_2\|(x_1 x_2))$.

where u is the weight of the cycle of A_1, and similarly, the weight of a cycle of $\Phi_1(A_1) \cap \Phi_2(\log A_2)$ is of the form $(u, u \log v)$, where v is the weight of a matching cycle in A_2.

Thus, the entropy semiring is closed both for $\Phi_1(A_1) \cap \Phi_2(\log A_2)$ and $\Phi_1(A_1) \cap \Phi_2(\log A_1)$ and the generic Floyd-Warshall algorithm can be applied to compute the shortest-distances $s[\Phi_1(A_1) \cap \Phi_2(\log A_2)]$ and $s[\Phi_1(A_1) \cap \Phi_2(\log A_1)]$.

The intersection $\Phi_1(A_1) \cap \Phi_2(\log A_1)$ can be computed in linear time $O(|A_1|)$ but the worst cost of computation of $\Phi_1(A_1) \cap \Phi_2(\log A_2)$ is quadratic, $O(|A_1||A_2|)$. The total time complexity of the computation of the relative entropy is thus $\Theta(|A_1 \cap A_2|^3)$. Its space complexity is $\Theta(|A_1 \cap A_2|^2)$. $\square$

This provides an exact algorithm for the computation of the relative entropy. The cubic time complexity of the algorithm with respect to the size of the intersection automaton makes it rather slow for large automata.

Its quadratic lower bound complexity with respect to the size of the intersection machine makes it prohibitive for use in many applications. In text and speech processing applications, a weighted automaton may have several hundred million states and transitions. Even, if A_1 has only about 100,000 states and $A_1 \cap A_2$ has about the same number of states, the algorithm requires maintaining a matrix d with 10 billion entries.

Approximate Algorithm

We now analyze the convergence rate of the approximate algorithm with the breadth-first queue discipline. Using a breadth-first queue discipline (as in the Bellman-Ford shortest distance algorithm), updates to the shortest distance estimates in iteration k can be formulated as $D^k = MD^{k-1}$, where M is the *matrix associated with the automaton*, that is the matrix representing the weighted graph defined by the automaton. Note that the matrix multiplication here is over the $\oplus$ and $\otimes$ operations of the semiring, so that $D^k[i] = \oplus_{j=1}^{|Q|} M[i,j] \otimes D^{k-1}[j]$.

Let us focus only on the first component of the distance pair in M. Let M_1 be the matrix obtained by taking the first part of each element of M. Assume that matrix M_1 is a stochastic matrix (the transition matrix of a stochastic automaton). For any row i, $\sum_{j=1}^{|Q|} M_1[i,j] = 1$. It is not hard to see that $\vec{1} = (1, \ldots, 1)$ is an eigenvector with eigenvalue 1. By the Perron-Frobenius theorem [51], 1 is also the maximum eigenvalue and its multiplicity is 1. Using the Jordan canonical form of M_1, the updates in the kth iteration are proportional to the kth power of the second largest eigenvalue of M_1, $|\lambda_2|^k$ (see [51] for a similar analysis). Thus, for the updates are at most ϵ, when $k = \frac{\log(1/\epsilon)}{\log(1/|\lambda_2|)}$. Substituting this expression for $N(q)$ in Equation 1.3, the overall complexity of the approximate algorithm is:

$$O\left(|Q| + (|E| + |Q|\log|Q|)\frac{\log(1/\epsilon)}{\log(1/|\lambda_2|)}\right).$$

For ϵ exponentially smaller than $|\lambda_2|$ ($\epsilon = |\lambda_2|^d$), the complexity is only linear: $O(|Q| + d(|E| + |Q|))$.

It is possible to use different queue disciplines in different parts of the graph and improve the running time of the algorithm. For example, for a large graph with several strongly connected components, one can use a topological order on the component graph, with shortest-first queue discipline in each strongly connected component [76]. If there are k strongly connected components, with the ith component having n_i vertices, then the running time is given by $O(|Q|+|E| \max_{q \in Q} N(q) + \log |\max_i n_i| \sum_{q \in Q} N(q))$. If the largest component has $O(n/k)$ vertices, then this improves the general complexity by an additive factor of $\sum_{q \in Q} N(q) \log k$. Our experience with such computations for very large graphs of several million states shows that the generic topological order with the shortest-first queue discipline within each strongly connected component often leads to the most efficient results in practice.

Comparison with Previous Work

In [16], the author describes a *procedure* for an approximate computation of the relative entropy of two deterministic stochastic automata. The procedure is based on an iterative method (which can be viewed as approximating the inverse of a matrix) for computing, for a stochastic automaton A, the probability of each state q, that is the sum of the weights of all paths going through q. The convergence is claimed but not proved and no bound is indicated on the maximum number of iterations.

The author in [16] reports no complexity result for the procedure described, which makes it difficult to compare with our algorithm. Our most favorable estimate of its complexity is $\Omega(|A|^2|B|^2(T+|\Sigma|))$, where T denotes the maximum number of iterations executed. This is because the procedure requires using a matrix of size $|A|^2|B|^2$. The complexity of the procedure also depends on the size of the alphabet, which, in some applications such as natural language processing applications, may be very large. Furthermore, a lower bound on the space complexity of this procedure is $\Omega(|A|^2|B|^2)$. This makes it unsuitable for computing the relative entropy of large weighted automata. Note that the experiments reported by the author were carried out with very small grammars of about 30 rules. Nevertheless, the procedure bears some resemblance to our approximate algorithm. It can be viewed as an alphabet-dependent non-sparse implementation of that algorithm for the particular case of a FIFO queue discipline.

Experiments

We implemented both the generic Floyd-Warshall algorithm and the approximate algorithm for the computation of the relative entropy of unambiguous probabilistic automata.

To avoid the numerical instability issues related to the multiplications of probabilities, the operations of the entropy semiring are implemented so that individual transition probabilities are never explicitly exponentiated. This is achieved by using negative log probabilities instead of the probabilities them-

selves and appropriately redefining the $\times$ and the $+$ operations.

To evaluate the efficiency of our approximate algorithm for computing the relative entropy we created two n-gram statistical models trained on a large corpus – one a bigram model ($n = 2$) and one a trigram model ($n = 3$). The minimal deterministic weighted automaton representing the bigram model had about 200,000 transitions, that of the trigram model about 400,000 transitions. It took about 3s on a single 2GHz Intel processor with 128MB of RAM to compute the relative entropy of these large weighted automata using a FIFO queue discipline. With a shortest-first queue discipline, the time was reduced to 2s.

1.3.2 L_{2p} Distance

In Section 1.3.1, we saw that formulating the problem of computing the relative entropy between two probabilistic automata as a shortest-distance problem over an appropriately defined semiring immediately yields efficient polynomial time algorithms to solve the problem. Here, we extend these ideas and show that the same techniques can be used to compute the L_k distance between two probabilistic automata efficiently when k is even. To make this clear, we consider the computation of L_{2p} distance for any integer p. We will analyze acyclic, unambiguous and arbitrary probabilistic automata in turn. Subsequently, we complete this set of results by showing that the problem of computing L_k distance is NP-hard for odd k.

In [84], the authors give an approximate algorithm to compute the L_2

distance between two HMMs A_1 and A_2. Their algorithm applies to the specific cases of HMMs in which each state belongs to at most one cycle. For more general HMMs, they claim without proof that an iterative version of their method yields an approximate algorithm that works in time $O((|A_1|+|A_2|)^{6p})$. The approximation factor does not appear explicitly in this complexity term however.

This section presents a simple and general algorithm for the computation of the L_{2p} distance of two arbitrary probabilistic automata, for $p \in \mathbb{N}$. Our algorithm computes $(L_{2p}(A_1, A_2))^{2p}$. The L_{2p} distance between A_1, A_2 can then be obtained straightforwardly by taking the $2p$th root. $(L_{2p}(A_1, A_2))^{2p}$ can be rewritten as:

$$
\begin{aligned}
(L_{2p}(A_1, A_2))^{2p} &= \sum_{x\in\Sigma^*} |[\![A_1]\!](x) - [\![A_2]\!](x)|^{2p} && (1.15)\\
&= \sum_{x\in\Sigma^*} ([\![A_1]\!](x) - [\![A_2]\!](x))^{2p} && (1.16)\\
&= \sum_{x\in\Sigma^*} \sum_{i=0}^{2p} \binom{2p}{i} ([\![A_1]\!](x))^i (-[\![A_2]\!](x))^{2p-i} && (1.17)\\
&= \sum_{i=0}^{2p} \binom{2p}{i} (-1)^i \sum_{x\in\Sigma^*} ([\![A_1]\!](x))^i ([\![A_2]\!](x))^{2p-i}. && (1.18)
\end{aligned}
$$

In the first line, we could remove the absolute values since the exponent is even. This is crucial and is the reason why we need to treat the case of the L_{2p+1} distance separately.

Let $T(i, 2p-i)$ denote $\sum_{x\in\Sigma^*} ([\![A_1]\!](x))^i ([\![A_2]\!](x))^{2p-i}$. First, let us consider

the case in which A_1 and A_2 are unambiguous. If $A_k = (\Sigma, Q_i, I_i, F_i, E_i, \lambda_i, \rho_i)$, $k \in \{1, 2\}$, then the transitions in the intersection automaton $A = A_1 \cap A_2$ are defined according to the following rule:

$$(q_1, a, w_1, q_1') \in E_1 \text{ and } (q_2, a, w_2, q_2') \in E_2 \Rightarrow ((q_1, q_2), a, w_1 w_2, (q_1', q_2')) \in E. \tag{1.19}$$

Since we are dealing with unambiguous automata, we can avoid the re-computation of the intersection automaton for different values of i. During intersection, instead of multiplying w_1 and w_2, we keep the pair (w_1, w_2). Then, we only need to intersect A_1 and A_2 once, and modify the weight of each transition in the intersection automaton for different values of i in the computation of $T(i, 2p - i)$ as $((q_1, q_2), a, (w_1^i (w_2)^{2p-i}), (q_1', q_2'))$.

Note that if the underlying automata were ambiguous, then this would not work for a reason that is similar to the one due to which our algorithm fails to compute the relative entropy of ambiguous automata using the semiring framework (see Comment 1.2).

Running the shortest-distance algorithm over the intersection automaton with weights modified as described above yields $T(i, 2p - i)$. Computing the intersection automaton takes $O(|A_1||A_2|)$ time.

For automata with cycles, if we use the exact algorithm to compute the shortest-distance, then for each i, computing $T(i, 2p - i)$ costs $O(|A_1 \cap A_2|^3)$ time and $\Theta(|A_1 \cap A_2|^2)$ space. Therefore, the time complexity of computing the $2p$-distance between A_1, A_2 is $O((2p)|A_1 \cap A_2|^3)$ and the space complexity

$\Theta(|A_1 \cap A_2|^2)$.

Theorem 1.2 *The L_{2p} distance of unambiguous probabilistic automata can be computed exactly in time $O(2p|A_1|^3|A_2|^3)$.*

Note that this theorem significantly improves the result of [84], which is exponential in p. Thus, for unambiguous automata, our algorithms are, to the best of our knowledge, the only polynomial time algorithms for computing the L_{2p} distance exactly.

Note that if A_1 and A_2 are acyclic, $T(i, 2p - i)$ can be computed exactly using a generalization of the single-source shortest-distance algorithm (Section 1.2.3 and [76]) in time that is linear in the size of $A_1 \cap A_2$. Thus, for acyclic automata, the L_{2p} distance can be computed exactly in time $O(2p|A_1||A_2|)$.

For the computation of the L_{2p}-distance of arbitrary automata, we can no longer intersect A_1 and A_2 just once. Since there may be multiple paths in A_k, $k \in \{1, 2\}$ with the same label, cross terms appear in $T(i, 2p - i)$. For example if w_1 and w_2 are the weights of two paths in A_1 with labels x and the path with weight w' is the (only) path in A_2 with label x, then the contribution of string x to $T(i, 2p - i)$ is $(w_1 + w_2)^i (w')^{2p-i}$, leading to cross terms of the type $\binom{i}{j} w_1^j w_2^{i-j} w'^{2p-i}$, $j \leq i$. This makes it necessary to perform separate intersections for each i, hence a total of $2p$ intersections. The computational cost and space complexity of intersection to compute $T(i, 2p - i)$ is in $O(|A_1|^i |A_2|^{2p-i})$. Thus, the exact shortest-distance algorithm has complexity $O((|A_1|^i |A_2|^{2p-i})^3)$. This leads us to the following result.

Theorem 1.3 *The L_{2p} distance of two arbitrary probabilistic automata A_1 and A_2 can be computed in time $\sum_{i=0}^{2p} O((|A_1|^i|A_2|^{2p-i})^3) = O((|A_1|+|A_2|)^{6p})$.*

1.3.3 Hellinger Distance

The ideas presented in the previous section can be used in a straightforward manner to compute the Hellinger distance of two unambiguous probabilistic automata. The Hellinger distance $\text{Hellinger}(A_1, A_2)$ of two probabilistic automata A_1 and A_2 is given by:

$$\text{Hellinger}(A_1, A_2) = \left[\sum_{x\in\Sigma^*} \left(\sqrt{[\![A_1]\!](x)} - \sqrt{[\![A_2]\!](x)}\right)^2\right]^{1/2}. \tag{1.20}$$

Thus,

$$\begin{aligned}
[\text{Hellinger}(A_1, A_2)]^2 &= \sum_{x\in\Sigma^*} (\sqrt{[\![A_1]\!](x)} - \sqrt{[\![A_2]\!](x)})^2 \qquad (1.21)\\
&= \sum_{x\in\Sigma^*} [\![A_1]\!](x) + \sum_{x\in\Sigma^*} [\![A_2]\!](x) - 2\sum_{x\in\Sigma^*} \sqrt{[\![A_1]\!](x)[\![A_2]\!](x)}\\
&= 2\left(1 - \sum_{x\in\Sigma^*} \sqrt{[\![A_1]\!](x)[\![A_2]\!](x)}\right).
\end{aligned}$$

The problem of computing the Hellinger distance between A_1, A_2 therefore reduces to efficiently computing $\sum_{x\in\Sigma^*} \sqrt{[\![A_1]\!](x)[\![A_2]\!](x)}$. Once again, as long as A_1 and A_2 are unambiguous there is at most one accepting string with label x in $A_1 \cap A_2$. Intersecting A_1 and A_2 over the probability semiring, the weight of the transition corresponding to the intersection of the transitions

$e_1 = (q_1, a, w_1, q_1')$ and $e_2 = (q_2, a, w_2, q_2')$ is given by $w_1 w_2$.

The function $\Phi : (\mathbb{R}_+, +, \times, 0, 1) \to (\mathbb{R}_+, +, \times, 0, 1)$ defined by $\Phi(x) = \sqrt{x}$ is clearly a monoid morphism. Since $0 \leq x < 1$, $0 \leq \sqrt{x} < 1$, it also preserves closedness. Since the Φ-norm of the intersection automaton is precisely the quantity we are interested in, we obtain an efficient algorithm to compute the Hellinger distance [30, 34]. The complexity of this computation is the same as the complexity of the shortest distance algorithm on the intersection automaton $A_1 \cap A_2$. If A_1 and A_2 are acyclic, then the shortest-distance computation can be done in linear time, i.e. $O(|A_1 \cap A_2|)$. For unambiguous A_1, A_2, one could compute the Hellinger distance exactly in time that is cubic in the size of the intersection automaton and space that is quadratic using a generalization of the classical Floyd-Warshall all-pairs shortest-distance algorithm that works for arbitrary closed semirings. However, a more efficient approximate solution that uses only linear space can be obtained using the general single-source shortest-distance algorithm [76].

1.4 Hardness Results

1.4.1 L_{2p+1} Distance and L_∞ Distance

In this section, we analyze the computation of the L_k distance between probabilistic automata for odd k. In contrast with the complexity of computing the L_{2p} distance, the problem of computing the L_{2p+1} distance turns out to be NP-hard even for acyclic automata.

The problem of computing the L_1 or L_∞ distance of two probabilistic automata was shown to be NP-hard by Lyngsø and Pederson [84], even for acyclic automata. Here, we extend these results to the case of arbitrary L_{2p+1} distances, where $p \in \mathbb{N}$.

Our proof of the hardness of computing the L_{2p+1} distance between two acyclic probabilistic automata is by reduction from the Max-Clique problem and is based on a technique used in [84].

Given a graph $G = (V, E)$, one can construct an acyclic weighted automaton A_G over the probability semiring of size polynomial in $|V| + |E|$ such that $[\![A_G]\!](x) = k$ for some string x if and only if G has a clique of size k.

Let $n = |V|$. A_G is constructed as follows. It has a single initial state q_s and a single final state q_t. For each $i \in V$, it admits the following transitions:

(a) a transition from q_s to $q_{i,0}$ with label ϵ and weight 1;

(b) a transition from $q_{i,n}$ to the final state q_t with label ϵ and weight 1;

(c) a transition from $q_{i,i-1}$ to $q_{i,i}$ with label i and weight 1;

(d) a transition from $q_{i,j-1}$ to $q_{i,j}$ with label ϵ and weight 1 for each $j \neq i$; and

(e) if $(i, j) \in E$, a transition from $q_{i,j-1}$ to $q_{i,j}$ with label j and weight 1.

The size of A_G is clearly polynomial in $|V| + |E|$. Given a set $S \subseteq V$, let $[S]$ denote the ordered tuple with elements of S. For example, if $S = \{3, 1, 2\}$, then $[S] = (1, 2, 3)$. By construction, for any clique S, A_G contains a distinct

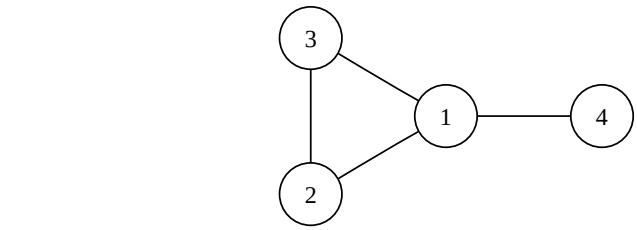
path labeled with $[S]$ starting at the initial state and going through $q_{i,0}$ for each $i \in S$ (see Fig. 1.4 for an example with $[S] = (1, 2, 3)$.) Since all accepting paths have the same weight 1, this proves the property that $[\![A]\!](x) = k$ for some string x if and only if G has a clique of size k.

The automaton A_G is not probabilistic. But, an equivalent probabilistic automaton without ϵ-transitions can be computed from A_G by using the weighted ϵ-removal algorithm [75], and a weight-pushing algorithm can be used to normalize the sum of its weights to one [73]. We first establish the result with A_G and later describe how to convert A_G into a probabilistic automaton.

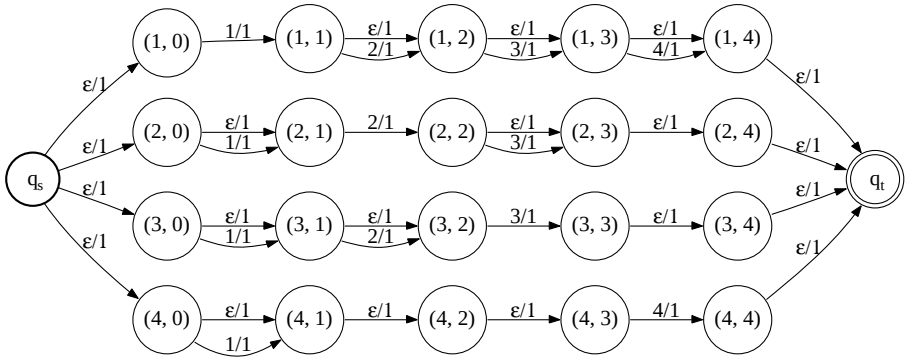
Theorem 1.4 *The problem of computing the L_{2p+1} distance of two probabilistic automata is NP-hard.*

Proof. The proof is based on a reduction using an algorithm for the computation of the L_{2p+1} distance as a subroutine to define an algorithm for solving the Max-Clique problem. Using the notation adopted in [84], let a_k denote the number of strings accepted by A_G with weight exactly k. Thus determining the maximum k such that $a_k \neq 0$ is equivalent to determining the size of the largest clique.

For each $i \in \{0, 1, \ldots, n\}$, let C_i denote the constant weighted automaton assigning the same weight i to all ordered subsequences of $\{1, \ldots, n\}$ and weight 0 to all other strings. Fig. 1.5 shows the constant automaton for $n = 4$.



(a)



(b)

Figure 1.4: (a) Undirected graph $G = (V, E)$. (b) The corresponding automaton A_G constructed in the reduction.

By definition of the L_{2p+1} distance,

$$\forall i \geq 0, \quad [L_{2p+1}(C_i, A_G)]^{2p+1} = \sum_{x \in \Sigma^*} |[\![A_G]\!](x) - [\![C_i]\!](x)|^{2p+1} \quad (1.22)$$

$$= \sum_{x \in \Sigma^*} |[\![A_G]\!](x) - i|^{2p+1} \quad (1.23)$$

$$= \sum_{j=0}^{n} a_j |i - j|^{2p+1} \quad (1.24)$$

This defines a system of linear equation with unknown variables a_j, $j =$

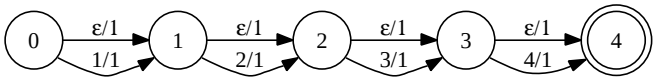


Figure 1.5: The constant automaton C_1 for G assigning weight 1 to all subsequences of the set $\{1, \ldots, 4\}$. Note that the final state has a final weight of 1.

$0, \ldots, n$. Let $M \in \mathbb{R}^{(n+1)\times(n+1)}$ be the matrix defined by $M_{i,j} = |i-j|^{2p+1}, i \in \{0, 1, \ldots, n\}$ and let $A \in \mathbb{R}^{n+1}$ be the column vector containing the a_js. If M is invertible, then A can be defined with respect the L_{2p+1} distance of the automata C_i and A_G, which will prove the statement of the theorem.

This matrix is a specific Toeplitz matrix, but it is not straightforward to compute its determinant [84]. Instead, we can do our reasoning in $\mathbb{Z}_3$. In $\mathbb{Z}_3$, the coefficients of M are either 0, 1, or -1, regardless of the value of p. The determinant of M in $\mathbb{Z}_3$ is given by:

$$\det(M) = \begin{cases} -1 & \text{if } n+1 \equiv 2 \pmod 3 \\ 1 & \text{if } n+1 \equiv 0 \pmod 3 \\ 0 & \text{if } n+1 \equiv 1 \pmod 3. \end{cases}$$

We delay the proof of this fact to Lemma 1.4.

This implies that for all $n \in \mathbb{N}$ such that n is of the form $n+1 \equiv 0 \pmod 3$ or $n+1 \equiv 2 \pmod 3$, the matrix M of size $(n+1)\times(n+1)$ defined by $M_{i,j} = |i-j|^{2p+1}, i \in \{0, 1, \ldots, n\}$ is invertible in $\mathbb{R}$. Therefore, for $n \equiv \pm 1 \pmod 3$, one can compute the column vector A and determine the size of the

largest clique in the original graph G. This leaves us only with the case where $n \equiv 0 \pmod 3$ in the original graph $G = (V, E)$. But, in this case, one can add a "dummy vertex" to G that is connected to all other vertices of V. Doing so increases the size of the largest clique by exactly one, and yields a graph $G' = (V', E')$ with $|V'| \equiv 1 \pmod 3$. Since the size of the largest clique in G is one less than the size of the largest clique in G', the reduction is complete. Thus, the problem of determining the computing $2p+1$ distance between two probabilistic automata is NP-hard. □

We conjecture that the problem of computing the L_{2p+1} distance, or L_∞, is in fact undecidable. Note that it was shown in [84] that, in view of the hardness of approximation results for cliques [95, 47], even a polynomial approximation of the L_∞ distance within a factor of $n^{\frac{1}{4}-\epsilon}$ is impossible unless NP = P.

Lemma 1.4 *The determinant of M in $\mathbb{Z}_3$ is given by*

$$\det(M) = \begin{cases} -1 & \text{if } n+1 \equiv 2 \pmod 3 \\ 1 & \text{if } n+1 \equiv 0 \pmod 3 \\ 0 & \text{if } n+1 \equiv 1 \pmod 3. \end{cases}$$

Proof. Let $M[n+1] \in \mathbb{R}^{(n+1)\times(n+1)}$ be the matrix defined by $M_{i,j} \equiv |i-j| \pmod 3$. Note that $|i-j|^{2p+1} \pmod 3 \equiv |i-j| \pmod 3$ for all $p \in \mathbb{N}$. To remain consistent with the previous description, throughout this proof, we consider the matrix M of size $(n+1) \times (n+1)$.

Let R_i, C_j denote the ith row and the jth column of M respectively. We

prove the lemma by showing that the following three identities in $\mathbb{Z}_3$ hold for all $k \in \mathbb{N}$, $k \geq 2$:

$$\begin{aligned} \det(M[3k+1]) &= 0 \\ \det(M[3k+2]) &= -\det(M[3k]) \\ \det(M[3k]) &= \det(M[3k-4]) - \det(M[3k-3]). \end{aligned}$$

Case 1. $n+1 \equiv 1 \pmod 3$**.** Let $n+1 = 3k+1$ for some $k \in \mathbb{N}$. For all $j \in \{1, \ldots, 3k+1\}$,

$$M_{3k+1,j} \equiv |3k+1-j| \pmod 3 \equiv (1-j) \pmod 3 \equiv -|1-j| \pmod 3 = -M_{1,j}.$$

Since the last row is a scalar multiple of the first row, $\det(M) = 0$ for $n+1 \equiv 1 \pmod 3$.

Case 2. $n+1 \equiv 2 \pmod 3$**.** Let $n+1 = 3k+2$ for some $k \in \mathbb{N}$. In this case, we show that $\det(M[3k+2]) = -\det(M[3k])$. Given $M[3k+2]$, we perform the following symmetric row and column operations:

$$R_1 \leftarrow R_1 + R_{3k+1} \qquad C_1 \leftarrow C_1 + C_{3k+1}. \tag{1.25}$$

Note that in Case 1, we observed that R_{3k+1} was the negation of R_1. The same argument shows that the above row operation will set all but the last entry in the first row (and by symmetry, in the first column) to zero. Let M' denote the resulting matrix. Then, $M'_{1,i} = M'_{i,1} = 0$ for $1 \leq i \leq 3k+1$ and

$M'_{1,3k+2} = M'_{3k+2,1} = -1$. The entries in rows and columns 2 through $3k+1$ are unaffected. Let S be the submatrix of M' induced by rows $\{2, \ldots, 3k+2\}$ and columns $\{1, \ldots, 3k+1\}$. Fig. 1.6(a) illustrates the structure of the matrix M'. Developing the determinant of M' along R_1 and simplifying the powers of -1 yield:

$$\det(M) = \det(M') = (-1)^{(3k+2)+1} \left[(-1) \det(S)\right] = (-1)^{3k} \det(S). \tag{1.26}$$

Developing the determinant of S along the first column leads to:

$$\det(S) = (-1)^{(3k+1)+1} \left[(-1) \det(M[3k])\right] = (-1)^{3(k+1)} \det(M[3k]). \tag{1.27}$$

It follows that:

$$\det(M) = (-1)^{3(2k+1)} \det(M[3k]) = -\det(M[3k]).$$

Case 3. $n+1 \equiv 0 \pmod 3$**.** Let $n+1 = 3k$ for $k \in \mathbb{N}$. We show that $\det(M[3k]) = \det(M[3k-4]) - \det(M[3k-3])$. Given $M[3k]$, we perform the

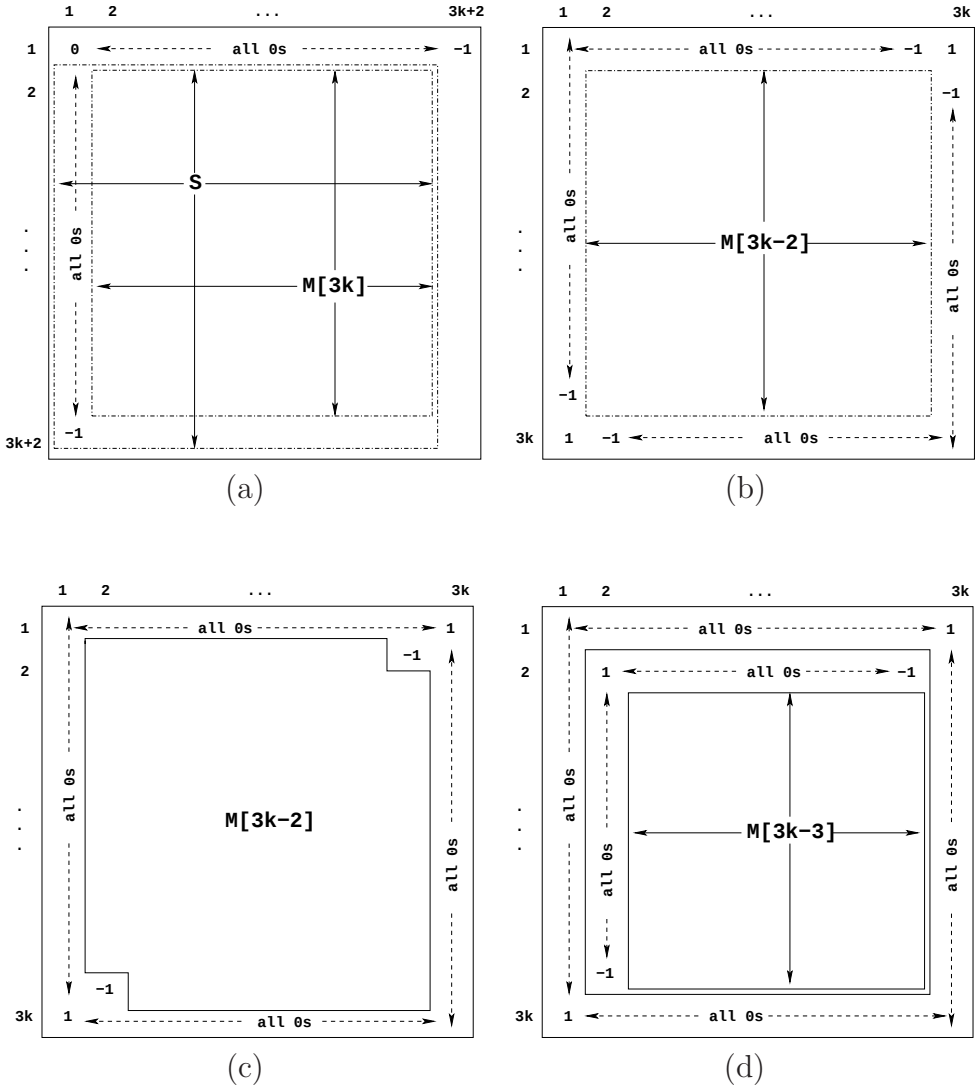


Figure 1.6: (a) Case 2. The matrix M' obtained from $M[3k+2]$ after the row and column operations described in Equation 1.25. (b) Case 3. The matrix obtained from $M[3k]$ after the first four (row and column) operations described in Equation 1.28. (c) Case 3. The matrix obtained after the next four (row and column) operations. (d) Case 3. The final matrix after all row and column operations.

following symmetric operations:

$$\begin{array}{llll} R_1 & \leftarrow R_1 + R_{3k-2} & C_1 & \leftarrow C_1 + C_{3k-2} \\ R_{3k} & \leftarrow R_{3k} + R_3 & C_{3k} & \leftarrow C_{3k} + C_3 \\ R_2 & \leftarrow R_2 + R_1 & C_2 & \leftarrow C_2 + C_1 \\ R_{3k-1} & \leftarrow R_{3k} + R_{3k-1} & C_{3k-1} & \leftarrow C_{3k} + C_{3k-1} \\ R_2 & \leftarrow R_2 + R_{3k-1} & C_2 & \leftarrow C_2 + C_{3k-1}. \end{array} \tag{1.28}$$

The entries of the resulting matrix are all zero in the first and last row and column, except for $M_{1,3k} = 1, M_{3k,1} = 1$ (see Fig. 1.6(b), 1.6(c) and 1.6(d)). Let S denote the submatrix induced by rows i and j with $i, j \in \{2, \ldots, 3k-1\}$. Thus S is a $(3k-2) \times (3k-2)$ matrix. For S, we have $S_{1,1} = 1, S_{1,3k-2} = -1, S_{3k-2,1} = -1$. The remaining entries in the first row and the first column of S are all zero. Furthermore, the submatrix of S induced by rows i and j with $i, j \in \{3, \ldots, 3k-1\}$ is the same as $M[3k-3]$. Developing the determinant of S along the first row and simplifying the powers of -1 yields:

$$\det(S) = \det(M[3k-3]) - \det(M[3k-4]).$$

Developing the determinant of matrix M after the row and column operations described above along R_1 followed by R_{3k} (both these rows have only one non-zero entry, namely, $M_{1,3k} = M_{3k,1} = 1$) yields:

$$\det(M[3k]) = -\det(S) = \det(M[3k-4]) - \det(M[3k-3]),$$

and ends the proof. □

We now comment on the fact that the automata A_G and C_i are not probabilistic. Let $L(A)$ denote the language accepted by automaton A and $\deg(v)$ denote the degree of vertex v in G. The analysis presented here is similar to that of [84]; we outline it for the sake of completeness.

Lemma 1.5 *The sums of the weights of all accepting paths in A_G and C_i are given by*

$$\sum_{x \in L(A_G)} [\![A_G]\!](x) = \sum_{v \in V} 2^{\deg(v)}, \qquad \sum_{x \in L(C_i)} [\![C_i]\!](x) = i|L(C_i)| = i2^n.$$

Proof. Since each transition in A_G has weight 1, the weight of every accepting path in A_G is 1. Thus, the sum of the weights of all accepting paths in A_G is exactly equal to the number of accepting paths. Let N_i denote the number of accepting paths in A_G that pass through state $q_{i,0}$. A vertex $i \in V$ has $\deg(i)$ vertices adjacent to it in G. By construction, we introduce two transitions from state $q_{i,j-1}$ to $q_{i,j}$ for each neighbor j of i, one with label j and weight 1 and another with label ϵ and weight 1, and this doubles the number of accepting paths that pass through $q_{i,0}$. Thus, $N_i = 2^{\deg(i)}$ and the number of accepting paths in A_G is equal to $\sum_{i=0}^{n} 2^{\deg(i)}$.

For automaton C_i, each string has weight i, and the language accepts 2^n strings. Thus, the sum of the weights of all accepting paths in C_i is exactly $i2^n$. □

Let Z_G denote $\sum_{v \in V} 2^{\deg(v)}$. One way to make A_G and C_i probabilistic is to assign a final weight $1/Z_G$ to A_G and $1/i2^n$ to C_i. However, this would result in a modification of matrix M as $M_{i,j}$ would then become $|i/(i2^n) - j/Z_G|^{2p+1}$ and we wish to use our proof of the invertability of M for $M_{i,j} = |i-j|^{2p+1}$ for $n+1 \not\equiv 1 \pmod 3$. This can be achieved as follows:

1. If $Z_G \geq i2^n$, then we normalize both automata A_G and C_i by assigning them the final weight $1/Z_G$. The sum of the weights of all accepting paths in A_G is then one but that in C_i is given by $i2^n/Z_G$, which is less than one. To make C_i probabilistic (i.e. the sum of the weights of all accepting paths in C_i is exactly one), we introduce a new symbol, say \$, and add a transition in C_i from its start state to its final state with input label \$ and weight $1 - i2^n/Z_G$. Let $\widehat{A}_G, \widehat{C}_i$ denote automata A_G and C_i modified as described. It is straightforward to verify that

$$\left[L_{2p+1}(\widehat{A}_G, \widehat{C}_i)\right]^{2p+1} = \sum_{j=0}^{n} \frac{a_j}{Z_G}|i-j|^{2p+1} + \left(1 - \frac{i2^n}{Z_G}\right)^{2p+1}. \qquad (1.29)$$

2. If $Z_G < i2^n$, then we normalize A_G and C_i by assigning them the final weight $1/i2^n$. Now the sum of the weights of all accepting paths in C_i is one but that in A_G is given by $Z_G/i2^n$. Again, we can introduce a new symbol, say \$, and add a transition in A_G from its start state to its final state with input label \$ and weight $1 - Z_G/i2^n$. For the modified

automata $\widehat{A}_G$ and $\widehat{C}_i$, as before, we obtain

$$\left[L_{2p+1}(\widehat{A}_G, \widehat{C}_i)\right]^{2p+1} = \sum_{j=0}^{n} \frac{a_j}{i2^n}|i-j|^{2p+1} + \left(1 - \frac{Z_G}{i2^n}\right)^{2p+1}. \quad (1.30)$$

By Equation 1.29 and Equation 1.30, the following holds:

$$\sum_{j=0}^{n} a_j|i-j|^{2p+1} = \begin{cases} Z_G\left((L_{2p+1}(\widehat{A}_G, \widehat{C}_i))^{2p+1} + (1 - \frac{i2^n}{Z_G})^{2p+1}\right) & \text{if } Z_G \geq i2^n \\ i2^n\left((L_{2p+1}(\widehat{A}_G, \widehat{C}_i))^{2p+1} + (1 - \frac{Z_G}{i2^n})^{2p+1}\right) & \text{if } Z_G < i2^n. \end{cases} \quad (1.31)$$

Since it is NP-hard to compute $\sum_{j=0}^{n} a_j|i-j|^{2p+1}$ for all i (by the previous reduction), it must be NP-hard to compute the L_{2p+1} distance between $\widehat{A}_G$ and $\widehat{C}_i$, which are both probabilistic.

Absolute Value Automata

The hardness results for the computation of the L_{2p+1} distances of probabilistic automata seem to be related to the obligatory presence of the absolute values in the definition of these distances. This brings us to examine several questions related to the absolute value.

In particular, one may ask if in general there exists a weighted automaton C over the real semiring $(\mathbb{R}, +, \times, 0, 1)$ representing the absolute value of the difference of two probabilistic automata A and B, that is such that

$$\forall x \in \Sigma^*,\ [\![C]\!](x) = |[\![A]\!](x) - [\![A_2]\!](x)|.$$

We could refer to C as the *absolute value automaton* and denote it by $|A - B|$. The general existence of C and even its efficient computation would not be sufficient to guarantee the efficient computability of the L_1 distance (or L_{2p+1} distance).

Indeed, by the definition of C, to compute the L_1 distance of A and B, one can sum the weights of all successful paths of C. But, since the semiring $(\mathbb{R}, +, \times, 0, 1)$ is not closed, no general algorithm is available for computing this sum. Note that $[\![C]\!]$ takes its values in $\mathbb{R}_+$, but this does not imply that its transition weights are necessarily in $\mathbb{R}_+$, nor does it even imply the existence of an equivalent weighted automaton C' over $(\mathbb{R}_+, +, \times, 0, 1)$. This is because $\mathbb{R}$ is not a *Fatou extension* of $\mathbb{R}_+$ [12]; indeed there exist weighted automata over the real semiring taking their values in $\mathbb{R}_+$ that do not admit an equivalent weighted automaton over $(\mathbb{R}_+, +, \times, 0, 1)$ [85, 63].

However, the hardness of the computation of the L_1 distance guarantees that in general, unless P = NP, there exists no absolute value weighted automaton C over $(\mathbb{R}_+, +, \times, 0, 1)$ that can be computed efficiently since the sum of the weights of the paths of C, i.e., the L_1 distance, could then be computed efficiently.

Note also that the general problem of determining if a weighted automaton A defined over the real semiring $(\mathbb{R}, +, \times, 0, 1)$ accepts no string of negative weight is undecidable [85, 63]. Since there exists an efficient algorithm for testing the equivalence of two weighted automata over the real semiring [90], this implies that in general there does not exists a computable absolute value

automaton $|A|$ such that $\forall x \in \Sigma^*,\ [\![|A|]\!](x) = |[\![A]\!](x)|$.

1.4.2 Relative Entropy of Arbitrary Automata

This section proves a hardness result suggesting that the problem of computing the relative entropy of arbitrary probabilistic automata is intractable.

Hardness Result

We describe a reduction of the problem of determining whether the language accepted by an automaton is Σ^* to the that of determining whether the relative entropy of two probabilistic automata is infinite.

Automaton A_0. We first describe an automaton A_0 that is used in our reduction. Fix a real number $\alpha > 0$ such that $\alpha|\Sigma| < 1$ and let A_0 be the one-state weighted automaton representing the weighted regular expression $(1-\alpha)(\sum_{x\in\Sigma} \alpha x)^*$ shown in Figure 1.7 for $\Sigma = \{a, b\}$. By definition, A_0 accepts all strings $x \in \Sigma^*$ and for all $x \in \Sigma^*$, $[\![A]\!](x) = \alpha^{|x|}(1 - |\Sigma|\alpha)$, where $|\Sigma|\alpha < 1$. By construction, A_0 is stochastic and thus probabilistic. Here also is a direct verification:

$$\sum_{x\in\Sigma^*} [\![A_0]\!](x) = \sum_{n=0}^{\infty} \sum_{|x|=n} \alpha^n(1 - |\Sigma|\alpha) = \sum_{n=0}^{\infty} |\Sigma|^n \alpha^n (1 - |\Sigma|\alpha) \quad (1.32)$$

$$= (1 - |\Sigma|\alpha)\frac{1}{1 - |\Sigma|\alpha} = 1. \quad (1.33)$$

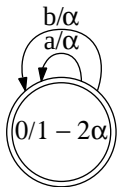


Figure 1.7: The automaton A_0 that accepts all strings, $\{a, b\}^*$, and assigns a weight of $\alpha^n(1-\alpha)$ to any string of length n. $\alpha > 0$ is a constant such that $2\alpha < 1$.

The following theorem shows that the problem of determining the relative entropy of two arbitrary probabilistic automata is at least as hard as determining if a finite automaton accepts Σ^*.

Theorem 1.5 *Let A be an arbitrary probabilistic automaton, then $D(A_0\|A) < \infty$ if and only if A accepts Σ^*.*

Proof. Assume that $[\![A]\!](x) = 0$ for some $x \in \Sigma^*$. Then, since $[\![A_0]\!](x) > 0$, $[\![A_0]\!](x) \log \frac{[\![A_0]\!](x)}{[\![A]\!](x)}$ is infinite and $D(A_0\|A) = \infty$.

Assume now that A accepts Σ^*, thus $[\![A]\!](x) \neq 0$ for all $x \in \Sigma^*$. Without loss of generality, we can assume A to be trim. Let E denote the set of transitions of A and let δ denote the minimum weight of a transition: $\delta = \min_{e \in E} w[e]$. By assumption, $\delta > 0$ since the automaton A is trim and probabilistic. For $x \in \Sigma^*, |x| = n$, $[\![A]\!](x) \geq \delta^n$. Thus

$$\forall x \in \Sigma^*, \ \frac{[\![A_0]\!](x)}{[\![A]\!](x)} = \frac{\alpha^n(1-|\Sigma|\alpha)}{[\![A]\!](x)} \leq (1-|\Sigma|\alpha)\left(\frac{\alpha}{\delta}\right)^n.$$

It follows that:

$$\forall x \in \Sigma^*,\ [\![A_0]\!](x) \log \frac{[\![A_0]\!](x)}{[\![A]\!](x)} \leq \alpha^n (1 - |\Sigma|\alpha) \left(n \log(\alpha/\delta) + \log(1 - |\Sigma|\alpha)\right).$$

For any positive integer N, summing over all strings x of length at most N, in the order of increasing $|x|$, yields:

$$\begin{aligned} \sum_{|x| \leq N} [\![A_0]\!](x) \log \frac{[\![A_0]\!](x)}{[\![A]\!](x)} &= \sum_{n=0}^{N} \sum_{x:|x|=n} [\![A_0]\!](x) \log \frac{[\![A_0]\!](x)}{[\![A]\!](x)} \qquad (1.34) \\ &\leq \sum_{n=0}^{N} |\Sigma|^n \alpha^n (1 - |\Sigma|\alpha) \left(n \log(\alpha/\delta) + \log(1 - |\Sigma|\alpha)\right). \end{aligned}$$

Since $\alpha|\Sigma| < 1$ the two series in this summation, $\sum_n n\beta^n$ and $\sum_n \beta^n$ with $\beta = |\Sigma|\alpha < 1$, converge. It is straightforward to verify that for $0 \leq \beta < 1$, $\sum_{n=0}^{\infty} n\beta^n = \frac{\beta}{(1-\beta)^2}$. Using this identity, we obtain the following bound on $D(A_0 \| A)$:

$$D(A_0 \| A) \leq (1 - |\Sigma|\alpha) \left(\frac{|\Sigma|\alpha \log(\alpha/\delta)}{(1 - |\Sigma|\alpha)^2} + \frac{\log(1 - |\Sigma|\alpha)}{1 - |\Sigma|\alpha} \right). \qquad (1.35)$$

Thus $D(A_0 \| A) < \infty$. □

Theorem 1.6 *The problem of computing the relative entropy of two arbitrary probabilistic automata is PSPACE-hard.*

Proof. The universality problem, i.e., the problem of deciding if a trim finite automaton A accepts Σ^*, is PSPACE-complete [96, 49]. The transitions of

any trim finite automaton A can be augmented with positive weights so that it becomes a probabilistic automaton. This can be done by weighting each outgoing transition of state q, or final weight if q is final, by $1/n_q$ where n_q is the out-degree of q, plus one if q is final. The encoding of $1/n_q$ takes $O(\log_2 n_q)$ space, thus the space and time complexity of this construction is polynomial in the size of A. By Theorem 1.5, it can be decided if a probabilistic automaton A accepts all strings by computing the relative entropy $D(A_0 \| A)$ and testing its finiteness. Thus, the computation of the relative entropy can determine if a trim finite automaton A accepts Σ^*. $\square$

Remarks

Theorem 1.6 suggests that the general problem of computing the relative entropy of arbitrary probabilistic automata is intractable. However, one may resort to various approximations of practical importance. An example is an approximation based on the use of the log-sum inequality in [93] in the context of machine learning. We have initiated a specific study of such approximations, in particular by examining the quality of an approximation when using the algorithms we presented for the unambiguous case. Initial results are presented in Section 1.7.4.

Note that the general problem of determining if a weighted automaton over the $(\mathbb{R}, +, \times, 0, 1)$ semiring accepts the full free monoid Σ^* is undecidable [12]. Here, we are considering the same decidability question but only for probabilistic automata, which form a restricted class of all weighted automata

over the $(\mathbb{R}, +, \times, 0, 1)$ semiring. However, we conjecture that the problem is in fact undecidable even in this case.

1.5 Equivalence of Probabilistic Automata

Clearly, our algorithm for computing the L_{2p} distance of two arbitrary probabilistic automata A_1 and A_2 also provides an efficient method for testing their equivalence since A_1 and A_2 are equivalent if and only if their L_p distance is zero. For $p = 1$, our exact algorithm can be used to test for equivalence in time $O((|A_1||A_2|)^3)$. However, the standardization algorithm of Schützenberger [90] can be used to derive a more efficient algorithm.

Theorem 1.7 *The equivalence of two arbitrary probabilistic automata A_1 and A_2 can be computed in time $O(|\Sigma|\,(|A_1| + |A_2|)^3)$.*

Proof. The standardization algorithm of Schützenberger [90, 12] applies to any weighted automaton defined over a field. It leads to a representation of a weighted automaton with the smallest number of states. The algorithm requires the construction of bases for vectorial spaces for which spanning sets are known. Using LUP decompositions, the complexity of the standardization algorithm applied to a weighted automaton A is in $O(|\Sigma||A|^3)$.

For the purpose of equivalence, we may view a probabilistic automaton as an automaton over the field $(\mathbb{R}, +, \cdot, 0, 1)$. Since negation is allowed over this field, we can construct the automaton $A = A_1 - A_2$, in linear time, and

apply standardization. A_1 and A_2 are equivalent if and only if A is equivalent to the zero weighted machine, that is if and only after standardization A has no state. Thus, this leads to an algorithm for testing the equivalence of two probabilistic automata A_1 and A_2 with overall complexity $O(|\Sigma|\,|A|^3) = O(|\Sigma|\,(|A_1| + |A_2|)^3)$. $\square$

To our knowledge, this is the most efficient algorithm for testing the equivalence of probabilistic automata. Note that the same algorithm can be used to test the equivalence of probabilistic automata as defined by Rabin [81]. The best algorithm previously reported in the literature was that of Wen-Guey Tzeng whose complexity is $O(|\Sigma|\,(|A_1|+|A_2|)^4)$ [99]. The alphabet factor does not appear in the expression of the complexity reported by the author most likely because the proof is restricted to a binary alphabet. The technique described by Wen-Guey Tzeng is in fact closely related to the standardization algorithm of Schützenberger [90], which the author was apparently not aware of.

Also there is a claim in [2] that the equivalence of weighted automata with transition weights in $\mathbb{Z}$ can be tested in cubic time. However, the paper does not include a full proof of the correctness of the algorithm and its complexity. Instead it relies on several claims made by others in private communications or results appearing in a Siberian journal not accessible to us. It also seems to be specifically using the property of the coefficients being integers. The algorithm we are describing does not require transition weights to be integers and applies to all probabilistic automata and other weighted automata with

real-valued weights.

1.6 Relative Entropy As a Kernel

This section examines the use of the relative entropy, or its symmetricized version, in machine learning algorithms. The results hold in general and are not limited to the particular case of probabilistic automata.

In machine learning, functions $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ are called *kernels*. A kernel is said to be *positive definite symmetric* (PDS for short) if it is *symmetric*, $K(x, y) = K(y, x)$ for all $x, y \in \mathcal{X}$, and if for any subset $\{x_1, \ldots, x_m\} \subseteq \mathcal{X}$, the eigenvalues of the matrix $[K(x_i, x_j)]_{1 \leq i,j \leq m}$ are non-negative. PDS kernels play an important role in machine learning since they can be combined with discriminant algorithms such as support vector machines (SVMs) to create powerful predictors [88], the PDS condition ensuring the convergence of training.

In some cases, a symmetric kernel K is not positive definite but $\exp(-\lambda K)$ is PDS for any $\lambda > 0$. K is then said to be *negative definite symmetric* (NDS). Such kernels are also important since they can be used to defined PDS kernels as in the case of Gaussian kernels.

However, we will show that the symmetricized relative entropy is neither PDS nor NDS, contrary to what is stated in a number of machine learning papers, which limits its use and application in kernel methods.

The *symmetricized relative entropy* of two distributions p and q is given

by:

$$D_{sym}(p\|q) = \frac{D(p\|q) + D(q\|p)}{2} = \sum_{x\in\mathcal{X}} [p(x) - q(x)] \log \frac{p(x)}{q(x)}.$$

Theorem 1.8 *The symmetricized relative entropy is not a PDS kernel.*

Proof. Let $\{q_1, q_2, \ldots, q_m\}$ be a set of probability distributions over $\mathcal{X}$. Consider the Gram matrix $\mathbf{K} \in \mathbb{R}^{m\times m}$ defined by $\mathbf{K}_{i,j} = D_{sym}(q_i\|q_j)$. By definition of D_{sym}, $D_{sym}(q_i\|q_i) = 0$ for all $i \in [1, m]$, thus $\mathbf{tr}(\mathbf{K}) = 0$. When $\mathbf{K} \neq 0$, this implies that $\mathbf{K}$ admits at least one negative eigenvalue [51]. □

To show that the symmetricized relative entropy is not an NDS kernel, we use the following theorem from [87].

Theorem 1.9 ([87, 11]) *Let $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be an NDS kernel such that for $x, y \in \mathcal{X}$, $K(x, y) = 0$ if and only if $x = y$. Then, there exists a Hilbert space H and a mapping $\Phi : \mathcal{X} \to H$ such that*

$$\forall x, y \in \mathcal{X},\ K(x, y) = ||\Phi(x) - \Phi(y)||^2.$$

Under the hypothesis of the theorem, $\sqrt{K}$ defines a metric.

Theorem 1.10 *The symmetricized relative entropy is not an NDS kernel.*

Proof. Note that for any two distributions p and q, $D(p\|q) \geq 0$ and $D(q\|p) \geq 0$. Since $D_{sym}(p\|q) = 0$ is the average of $D(p\|q)$ and $D(q\|p)$, it is zero if and only if $D(p\|q) = 0$ and $D(q\|p) = 0$. This happens only when $p = q$. Thus, by

Theorem 1.9, if D_{sym} is an NDS kernel, $\sqrt{D_{sym}}$ defines a metric. We prove that $\sqrt{D_{sym}}$ does not obey the triangle inequality, which will show that D_{sym} is not NDS.

For simplicity, the proof is given in the case of a universe of events limited to two elements: $\mathcal{X} = \{x_1, x_2\}$. Let $\epsilon > 0$ and let q_1, q_2, q_3 be the three distributions over $\mathcal{X}$ defined by:

$$\forall i \in [1,3],\ q_i(x_1) = 1 - i\epsilon \text{ and } q_i(x_2) = i\epsilon.$$

By definition of the symmetricized relative entropy,

$$D_{sym}(q_1\|q_2) = \epsilon \log \frac{1-\epsilon}{1-2\epsilon} - \epsilon \log \frac{\epsilon}{2\epsilon} = \epsilon \log \frac{2(1-\epsilon)}{1-2\epsilon}.$$

Similarly, $D_{sym}(q_2\|q_3) = \epsilon \log \frac{3(1-2\epsilon)}{2(1-3\epsilon)}$ and $D_{sym}(q_1\|q_3) = 2\epsilon \log \frac{3(1-\epsilon)}{1-3\epsilon}$. Note that:

$$\begin{aligned} D_{sym}(q_1\|q_3) &= 2\epsilon \log \tfrac{3(1-\epsilon)}{1-3\epsilon} = 2\big(\epsilon \log \tfrac{2(1-\epsilon)}{1-2\epsilon} + \epsilon \log \tfrac{3(1-2\epsilon)}{2(1-3\epsilon)}\big) \\ &= 2\big(D_{sym}(q_1\|q_2) + D_{sym}(q_2\|q_3)\big). \end{aligned}$$

$$\sqrt{D_{sym}(q_1\|q_3)} = \sqrt{2D_{sym}(q_1\|q_2) + 2D_{sym}(q_2\|q_3)}$$

Since $\sqrt{\cdot}$ is strictly concave, for positive $x > 0, y > 0$, $x \neq y$ (i.e. $\epsilon \neq \frac{7}{15}$),

$$\sqrt{\frac{x+y}{2}} > \frac{\sqrt{x} + \sqrt{y}}{2}.$$

Therefore,

$$\sqrt{\frac{D_{sym}(q_1\|q_2) + D_{sym}(q_2\|q_3)}{2}} > \frac{\sqrt{D_{sym}(q_1\|q_2)} + \sqrt{D_{sym}(q_2\|q_3)}}{2}.$$

Or, in other words,

$$\begin{aligned} \sqrt{D_{sym}(q_1\|q_3)} &= \sqrt{2D_{sym}(q_1\|q_2) + 2D_{sym}(q_2\|q_3)} && (1.36) \\ &> \sqrt{D_{sym}(q_1\|q_2)} + \sqrt{D_{sym}(q_2\|q_3)}. && (1.37) \end{aligned}$$

Thus, $\sqrt{D_{sym}}$ does not obey the triangle inequality and the proof is complete. □

1.7 Computation of the Norm

In Section 1.3.1, we gave a general algorithm for computing the relative entropy of two unambiguous probabilistic automata by relating this problem to a shortest-distance problem over the appropriate semiring. A special case of that algorithm can be used to compute the entropy of a single unambiguous probabilistic automaton. One may ask if such results could be generalized to the computation of other similar quantities that we will refer to as the *norm of an unambiguous probabilistic automaton*. This section shows how they can be generalized by considering an arbitrary monoid morphism.

1.7.1 Norm of an Unambiguous Automaton

Let $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ be a closed semiring, or an ϵ-k-closed semiring for an automaton A. Let $\Phi : (\mathbb{R}_+, \cdot, 1) \to (\mathbb{K}, \otimes, \overline{1})$ be a monoid morphism. We will say that Φ *preserves closedness*, if for all x, $0 \leq x < 1$, $\bigoplus_{n=0}^{\infty} \Phi(x^n)$ is well-defined and in $\mathbb{K}$. For a such a morphism, we can define the Φ-*norm of a probabilistic automaton* as:

$$\|A\|_\Phi = \bigoplus_{x \in \Sigma^*} \Phi([\![A]\!](x)).$$

Theorem 1.11 *Let $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ be a closed or ϵ-k-closed semiring and let $\Phi : (\mathbb{R}_+, \cdot, 1) \to (\mathbb{K}, \otimes, \overline{1})$ be a monoid morphism preserving closedness. Then, for any unambiguous probabilistic automaton A, $\|A\|_\Phi$ can be computed exactly in time $O(|A|^3)$.*

Proof. The automaton $\Phi(A)$ derived from A by replacing each weight x by $\Phi(x)$ is a weighted automaton over the semiring $\mathbb{K}$. Since A is unambiguous, at most one successful path in A, $\pi = e_1 \cdots e_k$, is labeled with any string $x \in \Sigma^*$. Since Φ is a monoid morphism, $\Phi([\![A]\!](x)) = \bigotimes_{j=1}^{k} \Phi(w[e_j])$, that is the weight of the path labeled with x in $\Phi(A)$. This shows that $\|A\|_\Phi = s(A)$ and proves the theorem. □

Theorem 1.11 provides an algorithm for computing the Φ-norm of unambiguous probabilistic automata for arbitrary monoid morphisms preserving closedness. We will briefly illustrate two applications of the theorem.

(a) Entropy of a Probabilistic Automaton.

Let $(\mathbb{K}, \oplus, \otimes, (0,0), (1,0))$ be the entropy semiring. It is not hard to see that the function $\Phi : (\mathbb{R}_+, +, \times, 0, 1) \to (\mathbb{K}, \oplus, \otimes, (0,0), (1,0))$ defined by: $\forall x \in \mathbb{R}_+, \Phi(x) = (x, -x \log x)$, is a monoid morphism preserving closedness. Thus, the Φ-norm of an unambiguous probabilistic automaton can be computed efficiently using a single-source shortest-distance algorithm. Its second component is exactly the entropy of A, thus this provides an efficient and simple algorithm for computing the entropy of A.

(b) Norm L_α of a Probabilistic Automaton, $\alpha \in \mathbb{R}_+$.

The function $\Phi : (\mathbb{R}_+, +, \times, 0, 1) \to (\mathbb{R}_+, +, \times, 0, 1)$ defined by $\Phi(x) = x^\alpha$ is clearly a monoid morphism. Since for $0 \leq x < 1$, $0 \leq x^\alpha < 1$, it also preserves closedness. Thus, the L_α-norm of an unambiguous probabilistic automaton A can be computed efficiently using a shortest-distance algorithm. In particular, the Bhattacharya norm of A, its $L_{\frac{1}{2}}$-norm, can be computed efficiently.

1.7.2 Norm of Arbitrary Automata

In general, a probabilistic automaton may not be unambiguous. But the L_p-norm can still be computed in polynomial time for any integer $p \geq 1$.

Theorem 1.12 *The L_p-norm of a probabilistic automaton A can be computed exactly in time $O(|A|^{3p})$ time and $\Theta(|A|^{2p})$ space.*

Proof. Let $A^{(p)}$ denote the automaton obtained by intersecting A with itself $p-1$ times. Then, by the definition of intersection, $(s[A^{(p)}])^{1/p}$ represents the L_p-norm of A. The cost of the intersections to create $A^{(p)}$ is in $O(|A|^p)$. □

Note that the problem of computing the L_∞ norm of a probabilistic automaton is NP-hard [84].

1.7.3 Approximate Computation of L_p-norm

Here we consider the specific case of the computation of the L_p-norm of a probabilistic automaton. Our results can be generalized to cover more general cases, in particular the case of unambiguous automata.

Since for any $\epsilon > 0$, a probabilistic automaton is ϵ-k-closed for the probability semiring, instead of the (generalized) Floyd-Warshall algorithm, we can use a single-source shortest-distance algorithm to compute $s[A]$ as already described in Section 1.2.3. This algorithm works with any queue discipline and its space complexity is linear, which is significantly more efficient than the Floyd-Warshall algorithm. The complexity results and analyses detailed in Section 1.2.3 apply identically here.

1.7.4 Approximate Computation of Entropy

Recall that the entropy $H(A)$ of a probabilistic automaton A is defined as:

$$H(A) = -\sum_{x \in \Sigma^*} [\![A]\!](x) \log([\![A]\!](x)).$$

In Section 1.7.1, we saw that the entropy of an unambiguous probabilistic automaton can be computed efficiently using the generalized shortest-distance algorithm over the entropy semiring. However, the algorithm only works for *unambiguous* probabilistic automata. Comment 1.2 explains why the lack of ambiguity is necessary for correct computation in the semiring framework.

In this section, we show that the same shortest-distance algorithm that computes the entropy of unambiguous probabilistic automata exactly yields an approximation of the entropy of an arbitrary probabilistic automaton A, where the approximation quality is a function of the degree of ambiguity of A, made precise later. Our proofs make use of the standard log-sum inequality [36], a special case of Jensen's inequality, which holds for any positive reals $a_1, \ldots, a_k$, and $b_1, \ldots, b_k$:

$$\sum_{i=1}^{k} a_i \log \frac{a_i}{b_i} \geq \left(\sum_{i=1}^{k} a_i \right) \log \frac{\sum_{i=1}^{k} a_i}{\sum_{i=1}^{k} b_i}. \tag{1.38}$$

We first review the notion of the degree of ambiguity of a finite automaton. The *degree of ambiguity* of a string x in an automaton A, denoted by $\mathrm{da}(A, x)$, is the number of successful paths in A labeled with x.

The *degree of ambiguity* of A is defined as $\mathrm{da}(A) = \sup_{x \in \Sigma^*} \mathrm{da}(A, x)$. A is said *finitely ambiguous* if $\mathrm{da}(A) < \infty$ and *infinitely ambiguous* if $\mathrm{da}(A) = \infty$. A is called *polynomially ambiguous* with degree h if there exists an integer h such that $\mathrm{da}(A, x) \leq |x|^h$ for all $x \in \Sigma^*$. The minimal h for which this holds is called the *degree of polynomial ambiguity* of A, denoted by $\mathrm{dpa}(A)$. By definition, $\mathrm{dpa}(A) = 0$ if and only if A is finitely ambiguous. When A is infinitely ambiguous but not polynomially ambiguous, we say that A is *exponentially ambiguous* and that $\mathrm{dpa}(A) = \infty$.

Lemma 1.6 *Let A be a probabilistic automaton and let $x \in \Sigma^+$ be a string accepted by A on k paths $\pi_1, \ldots, \pi_k$. Let $w(\pi_i)$ be the probability of path π_i. Clearly, $[\![A]\!](x) = \sum_{i=1}^{k} w(\pi_i)$. Then,*

$$\sum_{i=1}^{k} w(\pi_i) \log w(\pi_i) \geq [\![A]\!](x)(\log [\![A]\!](x) - \log k).$$

Proof. The result follows straightforwardly from the log-sum inequality, with $a_i = w(\pi_i)$ and $b_i = 1$:

$$\sum_{i=1}^{k} w(\pi_i) \log w(\pi_i) \geq \left(\sum_{i=1}^{k} w(\pi_i) \right) \log \frac{\sum_{i=1}^{k} w(\pi_i)}{k} = [\![A]\!](x)(\log [\![A]\!](x) - \log k).$$

□

For a probabilistic automaton A, let $s[A]$ be the quantity computed by the generalized shortest-distance algorithm with the entropy semiring (see Equation 1.2). The discussion following Theorem 1.11 (a), together with Theo-

rem 1.11 show that if A is unambiguous, then $S[A] = H(A)$.

Theorem 1.13 *Let A be a probabilistic automaton and let L denote the expected length of strings accepted by A (i.e. $L = \sum_{x\in\Sigma^*} |x|[\![A]\!](x)$). Then,*

1. *If A is* finitely ambiguous *with degree of ambiguity k (i.e.* $\mathrm{da}(A) = k$ *for some $k \in \mathbb{N}$), then $H(A) \leq s[A] \leq H(A) + \log k$.*

2. *If A is* polynomially ambiguous *with degree of polynomial ambiguity k (i.e.* $\mathrm{dpa}(A) = k$ *for some $k \in \mathbb{N}$), then $H(A) \leq s[A] \leq H(A) + k \log L$.*

Proof. The lower bound, $s[A] \geq H(A)$ follows from the observation that for a string x that is accepted by A on k paths $\pi_1, \ldots, \pi_k$,

$$\sum_{i=1}^{k} w(\pi_i) \log(w(\pi_i)) \leq (\sum_{i=1}^{k} w(\pi_i)) \log(\sum_{i=1}^{k} w(\pi_i)).$$

Since the quantity $-\sum_{i=1}^{k} w(\pi_i) \log(w(\pi_i))$ is string x's contribution to $s[A]$ and the quantity $-(\sum_{i=1}^{k} w(\pi_i)) \log(\sum_{i=1}^{k} w(\pi_i))$ its contribution to $H(A)$, summing over all accepted strings x, we obtain $H(A) \leq s[A]$.

Assume that A is finitely ambiguous with degree of ambiguity k. Let $x \in \Sigma^*$ be a string that is accepted on $l_x \leq k$ paths $\pi_1, \ldots, \pi_{l_x}$. By Lemma 1.6, we have $\sum_{i=1}^{l_x} w(\pi_i) \log w(\pi_i) \geq [\![A]\!](x)(\log[\![A]\!](x) - \log k)$. Thus,

$$s[A] = -\sum_{x\in\Sigma^*} \sum_{i=1}^{l_x} w(\pi_i) \log w(\pi_i) \leq H(A) + \sum_{x\in\Sigma^*} (\log k)[\![A]\!](x) = H(A) + \log k.$$

This proves the first statement of the theorem.

Next, assume that A is polynomially ambiguous with degree of polynomial ambiguity k. By Lemma 1.6, we have $\sum_{i=1}^{l_x} w(\pi_i) \log w(\pi_i) \geq [\![A]\!](x)(\log [\![A]\!](x) - \log l_x) \geq [\![A]\!](x)(\log [\![A]\!](x) - \log(|x|^k))$. Thus,

$$
\begin{aligned}
s[A] &\leq H(A) + \sum_{x \in \Sigma^*} k [\![A]\!](x) \log |x| = H(A) + k \mathbb{E}_A[\log |x|] && (1.39) \\
&\leq H(A) + k \log \mathbb{E}_A[|x|] = H(A) + k \log L, && \text{(by Jensen's inequality)}
\end{aligned}
$$

which proves the second statement of the theorem. □

The quality of the approximation of the entropy of a probabilistic automaton A depends on the expected length L of an accepted string. L can be computed efficiently for an arbitrary probabilistic automaton using the *expectation semiring* and the generalized shortest-distance algorithms. The $\oplus$ and the $\otimes$ operations of the expectation semiring have exactly the same definition as those of the entropy semiring. The difference is in the initial step of the algorithm, where the initial weight of each transition in A is mapped to a pair of elements according to the mapping is $w[e] \mapsto (w[e], w[e])$.

1.8 Conclusion

In this chapter, we presented a number of results on the complexity of computing distances between probabilistic automata. On the algorithmic side, we showed that the relative entropy of two unambiguous probabilistic automata and the L_p distance for even-valued p between two arbitrary probabilistic au-

tomata can be computed in polynomial time using the semiring-framework with a generalized shortest-distance algorithm. We also showed that the our techniques can be straightforwardly generalized to compute Hellinger distance. We believe that many of these algorithmic results can be extended to finitely ambiguous probabilistic automata. We then presented a cubic time algorithm for testing the equivalence of two arbitrary probabilistic automata, significantly improving best previously reported algorithm for this problem.

We also presented a number of hardness results. We showed that it is NP-hard to compute the L_p distance between probabilistic automata for odd values of p and proved that the general problem of computing the relative entropy of probabilistic automata is (at least) P-SPACE hard. The NP-hardness result critically depends on ambiguity (or non-determinism) in the automata that is constructed in the reduction. Determining the complexity of computing the L_p distance for odd values of p for unambiguous probabilistic automata is an open problem.

We investigated the use of symmetricized relative entropy as a Kernel in machine learning and showed that, contrary to what is suggested by a number of publications, it is neither positive definite symmetric nor negative definite symmetric.

We showed that our algorithmic results can be generalized to compute arbitrary norms of a probabilistic automaton. Finally, we also examined how ambiguity (or non-determinism) influences the quality of approximation produced by the semiring-based shortest-distance algorithm. Determining the

complexity of computing the entropy of ambiguous probabilistic automata is an interesting open question. We conjecture that this problem is NP-hard.

Our results demonstrate the benefit of semiring theory for computing distances between probabilistic automata. The shortest-distance based approach yields simple but efficient algorithms, both exact and approximate, for the computation of the various distances. As shown by our experimental results, our algorithms scale to large automata with several hundred thousand transitions.

Chapter 2

Magnitude-Preserving Ranking Algorithms

2.1 Introduction and Motivation

The learning problem of ranking has gained an increasing amount of interest in the machine learning community over the last decade, in part due to the remarkable success of web search engines and recommender systems [48, 37, 60, 92, 27, 83, 1]. The recent Netflix challenge has further stimulated the learning community by fueling its research with invaluable datasets [79].

The goal of information retrieval engines is to return a set of documents, or clusters of documents, ranked in decreasing order of relevance to the user. The order may be common to all users, as with most search engines, or tuned to individuals to provide personalized search results or recommendations. The

accuracy of this ordered list is the key quality measure of these systems.

With the Netflix challenge [79], the problem is one of learning the preferences of an individual user. Previously viewed (and rated) movies by the user comprise the training set. The goal of the learning algorithm is to accurately predict the user's preference on an unseen pair of movies. The ratings of other users for a movie serve as the features for that movie. This setting is also sometimes referred to as collaborative filtering.

Another special case of the general ranking problem is the bipartite ranking problem [1]. In the bipartite ranking problem, one is given two sets, a positive set whose elements are drawn from a distribution, say D_+, and a negative set whose elements are drawn from a different distribution, say D_-. The goal is to learn a ranking function that assigns higher scores to positive examples than to negative ones. In the context of learning movie preferences of a user, bipartite ranking suggests that there are two kinds of movies: those that are liked by this user, and those that aren't, and the objective of the learning algorithm is to effectively distinguish between them. Bipartite ranking is distinct from classification in that the performance of the hypothesis is based on a cost-function that is defined over a pair of points.

In most previous research studies, the problem of ranking has been formulated as that of learning from a labeled sample of pairwise preferences a scoring function with small pairwise misranking error [48, 54, 37, 60, 83, 1]. Such a formulation of the ranking problem has several shortcomings. One point to note is that most users inspect only the top results. Thus, it would be natural

to enforce that the results returned near the top be particularly relevant and correctly ordered. The quality and ordering of the results further down the list matter less. An average pairwise misranking error directly penalizes errors at both extremes of a list more heavily than errors towards the middle of the list, since errors at the extremes result in more misranked pairs. However, one may wish to explicitly encode the requirement of ranking quality at the top in the cost function. One common solution is to weigh examples differently during training so that more important or high-quality results be assigned larger weights. This imposes higher accuracy on these examples, but does not ensure a high-quality ordering at the top. A good formulation of this problem leading to a convex optimization problem with a unique minimum is still an open question.

Another shortcoming of the pairwise misranking error is that this formulation of the problem and thus the scoring function learned ignore the magnitude of the preferences. In many applications, it is not sufficient to determine if one example is preferred to another. One may further request an assessment of how large that preference is. Taking this magnitude of preference into consideration is critical, for example in the design of search engines, which originally motivated our study, but also in other recommendation systems. For a recommendation system, one may choose to truncate the ordered list returned where a large gap in predicted preference is found. For a search engine, this may trigger a search in parallel corpora to display more relevant results.

This motivated our study of the problem of ranking while preserving the

magnitude of preferences, which we will refer to in short by *magnitude-preserving ranking*. The results of this chapter have appeared in papers [31, 33].

The problem that we are studying bears some resemblance with that of ordinal regression [67, 68, 92, 20]. It is however distinct from ordinal regression since in ordinal regression the magnitude of the difference in target values is not taken into consideration in the formulation of the problem or the solutions proposed. The algorithm of [20] does take into account the ordering of the classes by imposing that the thresholds be monotonically increasing, but this still ignores the difference of target values and thus does not follow the same objective. A crucial aspect of the algorithms we propose is that they penalize misranking errors more heavily in the case of larger magnitudes of preferences.

We describe and analyze several cost functions for this learning problem and give stability bounds for their generalization error, extending previously known stability results to non-bipartite ranking and magnitude of preference-preserving algorithms. In particular, our bounds extend the framework of [13, 14] to the case of cost functions over pairs of examples, and extend the bounds of [1] beyond the bipartite ranking problem. Our bounds also apply to algorithms optimizing the so-called *hinge rank loss*.

We present several algorithms optimizing these cost functions, and in one instance detail both a batch and an on-line version. For this algorithm, MPRank, we also show how the leave-one-out error can be computed and approximated efficiently, which can be used to determine the optimal values of the trade-off parameter in the cost function. We also report the results

of experiments comparing these algorithms on several datasets and contrast them with those obtained using RankBoost [48, 83], an algorithm designed to minimize the exponentiated loss associated with the Area Under the ROC Curve (AUC), or pairwise misranking. We also compare training times and performance results for the on-line and batch versions of MPRank, demonstrating that our on-line algorithm scales to relatively large datasets with no significant loss in accuracy.

The remainder of this chapter is organized as follows. We begin with the preliminaries in Section 2.2. Section 2.3 describes and analyzes our algorithms in detail. Section 2.4 presents stability-based generalization bounds for a family of magnitude-preserving algorithms. Section 2.5 presents the results of our experiments with these algorithms on several datasets.

2.2 Preliminaries

2.2.1 Formulation of the Problem

Let S be a sample of m labeled examples drawn in an independent and identically distributed fashion (i.i.d.) from a set X according to some distribution D:

$$(x_1, y_1), \ldots, (x_m, y_m) \in X \times \mathbb{R}.$$

Here, X is the domain from which the inputs, or instances x_i are drawn and the y_i are called labels or targets.

For any $i \in [1, m]$, we denote by S^{-i} the sample derived from S by omitting example (x_i, y_i), and by S^i the sample derived from S by replacing example (x_i, y_i) with another example (x'_i, y'_i) drawn i.i.d. from X according to D. For convenience, we will sometimes denote by $y_x = y_i$ the label of a point $x = x_i \in X$.

The quality of the ranking algorithms we consider is measured with respect to pairs of examples. Thus, a cost functions c takes as arguments two sample points. For a fixed cost function c, the empirical error $\widehat{R}(h, S)$ of a hypothesis $h : X \mapsto \mathbb{R}$ on a sample S is defined by:

$$\widehat{R}(h, S) = \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} c(h, x_i, x_j).$$

The true error $R(h)$ is defined by

$$R(h) = \mathbb{E}_{x,x' \sim D}[c(h, x, x')].$$

2.2.2 Cost Functions

As mentioned before, most ranking algorithms focus on pairwise misranking. For points $x, x' \in X$, hypothesis h misranks x and x' if and only if $(h(x') - h(x))(y_{x'} - y_x) < 0$. The pairwise misranking cost function is therefore given by:

$$c_{\text{PMR}}(h, x, x') = \begin{cases} 0, \text{ if } (h(x') - h(x))(y_{x'} - y_x) \geq 0 \\ 1, \text{ otherwise.} \end{cases}$$

We introduce several cost functions related to magnitude-preserving ranking. The first one is the so-called *hinge rank loss* which is a natural extension of the pairwise misranking loss [27, 83]. It penalizes a pairwise misranking by the magnitude of preference predicted or the nth power of that magnitude ($k = 1$ or $k = 2$):

$$c^k_{\mathrm{HR}}(h, x, x') = \begin{cases} 0, \text{ if } (h(x') - h(x))(y_{x'} - y_x) \geq 0 \\ \big|(h(x') - h(x))\big|^k, \text{ otherwise.} \end{cases}$$

c^k_{HR} does not take into consideration the true magnitude of preference $y_{x'} - y_x$ for each pair (x, x') however. The following cost function has this property and penalizes deviations of the predicted magnitude with respect to the true one. Thus, it matches our objective of magnitude-preserving ranking ($k = 1, 2$):

$$c^k_{\mathrm{MP}}(h, x, x') = \big|(h(x') - h(x)) - (y_{x'} - y_x)\big|^k.$$

A one-sided version of that cost function penalizing only misranked pairs is given by ($k = 1, 2$):

$$c^k_{\mathrm{HMP}}(h, x, x') = \begin{cases} 0, \text{ if } (h(x') - h(x))(y_{x'} - y_x) \geq 0 \\ \big|(h(x') - h(x)) - (y_{x'} - y_x)\big|^k, \text{ otherwise.} \end{cases}$$

Finally, we will consider the following cost function derived from the ϵ-insensitive cost function used in SVM regression (SVR) [102] ($k = 1, 2$):

$$c^k_{\mathrm{SVR}}(h, x, x') = \begin{cases} 0, \text{ if } \left|\left[(h(x') - h(x)) - (y_{x'} - y_x)\right]\right| \leq \epsilon \\ \left|(h(x') - h(x)) - (y_{x'} - y_x) - \epsilon\right|^k, \text{ otherwise.} \end{cases}$$

Note that with the exception of c_{PMR}, all of these cost functions are convex functions of $h(x)$ and $h(x')$.

2.2.3 Kernels and Regularization

In this section, we present technical definitions that are necessary for the presentation of our results. Some of these definitions are standard in machine learning, but we describe them here for the sake of completeness. A good reference for this material is [89].

A kernel $k : X \times X \mapsto \mathbb{R}$ can be thought of as a function that measures the similarity between a pair of points in the input space X. It is standard to assume that this similarity is symmetric (i.e. $k(x_1, x_2) = k(x_2, x_1)$). Kernels correspond to dot products in a feature space F via a map $\Phi : X \mapsto F$. Thus, for $x_1, x_2 \in X$,

$$K(x_1, x_2) = \langle \Phi(x_1), \Phi(x_2) \rangle.$$

Note that we place no restrictions on the set X, and it may contain an arbitrary set of discrete objects, such as graphs or strings.

Definition 2.1 (Gram Matrix) *Given a sample* $S = \{x_1, \ldots, x_m\} \subseteq X$,

and a function $k : X \times X \mapsto \mathbb{R}$*, the* $m \times m$ *matrix* K *with elements* $K_{ij} = K(x_i, x_j)$ *is called the* Gram Matrix *with respect to* S*.*

Note that when the kernel function is symmetric, the Gram matrix is symmetric. We will make this assumption for the rest of this chapter.

Definition 2.2 (Positive Semi-Definite Symmetric Matrix) *A real* $m \times m$ *symmetric matrix* K *satisfying*

$$\sum_{i,j=1}^{n} c_i c_j K(x_i, x_j) \geq 0. \tag{2.1}$$

for all $c_i \in \mathbb{R}$ *is called* positive definite *symmetric.*

The left hand side of Equation 2.1 is often referred to as the quadratic form induced by K. Note that if $\mathbf{c}$ is an m-dimensional vector whose ith component is c_i, and $\mathbf{K}$ is an $m \times m$-dimensional positive definite symmetric matrix, then the condition in Equation 2.1 can equivalently be written as $\mathbf{c}^\top \mathbf{K}\mathbf{c} \geq 0$. Thus, a matrix is positive definite symmetric if and only if all of its eigenvalues are non-negative.

Definition 2.3 (Positive Definite Symmetric Kernel) *Let* X *be a non-empty set. A symmetric function* k *on* $X \times X$ *which for all* $m \in \mathbb{N}$ *and* $x_1, \ldots, x_m \in X$ *gives rise to a positive semi-definite symmetric Gram matrix is called a positive definite symmetric kernel. We refer to it simply as a kernel.*

Kernels allow non-linear mappings from the input space to a high-dimensional feature space $\Phi : X \mapsto F$. The key advantages of using kernels are efficiency

and flexibility. The kernel function $k(\cdot, \cdot)$ provides an efficient way to compute dot products in a high-dimensional feature space, without explicit knowledge of Φ. This is essential when the feature space F has a very large (or even infinite) dimension. The kernel function $k(\cdot, \cdot)$ can be chosen arbitrarily so long as the existence of Φ is guaranteed. If the set of points X is discrete, then the existence of Φ is guaranteed if the induced Gram matrix is positive semi-definite symmetric.

Minimizing the training error $\widehat{R}(h)$ alone has the potential risk of learning a hypothesis that over-fits the training sample. This results in a hypothesis with a bad generalization error $R(h)$. One way to avoid this problem is to restrict the class of admissible solutions, for instance to a compact set. This technique has been applied to learning problems with great success. When the hypothesis is a linear function in some high-dimensional feature (Hilbert) space (i.e. $h(x) = w \cdot \Phi(x)$), where $w \in \mathbb{R}^N$ (as is the case with support vector machines), this restriction is achieved by directly penalizing the weight vector w in the feature space [102]. This is achieved by adding a term corresponding to $\|w\|_2^2$ together with the training error in the objective function minimized by the objective function.

2.3 Algorithms

2.3.1 Objective Functions

The regularization algorithms based on the cost functions c^k_{MP} and c^k_{SVR} correspond closely to the idea of preserving the magnitude of preferences since these cost functions penalize deviations of a predicted difference of score from the target preferences. We will refer by MPRank to the algorithm minimizing the regularization-based objective function based on c^k_{MP}:

$$F(h, S) = \|h\|^2_K + C\frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} c^k_{\mathrm{MP}}(h, x_i, x_j),$$

and by SVRank to the one based on the cost function c^k_{SVR}

$$F(h, S) = \|h\|^2_K + C\frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} c^k_{\mathrm{SVR}}(h, x_i, x_j).$$

The objective function $F(h, S)$ is minimized over $h \in H$. As we will see, hypothesis set H we are considering is that of linear functions h. That is, for all $x \in X$, $h(x) = w \cdot \Phi(x)$. In this case, $\|h\|^2_K = \|w\|^2$.

For a fixed k, $k = 1, 2$, the same stability bounds hold for both algorithms as seen in the following section. However, their time complexity is significantly different.

Comment 2.1 *We note that minimizing pairwise misranking* $c_{\mathrm{PMR}}(\cdot)$ *alone over a given set of points is a difficult problem. The number of disagreements*

between hypothesis h and the true ranking for a set of points $\{x_1, \ldots, x_m\}$ is $\sum_{i \neq j} c_{\text{PMR}}(h, x_i, x_j)$. This quantity is also called Kemeny distance *and can be thought of as the bubble sort distance between the permutations induced by $(y_{x_i})_{i \in [m]}$ and $(h(x_i))_{i \in [m]}$. Minimizing the Kemeny distance between two permutations is NP-hard.*

2.3.2 MPRank

We will examine the algorithm with squared-loss function, $c^2_{\text{MP}}(\cdot)$. Let $\Phi : X \mapsto F$ be the mapping from X to the high-dimensional feature space F. The hypothesis set H that we are considering is that of linear functions h in a high-dimensional feature space, that is $\forall x \in X, h(x) = w \cdot \Phi(x)$. The objective function can be expressed as follows

$$\begin{aligned} F(h, S) &= \|w\|^2 + C \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} \left[(w \cdot \Phi(x_j) - w \cdot \Phi(x_i)) - (y_j - y_i) \right]^2 \\ &= \|w\|^2 + \frac{2C}{m} \sum_{i=1}^{m} \|w \cdot \Phi(x_i) - y_i\|^2 - 2C \|w \cdot \bar{\Phi} - \bar{y}\|^2, \end{aligned}$$

where $\bar{\Phi} = \frac{1}{m} \sum_{i=1}^{m} \Phi(x_i)$ and $\bar{y} = \frac{1}{m} \sum_{i=1}^{m} y_i$. The objective function can thus be written with a single sum over the training examples, which results in a more efficient computation of the solution.

Let N be the dimension of the feature space F. For $i = 1, \ldots, m$, let $\mathbf{M}_{x_i} \in \mathbb{R}^{N \times 1}$ denote the column matrix representing $\Phi(x_i)$, $\mathbf{M}_{\bar{\Phi}} \in \mathbb{R}^{N \times 1}$ a column matrix representing $\bar{\Phi}$, $\mathbf{W} \in \mathbb{R}^{N \times 1}$ a column matrix representing the vector

w, $\mathbf{M}_Y \in \mathbb{R}^{m\times 1}$ a column matrix whose ith component is y_i, and $\mathbf{M}_{\bar{Y}} \in \mathbb{R}^{m\times 1}$ a column matrix with all its components equal to $\bar{y}$. Let $\mathbf{M}_X, \mathbf{M}_{\bar{X}} \in \mathbb{R}^{N\times m}$ be the matrices defined by:

$$\mathbf{M}_X = [\mathbf{M}_{x_1} \dots \mathbf{M}_{x_m}] \quad \mathbf{M}_X = [\mathbf{M}_{\bar{\Phi}} \dots \mathbf{M}_{\bar{\Phi}}].$$

Then, the expression giving F can be rewritten as

$$F = \|\mathbf{W}\|^2 + \frac{2C}{m}\|\mathbf{M}_X^\top \mathbf{W} - \mathbf{M}_Y\|^2 - \frac{2C}{m}\|\mathbf{M}_{\bar{X}}^\top \mathbf{W} - \mathbf{M}_{\bar{Y}}\|^2.$$

The gradient of F is then given by:

$$\nabla F = 2\mathbf{W} + \frac{4C}{m}\mathbf{M}_X(\mathbf{M}_X^\top \mathbf{W} - \mathbf{M}_Y) - \frac{4C}{m}\mathbf{M}_{\bar{X}}(\mathbf{M}_{\bar{X}}^\top \mathbf{W} - \mathbf{M}_{\bar{Y}}).$$

Setting $\nabla F = 0$ yields:

$$\mathbf{W}\left(\mathbf{I} + C'(\mathbf{M}_X\mathbf{M}_X^\top - \mathbf{M}_{\bar{X}}\mathbf{M}_{\bar{X}}^\top)\right) = C'(\mathbf{M}_X\mathbf{M}_Y - \mathbf{M}_{\bar{X}}\mathbf{M}_{\bar{Y}}),$$

where $C' = \frac{2C}{m}$. Upon simplification, we obtain:

$$\mathbf{W} = C'\big(\mathbf{I} + C'(\mathbf{M}_X - \mathbf{M}_{\bar{X}})(\mathbf{M}_X - \mathbf{M}_{\bar{X}})^\top\big)^{-1}(\mathbf{M}_X - \mathbf{M}_{\bar{X}})(\mathbf{M}_Y - \mathbf{M}_{\bar{Y}}), \quad (2.2)$$

Here, we are using the identities $\mathbf{M}_X\mathbf{M}_X^\top - \mathbf{M}_{\bar{X}}\mathbf{M}_{\bar{X}}^\top = (\mathbf{M}_X - \mathbf{M}_{\bar{X}})(\mathbf{M}_X - \mathbf{M}_{\bar{X}})^\top$ and $\mathbf{M}_X\mathbf{M}_Y - \mathbf{M}_{\bar{X}}\mathbf{M}_{\bar{Y}} = (\mathbf{M}_X - \mathbf{M}_{\bar{X}})(\mathbf{M}_Y - \mathbf{M}_{\bar{Y}})$, which are not hard to verify. This provides the solution of the primal problem. Since matrices

$(\mathbf{I} + C'(\mathbf{M}_X - \mathbf{M}_{\bar{X}})(\mathbf{M}_X - \mathbf{M}_{\bar{X}})^\top)^{-1}$ and $\mathbf{M}_X - \mathbf{M}_{\bar{X}}$ are symmetric, they commute and this leads to:

$$\mathbf{W} = C'(\mathbf{M}_X - \mathbf{M}_{\bar{X}})\big(\mathbf{I} + C'(\mathbf{M}_X - \mathbf{M}_{\bar{X}})(\mathbf{M}_X - \mathbf{M}_{\bar{X}})^\top\big)^{-1}(\mathbf{M}_Y - \mathbf{M}_{\bar{Y}}).$$

This expression for $\mathbf{W}$ can be used to derive the solution in the dual, which can then be kernelized (i.e. points in the input space appear only in dot products, see "Kernel trick" in [88]). For any $x' \in X$ (see [35] for a similar derivation of the dual),

$$h(x') = C'\mathbf{K}'(\mathbf{I} + \bar{\mathbf{K}})^{-1}(\mathbf{M}_Y - \mathbf{M}_{\bar{Y}}), \tag{2.3}$$

where $\mathbf{K}' \in \mathbb{R}^{1\times m}$ is the row matrix whose jth component is

$$K(x', x_j) - \frac{1}{m}\sum_{k=1}^{m} K(x', x_k)$$

and $\bar{\mathbf{K}}$ is the kernel matrix defined by

$$\frac{1}{C'}(\bar{\mathbf{K}})_{ij} = K(x_i, x_j) - \frac{1}{m}\sum_{k=1}^{m}(K(x_i, x_k) + K(x_j, x_k)) + \frac{1}{m^2}\sum_{k=1}^{m}\sum_{l=1}^{m} K(x_k, x_l),$$

for all $i, j \in [1, m]$. The solution of the optimization problem for MPRank is close to that of a kernel ridge regression [53] problem, but the presence of additional terms makes it distinct, a fact that can also be confirmed experimentally. However, remarkably, it has the same computational complexity, due to the fact that the optimization problem can be written in terms of a

single sum, as already pointed out above. The main computational cost of the algorithm is that of the matrix inversion, which can be computed in time $O(N^3)$ in the primal, and $O(m^3)$ in the dual case, or $O(N^{2+\alpha})$ and $O(m^{2+\alpha})$, with $\alpha \approx .376$, using faster matrix inversion methods such as that of Coppersmith and Winograd.

2.3.3 SVRank

We will examine the algorithm with the norm-one loss, $c^1_{\mathrm{MP}}(h, x, x')$. As with MPRank, the hypothesis set H that we are considering here is that of linear functions h, that is $\forall x \in X, h(x) = w \cdot \Phi(x)$. The constraint optimization problem associated with SVRank can thus be rewritten as

$$\text{minimize} \quad F(h, S) = \|w\|^2 + C\frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m}(\xi_{ij} + \xi^*_{ij})$$

$$\text{subject to} \begin{cases} w \cdot (\Phi(x_j) - \Phi(x_i)) - (y_j - y_i) \leq \epsilon + \xi_{ij} \\ (y_j - y_i) - w \cdot (\Phi(x_j) - \Phi(x_i)) \leq \epsilon + \xi^*_{ij} \\ \xi_{ij}, \xi^*_{ij} \geq 0, \end{cases}$$

for all $i, j \in [1, m]$. Note that the number of constraints are quadratic with respect to the number of examples. Thus, in general, this results in a problem that is more costly to solve than that of MPRank.

Introducing Lagrange multipliers $\alpha_{ij}, \alpha^*_{ij} \geq 0$, corresponding to the first two sets of constraints and $\beta_{ij}, \beta^*_{ij} \geq 0$ for the remaining constraints leads to

the following Lagrange function

$$\begin{aligned} L = \quad & \|w\|^2 + C\tfrac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m}(\xi_{ij}+\xi_{ij}^*)+ \\ & \sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_{ij}(w\cdot(\Phi(x_j)-\Phi(x_i))-(y_j-y_i)-\epsilon+\xi_{ij})+ \\ & \sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_{ij}^*(-w\cdot(\Phi(x_j)-\Phi(x_i))+(y_j-y_i)-\epsilon+\xi_{ij}^*)+ \\ & \sum_{i=1}^{m}\sum_{j=1}^{m}(\beta_{ij}\xi_{ij}+\beta_{ij}^*\xi_{ij}^*). \end{aligned}$$

Taking the gradients, setting them to zero, and applying the Karush-Kuhn-Tucker conditions leads to the following dual maximization problem

$$\begin{aligned} \text{maximize } & \frac{1}{2}\sum_{i,j=1}^{m}\sum_{k,l=1}^{m}(\alpha_{ij}^*-\alpha_{ij})(\alpha_{kl}^*-\alpha_{kl})K_{ij,kl}- \\ & \epsilon\sum_{i,j=1}^{m}(\alpha_{ij}^*-\alpha_{ij})+\sum_{i,j=1}^{m}(\alpha_{ij}^*-\alpha_{ij})(y_j-y_i) \\ \text{subject to } & 0\le\alpha_{ij},\alpha_{ij}^*\le C,\forall i,j\in[1,m], \end{aligned}$$

where $K_{ij,kl} = K(x_i,x_k)+K(x_j,x_l)-K(x_i,x_l)-K(x_j,x_k)$. This quadratic optimization problem can be solved in a way similar to SVM regression (SVR) [102] by defining a kernel K' over pairs with $K'((x_i,x_j),(x_k,x_l)) = K_{ij,kl}$, for all $i,j,k,l\in[1,m]$, and associating the target value y_i-y_j to the pair (x_i,x_j).

The computational complexity of the quadratic programming with respect to pairs makes this algorithm less attractive for relatively large samples.

2.3.4 On-line Version of MPRank

Recall from Section 2.3.2 that the cost function for MPRank can be written as

$$F(h,S) = \|w\|^2 + \frac{2C}{m}\sum_{i=1}^{m}\left[(w\cdot\Phi(x_i)-y_i)^2 - (w\cdot\bar{\Phi}-\bar{y})^2\right]. \tag{2.4}$$

This expression suggests that the solution w can be found by solving the following optimization problem

$$\begin{aligned}&\underset{w}{\text{minimize}} \quad F = \|w\|^2 + \frac{2C}{m}\sum_{i=1}^{m}\xi_i^2\\ &\text{subject to } (w\cdot\Phi(x_i)-y_i) - (w\cdot\bar{\Phi}-\bar{y}) = \xi_i \text{ for } i=1,\ldots,m\end{aligned}$$

Introducing the Lagrange multipliers β_i corresponding to the ith equality constraint leads to the following Lagrange function:

$$L(w,\xi,\beta) = \|w\|^2 + \frac{2C}{m}\sum_{i=1}^{m}\xi_i^2 - \sum_{i=1}^{m}\beta_i\left((w\cdot\Phi(x_i)-y_i) - (w\cdot\bar{\Phi}-\bar{y}) - \xi_i\right)$$

Setting $\partial L/\partial w = 0$, we obtain $w = \frac{1}{2}\sum_{i=1}^{m}\beta_i(\Phi(x_i)-\bar{\Phi})$, and setting $\partial L/\partial\xi_i = 0$ leads to $\xi_i = -\frac{m}{4C}\beta_i$. Substituting these expressions back in and letting $\alpha_i = \beta_i/2$ results in the optimization problem

$$\underset{\alpha_i}{\text{maximize}} \qquad -\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j\widetilde{K}(x_i,x_j) - \frac{m}{2C}\sum_{i=1}^{m}\alpha_i^2 + 2\sum_{i=1}^{m}\alpha_i\widetilde{y}_i, \tag{2.5}$$

where $\widetilde{K}(x_i, x_j) = K(x_i, x_j) - \frac{1}{m}\sum_{k=1}^{m}(K(x_i, x_k) + K(x_j, x_k)) + \frac{1}{m^2}\sum_{k,l=1}^{m} K(x_k, x_l)$ and $\widetilde{y_i} = y_i - \bar{y}$.

Based on the expressions for the partial derivatives of the Lagrange function, we can now describe a gradient descent algorithm that avoids the prohibitive complexity of MPRank that is associated with matrix inversion:

1 **for** $i \leftarrow 1$ **to** m **do** $\alpha_i \leftarrow 0$

2 **repeat**

3 **for** $i \leftarrow 1$ **to** m

4 **do** $\alpha_i \leftarrow \alpha_i + \eta\big[2(\widetilde{y_i} - \sum_{j=1}^{m} \alpha_j \widetilde{K}(x_i, x_j)) - \frac{m}{C}\alpha_i\big]$

5 **until** convergence

The gradient descent algorithm described above can be straightforwardly modified to an on-line algorithm where points in the training set are processed in T passes, one by one, and the complexity of updates for the ith point is $O(m)$ leading to an overall complexity of $O(T \cdot m^2)$. Note that even using the best matrix inversion algorithms, one only achieves a time complexity of $O(m^{2+\alpha})$, with $\alpha \approx .376$. In addition to a favorable complexity if $T = o(m^{.376})$, an appealing aspect of the gradient descent based algorithms is their simplicity. They are quite efficient in practice for datasets with a large number of training points.

2.3.5 Leave-One-Out Analysis for MPRank

The leave-one-out error of a learning algorithm is typically costly to compute in general as it requires training the algorithm on m subsamples of the original sample. This section shows how the leave-one-out error of MPRank can be computed and approximated efficiently by extending the techniques of [103].

The standard definition of the leave-one-out error holds for errors or cost functions defined over a single point. The definition can be extended to cost functions defined over pairs by each time leaving out one pair of points $(x_i, x_j), i \neq j$, instead of a single point.

To simplify the notation, we will denote $h_{S-\{x_i,x_j\}}$ by h_{ij} and $h_{S-\{x,x'\}}$ by $h_{xx'}$. The leave-one-out error of an algorithm L over a sample S returning the hypothesis h_{ij} for a training sample $S - \{x_i, x_j\}$ is then defined by

$$\mathrm{LOO}(L, S) = \frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{j=1, i \neq j}^{m} c(h_{ij}, x_i, x_j). \tag{2.6}$$

The following proposition shows that with our new definition, the fundamental property of LOO is preserved.

Proposition 2.1 *Let $m \geq 2$ and let h' be the hypothesis returned by L when trained over a sample S' of size $m - 2$. Then, the leave-one-out error over a sample S of size m is an unbiased estimate of the true error over a sample of size $m - 2$:*

$$\mathbb{E}_{S \sim D}[\mathrm{LOO}(L, S)] = R(h'),$$

Proof. Since all points of S are drawn i.i.d. and according to the same distribution D,

$$\begin{aligned}
\mathbb{E}_{S\sim D}[\mathrm{LOO}(L,S)] &= \frac{1}{m(m-1)}\sum_{i,j=1,j\neq i}^{m}\mathbb{E}_{S\sim D}[c(h_{ij},x_i,x_j)] \\
&= \frac{1}{m(m-1)}\sum_{x,x'\in S,x\neq x'}^{m}\mathbb{E}_{S\sim D,x,x'\in S}[c(h_{xx'},x,x')] \\
&= \mathbb{E}_{S\sim D,x,x'\in S}[c(h_{xx'},x,x')]
\end{aligned}$$

This last term coincides with $\mathbb{E}_{S',x,x'\sim D,|S'|=m-2}[c(h_{xx'},x,x')] = R(h')$. □

In Section 2.3.2, it was shown that the hypothesis returned by MPRank for a sample S is given by $h(x') = C'\mathbf{K}'(\mathbf{I}+\bar{\mathbf{K}})^{-1}(\mathbf{M}_Y - \mathbf{M}_{\bar{Y}})$ for all $x' \in \mathbf{M}_X$. Let $\mathbf{K}_c$ be the matrix derived from $\mathbf{K}$ by replacing each entry $\mathbf{K}_{ij}$ of $\mathbf{K}$ by the sum of the entries in the same column $\sum_{j=1}^{m}\mathbf{K}_{ij}$. Similarly, let $\mathbf{K}_r$ be the matrix derived from $\mathbf{K}$ by replacing each entry of $\mathbf{K}$ by the sum of the entries in the same row, and let $\mathbf{K}_{rc}$ be the matrix whose entries all are equal to the sum of all entries of $\mathbf{K}$. Note that the matrix $\bar{\mathbf{K}}$ can be written as:

$$\frac{1}{C'}\bar{\mathbf{K}} = \mathbf{K} - \frac{1}{m}(\mathbf{K}_r + \mathbf{K}_c) + \frac{1}{m^2}\mathbf{K}_{rc}.$$

Let $\mathbf{K}''$ and $\mathbf{U}$ be the matrices defined by:

$$\mathbf{K}'' = \mathbf{K} - \frac{1}{m}\mathbf{K}_r \quad \text{and} \quad \mathbf{U} = C'\mathbf{K}''(\mathbf{I}+\bar{\mathbf{K}})^{-1}. \tag{2.7}$$

Then, for all $i \in [1,m]$, $h(x_i) = \sum_{k=1}^{m}\mathbf{U}_{ik}(y_k - \bar{y})$. In the remainder of this

section, we will consider the particular case of the $k = 2$ cost function for MPRank, $c^2_{\mathrm{MP}}(h, x_i, x_j) = [(h(x_j) - y_j) - (h(x_i) - y_i)]^2$.

Proposition 2.2 *Let h' be the hypothesis returned by MPRank when trained on $S - \{x_i, x_j\}$ and let $\bar{h}' = \frac{1}{m}\sum_{k=1}^m h'(x_k)$. For all $i, j \in [1, m]$, let $\mathbf{V}_{ij} = \mathbf{U}_{ij} - \frac{1}{m-2}\sum_{k \notin \{i,j\}} \mathbf{U}_{ik}$. Then, the following identity holds for $c^2_{\mathrm{MP}}(h', x_i, x_j)$.*

$$\begin{aligned}\big[(1 - \mathbf{V}_{jj})(1 - \mathbf{V}_{ii}) - \mathbf{V}_{ij}\mathbf{V}_{ji}\big]^2 c^2_{\mathrm{MP}}(h', x_i, x_j) = \\ \big[(1 - \mathbf{V}_{ii} - \mathbf{V}_{ij})(h(x_j) - y_j) - (1 - \mathbf{V}_{ji} - \mathbf{V}_{jj})(h(x_i) - y_i) \\ -[(1 - \mathbf{V}_{ii} - \mathbf{V}_{ij})(\mathbf{V}_{jj} + \mathbf{V}_{ji})(1 - \mathbf{V}_{ji} - \mathbf{V}_{jj})(\mathbf{V}_{ii} + \mathbf{V}_{ij})](\bar{h}' - \bar{y})\big]^2 \quad .\end{aligned} \tag{2.8}$$

Proof. By Equation 2.4, the cost function of MPRank can be written as:

$$F = \|w\|^2 + \frac{2C}{m}\sum_{k=1}^m \big[(h(x_k) - y_k) - (\bar{h} - \bar{y})\big]^2,$$

where $\bar{h} = \frac{1}{m}\sum_{k=1}^m h(x_k)$. h' is the solution of the minimization of F when the terms corresponding to x_i and x_j are left out. Equivalently, one can keep these terms but select new values for y_i and y_j to ensure that these terms are zero. Proceeding this way, the new values y'_i and y'_j must verify the following:

$$h'(x_i) - y'_i = h'(x_j) - y'_j = \bar{h}' - \bar{y}',$$

with $\bar{y}' = \frac{1}{m}[y'(x_i) + y'(x_j) + \sum_{k \notin \{i,j\}} y_k]$. Thus, by Equation 2.7, $h'(x_i)$ is

given by $h'(x_i) = \sum_{k=1}^{m} \mathbf{U}_{ik}(y_k - \overline{y}')$. Therefore,

$$\begin{aligned}
h'(x_i) - y_i &= \sum_{k \notin \{i,j\}} \mathbf{U}_{ik}(y_k - \overline{y}') + \mathbf{U}_{ii}(y_i' - \overline{y}') + \mathbf{U}_{ij}(y_j' - \overline{y}') - y_i \\
&= \sum_{k \notin \{i,j\}} \mathbf{U}_{ik}(y_k - \overline{y}) - \sum_{k \notin \{i,j\}} \mathbf{U}_{ik}(\overline{y}' - \overline{y}) + \mathbf{U}_{ii}(h'(x_i) - \bar{h}') \\
&\quad + \mathbf{U}_{ij}(h'(x_j) - \bar{h}') - y_i \\
&= (h(x_i) - y_i) - \mathbf{U}_{ii}(y_i - \bar{y}) - \mathbf{U}_{ij}(y_j - \bar{y}) - \sum_{k \notin \{i,j\}} \mathbf{U}_{ik}(\overline{y}' - \overline{y}) \\
&\quad + \mathbf{U}_{ii}(h'(x_i) - \bar{h}') + \mathbf{U}_{ij}(h'(x_j) - \bar{h}') \\
&= (h(x_i) - y_i) + \mathbf{U}_{ii}(h'(x_i) - y_i) + \mathbf{U}_{ij}(h'(x_j) - y_j) - (\mathbf{U}_{ii} + \mathbf{U}_{ij})(\bar{h}' - \bar{y}) \\
&\quad - \sum_{k \notin \{i,j\}} \mathbf{U}_{ik}(\overline{y}' - \overline{y}) \\
&= (h(x_i) - y_i) + \mathbf{U}_{ii}(h'(x_i) - y_i) + \mathbf{U}_{ij}(h'(x_j) - y_j) - (\mathbf{U}_{ii} + \mathbf{U}_{ij})(\bar{h}' - \bar{y}) \\
&\quad - \sum_{k \notin \{i,j\}} \mathbf{U}_{ik} \frac{1}{m-2}[(h'(x_i) - y_i) + (h'(x_j) - y_j) - 2(\bar{h}' - \bar{y})] \\
&= (h(x_i) - y_i) + \mathbf{V}_{ii}(h'(x_i) - y_i) + \mathbf{V}_{ij}(h'(x_j) - y_j) - (\mathbf{V}_{ii} + \mathbf{V}_{ij})(\bar{h}' - \bar{y}).
\end{aligned}$$

Thus,

$$(1 - \mathbf{V}_{ii})(h'(x_i) - y_i) - \mathbf{V}_{ij}(h'(x_j) - y_j) = (h(x_i) - y_i) - (\mathbf{V}_{ii} + \mathbf{V}_{ij})(\bar{h}' - \bar{y}),$$

Similarly, we have

$$-\mathbf{V}_{ji}(h'(x_i) - y_i) + (1 - \mathbf{V}_{jj})(h'(x_j) - y_j) = (h(x_j) - y_j) - (\mathbf{V}_{jj} + \mathbf{V}_{ji})(\bar{h}' - \bar{y}).$$

Solving the linear system formed by these two equations with unknown vari-

ables $(h'(x_i) - y_i)$ and $(h'(x_j) - y_j)$ gives:

$$\big[(1-\mathbf{V}_{jj})(1-\mathbf{V}_{ii}) - \mathbf{V}_{ij}\mathbf{V}_{ji}\big](h'(x_i) - y_i) = (1-\mathbf{V}_{jj})(h(x_i) - y_i) + \mathbf{V}_{ij}(h(x_j) - y_j)$$
$$-[(\mathbf{V}_{ii} + \mathbf{V}_{ij})(1-\mathbf{V}_{jj}) + (\mathbf{V}_{jj} + \mathbf{V}_{ji})\mathbf{V}_{ij}](\bar{h'} - \bar{y}).$$

Similarly, we obtain:

$$\big[(1-\mathbf{V}_{jj})(1-\mathbf{V}_{ii}) - \mathbf{V}_{ij}\mathbf{V}_{ji}\big](h'(x_j) - y_j) = \mathbf{V}_{ji}(h(x_i) - y_i) + (1-\mathbf{V}_{ii})(h(x_j) - y_j)$$
$$-[(\mathbf{V}_{ii} + \mathbf{V}_{ij})\mathbf{V}_{ji} + (\mathbf{V}_{jj} + \mathbf{V}_{ji})(1-\mathbf{V}_{ii})](\bar{h'} - \bar{y}).$$

Taking the difference of these last two equations and squaring both sides yields the expression of $c^2_{\mathrm{MP}}(h', x_i, x_j)$ given in the statement of the proposition. □

Given $\bar{h'}$, Proposition 2.2 and Equation 2.6 can be used to compute the leave-one-out error of h efficiently, since the coefficients $\mathbf{U}_{ij}$ can be obtained in time $O(m^2)$ from the matrix $(\mathbf{I} + \bar{\mathbf{K}})^{-1}$ already computed to determine h.

Note that by the results of Section 2.3.2 and the strict convexity of the objective function, h' is uniquely determined and has a closed form. Thus, unless the points x_i and x_j coincide, the expression $[(1-\mathbf{V}_{jj})(1-\mathbf{V}_{ii}) - \mathbf{V}_{ij}\mathbf{V}_{ji}\big]$ factor of $c^2_{\mathrm{MP}}(h', x_i, x_j)$ cannot be null. Otherwise, the system of linear equations found in the proof is reduced to a single equation and $h'(x_i)$ (or $h'(x_j)$) is not uniquely specified.

For larger values of m, the average value of h over the sample S should not be much different from that of h', thus we can approximate $\bar{h'}$ by $\bar{h}$. Using this

approximation, for a sample with distinct points, we can write for L =MPRank

$$\mathrm{LOO}(L,S) \approx \frac{1}{m(m-1)} \sum_{i \neq j} \left[\frac{(1-\mathbf{V}_{ii}-\mathbf{V}_{ij})(h(x_j)-y_j) - (1-\mathbf{V}_{ji}-\mathbf{V}_{jj})(h(x_i)-y_i)}{\big[(1-\mathbf{V}_{jj})(1-\mathbf{V}_{ii}) - \mathbf{V}_{ij}\mathbf{V}_{ji}\big]} \right.$$
$$\left. - \frac{\big[(1-\mathbf{V}_{ii}-\mathbf{V}_{ij})(\mathbf{V}_{jj}+\mathbf{V}_{ji}) - (1-\mathbf{V}_{ji}-\mathbf{V}_{jj})(\mathbf{V}_{ii}+\mathbf{V}_{ij})\big]}{\big[(1-\mathbf{V}_{jj})(1-\mathbf{V}_{ii}) - \mathbf{V}_{ij}\mathbf{V}_{ji}\big]} (\bar{h}-\bar{y}) \right]^2 .$$

This can be used to determine the best value of the parameter C efficiently based on the leave-one-out error.

Observe that the sum of the entries of each row of $\bar{\mathbf{K}}$ or each row of $\mathbf{K}''$ is zero. Let $\mathbf{M}_1 \in \mathbb{R}^{m\times 1}$ be column matrix with all entries equal to 1. In view of this observation, $\bar{\mathbf{K}}\mathbf{M}_1 = 0$, thus $(\mathbf{I}+\bar{\mathbf{K}})\mathbf{M}_1 = \mathbf{M}_1$, $(\mathbf{I}+\bar{\mathbf{K}})^{-1}\mathbf{M}_1 = \mathbf{M}_1$, and $\mathbf{U}\mathbf{M}_1 = C'\mathbf{K}''(\mathbf{I}+\bar{\mathbf{K}})^{-1}\mathbf{M}_1 = C'\mathbf{K}''\mathbf{M}_1 = 0$. This shows that the sum of the entries of each row of $\mathbf{U}$ is also zero, which yields the following identity for the matrix $\mathbf{V}$:

$$\mathbf{V}_{ij} = \mathbf{U}_{ij} - \frac{1}{m-2} \sum_{k \notin \{i,j\}} \mathbf{U}_{ik} = \mathbf{U}_{ij} + \frac{\mathbf{U}_{ii}+\mathbf{U}_{ij}}{m-2} = \frac{(m-1)\mathbf{U}_{ij}+\mathbf{U}_{ii}}{m-2}.$$

Hence the matrix $\mathbf{V}$ computes

$$\sum_{k=1}^{m} \mathbf{V}_{ik}(y_k-\bar{y}) = \sum_{k=1}^{m} \mathbf{V}_{ik}(y_k-\bar{y}) = \frac{m-1}{m-2} h(x_i).$$

These identities further simplify the expression of matrix $\mathbf{V}$ and its relationship with h.

2.4 Stability bounds

In [13] and [14], Bousquet and Elisseeff gave stability bounds for several regression and classification algorithms. This section shows similar stability bounds for ranking and magnitude-preserving ranking algorithms. This also generalizes the results of [1] which were given in the specific case of bipartite ranking.

The following definitions are natural extensions, to the case of cost functions over pairs, of those given by [14].

Definition 2.4 *A learning algorithm L is said to be uniformly β-stable with respect to the sample S and cost function c if there exists $\beta \geq 0$ such that for all $S \in (X \times \mathbb{R})^m$ and $i \in [1, m]$,*

$$\forall x, x' \in X,\ |c(h_S, x, x') - c(h_{S^{-i}}, x, x')| \leq \beta.$$

Definition 2.5 *A cost function c is is σ-admissible with respect to a hypothesis set H if there exists $\sigma \geq 0$ such that for all $h, h' \in H$, and for all $x, x' \in X$,*

$$|c(h, x, x') - c(h', x, x')| \leq \sigma(|\Delta h(x')| + |\Delta h(x)|),$$

with $\Delta h = h' - h$.

2.4.1 Magnitude-preserving regularization algorithms

For a cost function c such as those just defined and a regularization function N, a regularization-based algorithm can be defined as one minimizing the following objective function:

$$F(h, S) = N(h) + C\frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} c(h, x_i, x_j),$$

where $C \geq 0$ is a constant determining the trade-off between the emphasis on the regularization term versus the error term. In much of what follows, we will consider the case where the hypothesis set H is a reproducing Hilbert space and where N is the squared norm in a that space, $N(h) = \|h\|_K^2$ for a kernel K, though some of our results can straightforwardly be generalized to the case of an arbitrary convex N. By the reproducing property, for any $h \in H$, $\forall x \in X,\ h(x) = \langle h, K(x, .)\rangle$ and by Cauchy-Schwarz's inequality,

$$\forall x \in X, |h(x)| \leq \|h\|_K \sqrt{K(x, x)}.$$

Assuming that for all $x \in X, K(x, x) \leq \kappa^2$ for some constant $\kappa \geq 0$, the inequality becomes: $\forall x \in X, |h(x)| \leq \kappa\|h\|_K$. With the cost functions previously discussed, the objective function F is then strictly convex and the optimization problem admits a unique solution. In what follows, we will refer to the algorithms minimizing the objective function F with a cost function defined in the previous section as *magnitude-preserving regularization algorithms*.

Lemma 2.1 *Assume that the hypotheses in H are bounded, that is for all $h \in H$ and $x \in S$, $|h(x) - y_x| \leq M$. Then, the cost functions c^k_{HR}, c^k_{MP}, c^k_{HMP}, and c^k_{SVR} are all σ_n-admissible with $\sigma_1 = 1$, $\sigma_2 = 4M$.*

Proof. We will give the proof in the case of c^k_{MP}, $k = 1, 2$, the other cases can be treated similarly.

By definition of c^1_{MP}, for all $x, x' \in X$,

$$\begin{aligned} |c^1_{\mathrm{MP}}(h', x, x') - c^1_{\mathrm{MP}}(h, x, x')| &= \big|\,|(h'(x') - h'(x)) - (y_{x'} - y_x)| - \\ &\quad |(h(x') - h(x)) - (y_{x'} - y_x)|\big|. \end{aligned} \tag{2.9}$$

Using the identity $\big|\,|X' - Y| - |X - Y|\big| \leq |X' - X|$, valid for all $X, X', Y \in \mathbb{R}$, it follows that

$$\begin{aligned} |c^1_{\mathrm{MP}}(h', x, x') - c^1_{\mathrm{MP}}(h, x, x')| &\leq |\Delta h(x') - \Delta h(x)| && (2.10) \\ &\leq |\Delta h(x')| + |\Delta h(x)|, && (2.11) \end{aligned}$$

which shows the σ-admissibility of c^1_{MP} with $\sigma = 1$. For c^2_{MP}, for all $x, x' \in X$,

$$\begin{aligned} |c^2_{\mathrm{MP}}(h', x, x') - c^2_{\mathrm{MP}}(h, x, x')| &= |\,|(h'(x') - h'(x)) - (y_{x'} - y_x)|^2 && (2.12) \\ &\quad -|(h(x') - h(x)) - (y_{x'} - y_x)|^2\,| \\ &\leq |\Delta h(x') - \Delta h(x)|(|h'(x') - y_{x'}| + && (2.13) \\ &\quad |h(x') - y_{x'}| + |h'(x) - y_x| + |h(x) - y_x|) \\ &\leq 4M(|\Delta h(x')| + |\Delta h(x)|), && (2.14) \end{aligned}$$

which shows the σ-admissibility of c^2_{MP} with $\sigma = 4M$. $\square$

Proposition 2.3 *Assume that the hypotheses in H are bounded, that is for all $h \in H$ and $x \in S$, $|h(x) - y_x| \leq M$. Then, a magnitude-preserving regularization algorithm as defined above is β-stable with $\beta = \frac{4C\sigma_n^2\kappa^2}{m}$.*

Proof. Fix the cost function to be c, one of the σ_n-admissible cost function previously discussed. Let h_S denote the function minimizing $F(h, S)$ and $h_{S^{-k}}$ the one minimizing $F(h, S^{-k})$. We denote by $\Delta h_S = h_{S^{-k}} - h_S$.

Since the cost function c is convex with respect to $h(x)$ and $h(x')$, $\widehat{R}(h, S)$ is also convex with respect to h and for $t \in [0, 1]$,

$$\widehat{R}(h_S + t\Delta h_S, S^{-k}) - \widehat{R}(h_S, S^{-k}) \leq t\left[\widehat{R}(h_{S^{-k}}, S^{-k}) - \widehat{R}(h_S, S^{-k})\right].$$

Similarly,

$$\widehat{R}(h_{S^{-k}} - t\Delta h_S, S^{-k}) - \widehat{R}(h_{S^{-k}}, S^{-k}) \leq t\left[\widehat{R}(h_S, S^{-k}) - \widehat{R}(h_{S^{-k}}, S^{-k})\right].$$

Summing these inequalities yields

$$\widehat{R}(h_S + t\Delta h_S, S^{-k}) - \widehat{R}(h_S, S^{-k}) + \widehat{R}(h_{S^{-k}} - t\Delta h_S, S^{-k}) - \widehat{R}(h_{S^{-k}}, S^{-k}) \leq 0. \quad (2.15)$$

By definition of h_S and $h_{S^{-k}}$ as functions minimizing the objective functions, for all $t \in [0,1]$,

$$F(h_S, S) - F(h_S + t\Delta h_S, S) \leq 0 \text{ and } F(h_{S^{-k}}, S^{-k}) - F(h_{S^{-k}} - t\Delta h_S, S^{-k}) \leq 0. \tag{2.16}$$

Multiplying Inequality 2.15 by C and summing it with the two Inequalities 2.16 lead to

$$A + \|h_S\|_K^2 - \|h_S + t\Delta h_S\|_K^2 + \|h_{S^{-k}}\|_K^2 - \|h_{S^{-k}} - t\Delta h_S\|_K^2 \leq 0. \tag{2.17}$$

with $A = C\left(\widehat{R}(h_S, S) - \widehat{R}(h_S, S^{-k}) + \widehat{R}(h_S + t\Delta h_S, S^{-k}) - \widehat{R}(h_S + t\Delta h_S, S)\right)$.
Since

$$\begin{aligned} A = \tfrac{C}{m^2}\big[& \sum_{i\neq k} c(h_S, x_i, x_k) - c(h_S + t\Delta h_S, x_i, x_k) + \\ & \sum_{i\neq k} c(h_S, x_k, x_i) - c(h_S + t\Delta h_S, x_k, x_i)\big], \end{aligned}$$

by the σ_n-admissibility of c,

$$|A| \leq \frac{2Ct\sigma_n}{m^2} \sum_{i\neq k} (|\Delta h_S(x_k)| + |\Delta h_S(x_i)|) \leq \frac{4Ct\sigma_n\kappa}{m} \|\Delta h_S\|_K.$$

Using the fact that $\|h\|_K^2 = \langle h, h\rangle$ for any h, it is not hard to show that

$$\|h_S\|_K^2 - \|h_S + t\Delta h_S\|_K^2 + \|h_{S^{-k}}\|_K^2 - \|h_{S^{-k}} - t\Delta h_S\|_K^2 = 2t(1-t)\|\Delta h_S\|_K^2.$$

In view of this and the inequality for $|A|$, Inequality 2.17 implies $2t(1-t)\|\Delta h_S\|_K^2 \leq \frac{4Ct\sigma_n\kappa}{m}\|\Delta h_S\|_K$, that is after dividing by t and taking $t \to 0$,

$$\|\Delta h_S\|_K \leq \frac{2C\sigma_n\kappa}{m}.$$

By the σ_n-admissibility of c, for all $x, x' \in X$,

$$\begin{aligned} |c(h_S, x, x') - c(h_{S^{-k}}, x, x')| &\leq \sigma_n(|\Delta h_S(x')| + |\Delta h_S(x)|) && (2.18)\\ &\leq 2\sigma_n\kappa\|\Delta h_S\|_K && (2.19)\\ &\leq \frac{4C\sigma_n^2\kappa^2}{m}. && (2.20) \end{aligned}$$

This shows the β-stability of the algorithm with $\beta = \frac{4C\sigma_n^2\kappa^2}{m}$. $\square$

To shorten the notation, in the absence of ambiguity, we will write in the following $\widehat{R}(h_S)$ instead of $\widehat{R}(h_S, S)$.

Theorem 2.1 *Let c be any of the cost functions defined in Section 2.2.2. Let L be a uniformly β-stable algorithm with respect to the sample S and cost function c and let h_S be the hypothesis returned by L. Assume that the hypotheses in H are bounded, that is for all $h \in H$, sample S, and $x \in S$, $|h(x) - y_x| \leq M$. Then, for any $\epsilon > 0$,*

$$\Pr_{S\sim D}\left[|R(h_S) - \widehat{R}(h_S)| > \epsilon + 2\beta\right] \leq 2e^{-\frac{m\epsilon^2}{2(\beta m + (2M)^k)^2}}.$$

Proof. We apply McDiarmid's inequality [70] to $\Phi(S) = R(h_S) - \widehat{R}(h_S, S)$. We will first give a bound on $\mathbb{E}[\Phi(S)]$ and then show that $\Phi(S)$ satisfies the conditions of McDiarmid's inequality.

We will denote by $S^{i,j}$ the sample derived from S by replacing x_i with x'_i and x_j with x'_j, sampled i.i.d. according to D.

Since the sample points in S are drawn in an i.i.d. fashion, for all $i, j \in [1, m]$,

$$\begin{aligned}
\mathbb{E}_S[\widehat{R}(h_S, S)] &= \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} \mathbb{E}[c(h_S, x_i, x_j)] && (2.21)\\
&= \mathbb{E}_{S \sim D}[c(h_S, x_i, x_j)] && (2.22)\\
&= \mathbb{E}_{S^{i,j} \sim D}[c(h_{S^{i,j}}, x'_i, x'_j)] && (2.23)\\
&= \mathbb{E}_{S, x'_i, x'_j \sim D}[c(h_{S^{i,j}}, x'_i, x'_j)]. && (2.24)
\end{aligned}$$

Note that by definition of $R(h_S)$, $\mathbb{E}_S[R(h_S)] = \mathbb{E}_{S, x'_i, x'_j \sim D}[c(h_S, x'_i, x'_j)]$. Thus, $\mathbb{E}_S[\Phi(S)] = \mathbb{E}_{S,x,x'}[c(h_S, x'_i, x'_j) - c(h_{S^{i,j}}, x'_i, x'_j)]$, and by β-stability (Proposition 2.3)

$$\begin{aligned}
|\,\mathbb{E}_S[\Phi(S)]| &\leq \mathbb{E}_{S,x,x'}[|c(h_S, x'_i, x'_j) - c(h_{S^i}, x'_i, x'_j)|] + && (2.25)\\
&\quad \mathbb{E}_{S,x,x'}[|c(h_{S^i}, x'_i, x'_j) - c(h_{S^{i,j}}, x'_i, x'_j)|] && (2.26)\\
&\leq 2\beta. && (2.27)
\end{aligned}$$

Now,

$$
\begin{aligned}
|R(h_S) - R(h_{S^k})| &= |\,\mathbb{E}_S[c(h_S, x, x') - c(h_{S^k}, x, x')]| && (2.28)\\
&\leq \mathbb{E}_S[|c(h_S, x, x') - c(h_{S^k}, x, x')|] && (2.29)\\
&\leq \beta. && (2.30)
\end{aligned}
$$

For any $x, x' \in X$, $|c(h_S, x_k, x_j) - c(h_{S^k}, x_i, x'_k)| < \mathbb{E}_S[|c(h_{S^k}, x, x') - c(h_{S^k}, x, x')|] \leq (2M)^k$, where $k = 1$ or $k = 2$. Thus, we have

$$
\begin{aligned}
|\widehat{R}(h_S) - \widehat{R}(h_S^k)| &\leq \frac{1}{m^2}\sum_{i\neq k}\sum_{j\neq k}|c(h_S, x_i, x_j) - c(h_{S^k}, x_i, x_j)| + && (2.31)\\
&\quad \frac{1}{m^2}\sum_{j=1}^{m}|c(h_S, x_k, x_j) - c(h_{S^k}, x'_k, x_j)| + && (2.32)\\
&\quad \frac{1}{m^2}\sum_{i=1}^{m}|c(h_S, x_k, x_j) - c(h_{S^k}, x_i, x'_k)| && (2.33)\\
&\leq \frac{1}{m^2}(m^2\beta) + \frac{m}{m^2}2(2M)^k = \beta + 2(2M)^k/m. && (2.34)
\end{aligned}
$$

Thus,

$$|\Phi(S) - \Phi(S^k)| \leq 2(\beta + (2M)^k/m),$$

and $\Phi(S)$ satisfies the hypotheses of McDiarmid's inequality. □

The following Corollary gives stability bounds for the generalization error of magnitude-preserving regularization algorithms.

Corollary 2.1 *Let L be a magnitude-preserving regularization algorithm and*

let c be the corresponding cost function and assume that for all $x \in X, K(x,x) \leq \kappa^2$. Assume that the hypothesis set H is bounded, that is for all $h \in H$, sample S, and $x \in S$, $|h(x) - y_x| \leq M$. Then, with probability at least $1-\delta$,

- *for $k = 1$,*

$$R(h_S) \leq \widehat{R}(h_S) + \frac{8\kappa^2 C}{m} + 2(2\kappa^2 C + M)\sqrt{\frac{2}{m}\log\frac{2}{\delta}};$$

- *for $k = 2$,*

$$R(h_S) \leq \widehat{R}(h_S) + \frac{128\kappa^2 C M^2}{m} + 4M^2(16\kappa^2 C + 1)\sqrt{\frac{2}{m}\log\frac{2}{\delta}}.$$

Proof. By Proposition 2.3, these algorithms are β-stable with $\beta = \frac{4C\sigma_n^2\kappa^2}{m}$. □

These bounds are of the form $R(h_S) \leq \widehat{R}(h_S) + O(\frac{C}{\sqrt{m}})$. Thus, they are effective for values of $C \ll \sqrt{m}$.

2.5 Experiments

In this section, we report the results of experiments with two of our magnitude-preserving algorithms, MPRank and SVRank.

The algorithms were tested on four publicly available data sets, three of which are commonly used for collaborative filtering: MovieLens, Book-Crossings, and Jester Joke. The fourth data set is the Netflix data. The first three datasets are available from the following URL:

http://www.grouplens.org/taxonomy/term/14.

The Netflix data set is available at

http://www.netflixprize.com/download.

Table 2.1: Performance results for MPRank, SVRank, and RankBoost.

Data set	Mean Squared Difference			Mean 1-Norm Difference		
	MPRank	SVRank	RBoost	MPRank	SVRank	RBoost
MovieLens 20-40	2.01 ± 0.02	2.43 ± 0.13	12.88 ± 2.15	1.04 ± 0.05	1.17 ± 0.03	2.59 ± 0.04
MovieLens 40-60	2.02 ± 0.06	2.36 ± 0.16	20.06 ± 2.76	1.04 ± 0.02	1.15 ± 0.07	2.99 ± 0.12
MovieLens 60-80	2.07 ± 0.05	2.66 ± 0.09	21.35 ± 2.71	1.06 ± 0.01	1.24 ± 0.02	3.82 ± 0.23
Jester 20-40	51.34 ± 2.90	55.00 ± 5.14	77.08 ± 17.1	5.08 ± 0.15	5.40 ± 0.20	5.97 ± 0.16
Jester 40-60	46.77 ± 2.03	57.75 ± 5.14	80.00 ± 18.2	4.98 ± 0.13	5.27 ± 0.20	6.18 ± 0.11
Jester 60-80	49.33 ± 3.11	56.06 ± 4.26	88.61 ± 18.6	4.88 ± 0.14	5.25 ± 0.19	6.46 ± 0.20
Netflix Density:32%	1.58 ± 0.04	1.80 ± 0.05	57.5 ± 7.8	0.92 ± 0.01	0.95 ± 0.02	6.48 ± 0.55
Netflix Density:46%	1.55 ± 0.03	1.90 ± 0.06	23.9 ± 2.9	0.95 ± 0.01	1.02 ± 0.02	4.10 ± 0.23
Netflix Density:58%	1.49 ± 0.03	1.93 ± 0.06	12.33 ± 1.47	0.94 ± 0.01	1.06 ± 0.02	3.01 ± 0.15
Books	4.00 ± 3.12	3.64 ± 3.04	7.58 ± 9.95	1.38 ± 0.60	1.32 ± 0.56	1.72 ± 1.05

2.5.1 MovieLens Dataset

The MovieLens dataset consists of approximately one million (1M) ratings by 6,040 users for 3,900 movies. A random subset of 250,000 ratings was selected from the one million ratings to speed up computation. Ratings are integers in the range of 1 to 5. For each user, a different predictive model is designed. The ratings of that user on the 3,900 movies (not all movies will be rated) form the target values y_i. The other users' ratings of the ith movie form the ith input vector x_i.

We followed the experimental set-up of [48] and grouped the reviewers according to the number of movies they reviewed. The groupings were $20-39$ movies, $40-59$ movies, and $60-79$ movies.

Test reviewers were selected among users who had reviewed between 50 and 300 movies. For a given test reviewer, 300 reference reviewers were chosen at random from one of the three groups and their rating were used to form the input vectors. Training was carried out on a randomly chosen subset of half of the test reviewer's movie ratings and testing was performed on the other half. The experiment was done for 300 different test reviewers and the average performance recorded. The whole process was then repeated ten times with a different set of 300 reviewers selected at random. We report mean values and standard deviation for these ten repeated experiments for each of the three groups. Missing review values in the input features were populated with the median review score of the given reference reviewer.

2.5.2 Jester Joke Dataset

The Jester Joke Recommender System dataset contains 4.1M continuous ratings in the range -10.00 to +10.00 of 100 jokes from 73,496 users. The experiments were set up in the same way as for the MovieLens dataset.

2.5.3 Netflix Dataset

The Netflix dataset contains more than 100M ratings by 480,000 users for 17,700 movies. Ratings are integers in the range of 1 to 5. We constructed three subsets of the data with different user densities. Subsets were obtained by thresholding against two parameters: the minimum number of movies rated by a user and the minimum of ratings for a movie. Thus, in choosing users for the training and testing set, we only consider those users who have reviewed more than 150, 500, or 1500 movies respectively. Analogously, in selecting the movies that would appear in the subset data, we only consider those movies that have received at least $360, 1200$, or 1800 reviews. The experiments were then set-up in the same way as for the MovieLens dataset. The mean densities of the three subsets (across the ten repetitions) were $32\%, 46\%$ and 58% respectively. Finally, the test raters were selected from a mixture of the three densities.

2.5.4 Book-Crossing Dataset

The book-crossing dataset contains 1,149,780 ratings in the range $[1, 10]$ for 271,379 books for a group of 278,858 users. The low density of ratings makes predictions very noisy in this task. Thus, we required users to have reviewed at least 200 books, and then only kept books with at least 10 reviews. This left us with a dataset of 89 books and 131 reviewers. For this dataset, each of the 131 reviewers was in turn selected as a test reviewer, and the other 130 reviewers served as input features. The results reported are mean values and standard deviations over these 131 leave-one-out experiments.

2.5.5 Performance Measures and Results

The performance measures we report correspond to the problem we are solving. The cost function of MPRank is designed to minimize the squared difference between all pairs of target values, hence we report the mean squared difference (MSD) over all pairs in the test set of size m' of a hypothesis h:

$$\frac{1}{m'^2} \sum_{i=1}^{m'} \sum_{j=1}^{m'} \left((h(x_j) - h(x_i)) - (y_j - y_i) \right)^2 .$$

The cost function of SVRank minimizes the absolute value of the difference between all pairs of examples, hence we report the average of the 1-norm

Table 2.2: Comparison of MPRank and RankBoost for pairwise misrankings.

Data set	Pairwise Misrankings	
	MPRank	RBoost
MovieLens 40-60	0.471 ± 0.005	0.476 0 ± 0.007
MovieLens 60-80	0.442 ± 0.005	0.463 ± 0.011
Jester 20-40	0.414 ± 0.005	0.479 ± 0.008
Jester 40-60	0.418 ± 0.007	0.432 ± 0.005
Netflix Density:32%	0.433 ± 0.018	0.447 ± 0.027
Netflix Density:46%	0.368 ± 0.014	0.327 ± 0.008
Netflix Density:58%	0.295 ± 0.006	0.318 ± 0.008

difference, M1D:

$$\frac{1}{m'^2}\sum_{i=1}^{m'}\sum_{j=1}^{m'}\left|(h(x_j)-h(x_i))-(y_j-y_i)\right|.$$

The results for MPRank and SVRank are obtained using Gaussian kernels. The width of the kernel and the other cost function parameters were first optimized on a held-out sample. The performance on their respective cost functions was optimized and the parameters fixed at these values.

The results are reported in Table 2.1. They demonstrate that the magnitude-

preserving algorithms are both successful at minimizing their respective objective. MPRank obtains the best MSD values and the two algorithms obtain comparable M1D values. However, overall, in view of these results and the superior computational efficiency of MPRank already pointed out in the previous section, we consider MPRank as the best performing algorithm for such tasks.

To further examine the ranking properties of MPRank we conducted a number of experiments where we compared the pairwise misranking performance of the algorithm to that of RankBoost, an algorithm designed to minimize the number of pairwise misrankings [83]. We used the same features for RankBoost as for MPRank that is we used as weak rankers threshold functions over other reviewers' ratings. As for the other algorithms, the parameter of RankBoost, that is the number of boosting rounds required to minimize pairwise misranking, was determined on a held-out sample and then fixed at this value.

Table 2.2 shows a comparison between these two algorithms. It reports the fraction of pairwise misrankings for both algorithms using the same experimental set-up as previously described:

$$\frac{\sum_{i,j=1}^{m'} 1_{y_i > y_j \wedge h(x_i) \leq h(x_j)}}{\sum_{i,j=1}^{m'} 1_{y_i > y_j}}.$$

The results show that the pairwise misranking error of MPRank is comparable to that of RankBoost. This further demonstrates the benefits of MPRank as

a ranking algorithm.

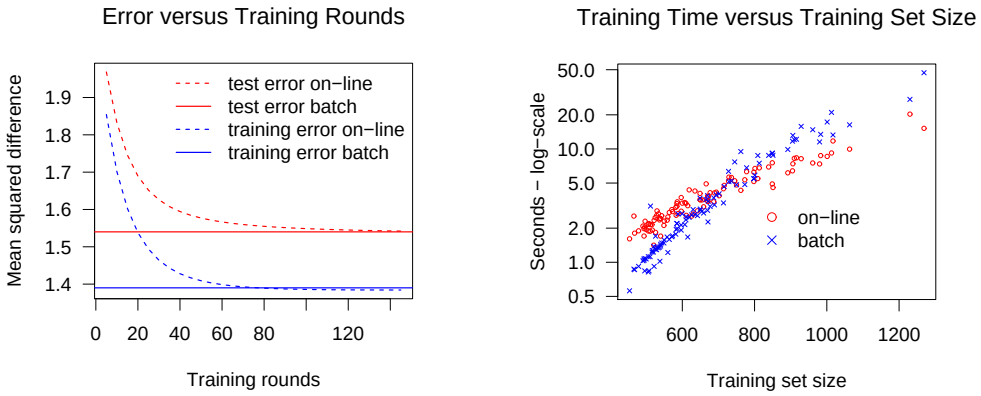
We also tested the performance of RankBoost with respect to MSD and M1D (see Table 2.1). Naturally, RankBoost is not designed to optimize these performance measure and does not lead to competitive results compared to MPRank and SVRank on any of the datasets examined.

2.5.6 On-line Version of MPRank

Using the Netflix data we also experimented with the on-line version of MPRank described in Section 2.3.4. The main questions we wished to investigate were the convergence rate and CPU time savings of the on-line version with respect to the batch algorithm MPRank (Equation 2.3). The batch solution requires a matrix inversion and becomes infeasible for large training sets.

Figure 2.1(a) illustrates the convergence rate for a typical reviewer. In this instance, the training and test sets each consisted of about 700 movies. As can be seen from the plot, the on-line version converges to the batch solution in about 120 rounds, where one round is a full cycle through the training set.

Based on monitoring several convergence plots, we decided on terminating learning in the on-line version of MPRank when consecutive rounds of iterations over the full training set would change the cost function by less than .01 %. Figure 2.1(b) compares the CPU time for the on-line version of MPRank with the batch solution. For both computations of the CPU times, the time to construct the Gram matrix is excluded. The figure shows that the on-line version is significantly faster for large datasets, which extends the applicability



(a) (b)

Figure 2.1: (a) Convergence of the on-line learning algorithm towards the batch solution. Rounding errors give rise to slightly different solutions. (b) Training time in seconds for the on-line and the batch algorithm. For small training set sizes the batch version is fastest, but for larger training set sizes the on-line version is faster. Eventually the batch version becomes infeasible.

of our algorithms beyond the limits of intractable matrix inversion.

2.6 Conclusion

We presented several algorithms for magnitude-preserving ranking problems and provided stability bounds for their generalization error. We also reported the results of several experiments on public datasets comparing these algorithms. We presented an on-line version of one of the algorithms and demonstrated its applicability for very large data sets. We view accurate magnitude-preserving ranking as important to improving the quality of modern recommendation and rating systems. An alternative for incorporating the magni-

tude of preferences in cost functions is to use weighted AUC, where the weights reflect the magnitude of preferences and extend existing algorithms. This however, does not exactly coincide with the objective of preserving the magnitude of preferences.

Chapter 3

Transductive Regression

3.1 Introduction

Many modern learning problems in information extraction, computational biology, natural language processing and other domains can be formulated as *transductive inference* problems [101]. In the transductive setting, the learning algorithm receives both a labeled training set, as in the standard induction setting, and a set of unlabeled test points. The objective is to predict the labels of these test points. No other points will ever be considered.

The transductive setting is an intermediate one between the supervised and the unsupervised learning scenarios. In unsupervised learning, one is given a set X of unlabeled points that are each drawn in an i.i.d. fashion from a common distribution D and the goal is to extract interesting structural properties of the data based on the distribution of points in the set X. The

labels of points in X are not available to the learning algorithm. Examples of unsupervised learning are clustering and dimensionality reduction [97, 9]. In supervised learning on the other hand, each point x in the input space X has an associated (often deterministic) label $y \in Y$ (for example $Y = \{+1, -1\}$ for the task of classification) and the goal is to learn a mapping from X to Y. The learning algorithm is provided a training set in the form of pairs (x_i, y_i), where the x_is are drawn in an i.i.d. fashion from the same distribution D according to which test points are drawn. When $Y \subseteq \mathbb{R}$, the task is that of regression. The performance of a hypothesis $h : X \mapsto Y$ is based on its predictions on unseen test examples. Thus,

$$\text{error}(h) = \mathbb{E}_{x \sim D} \left[c(h, x) \right],$$

where c is a cost function that measures the discrepancy between the true label y of x and the predicted label $h(x)$.

Semi-supervised learning has elements of both unsupervised and supervised learning. In this setting, the learning algorithm is given both labeled and unlabeled data as the training set. The goal, however is to produce a hypothesis that performs well on unseen points drawn according to the same distribution as the points in the training set. The quality of the hypothesis' prediction on unlabeled training points does not matter.

Transductive learning or transductive inference was introduced by Vapnik [101]. In this setting, as in semi-supervised learning, one is given both a labeled

set and an unlabeled set. However, unlike semi-supervised learning, the goal of transduction is only to predict the labels of the unlabeled points supplied in the sample. In fact, the hypothesis is never required to produce a label for an unseen point. This contrasts with inductive learning, where the task is to produce a function that maps the inputs to the labels. Intuitively, the problem of transductive inference seems easier than that of induction since the learning algorithm no longer needs to come up with a function that assigns a label to every point in the input space, but only to produce labels for the given set of test instances. Figure 3.1 contrasts the inductive setting with the transductive setting.

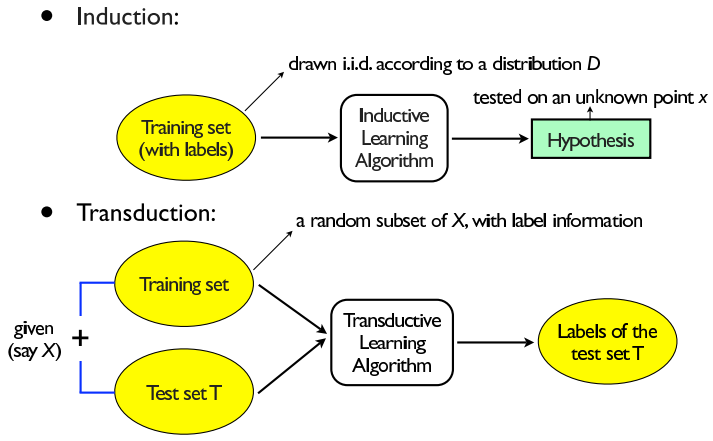


Figure 3.1: The transductive learning approach versus the inductive learning approach.

At first glance, it might seem unlikely that unlabeled points would provide any useful information to the learning algorithm. The hope is to exploit the

distribution of unlabeled examples to help guide the learning algorithm. A common approach is to assume that the target values or labels vary *smoothly*. In this chapter, our analysis also makes use of such local variations with the concept of *local stability*, which will be given a precise definition later. We shall present precise generalization bounds that relate the performance of the learning algorithm to local stability. At a high level, the framework of transductive algorithms we study consists of two steps: in the first step, we learn estimate labels (or *pseudo-targets*) for the unlabeled points in the sample. In the second step, we use true labels of the training samples and the estimate labels of the test points.

The transductive approach is useful in situations in which the amount of unlabeled data far exceeds that of labeled data. With the explosive growth of the Internet, large amounts of digitized data (documents, images, videos) are widely available but the cost of labeling is often prohibitive since it typically requires human assistance. A search engine seeking to build a document rating system or an image labeling system might have access to a small set of manually labeled instances (rated documents, hand-labeled images) and a much larger set of instances for which the labels have to be generated. This motivates the use of transductive algorithms which leverage the unlabeled data during training to improve learning performance.

This chapter deals with the transductive *regression* problem. In this setting, the labels are real-valued (i.e. $Y \subseteq \mathbb{R}$). Several algorithms have been devised for the specific setting of transductive regression [10, 17, 28, 91]. Sev-

eral other algorithms introduced for the classification setting can be viewed as transductive regression algorithms since their objective function is based on the squared loss, for example, [8, 10].

We study a family of transductive regression algorithms. This includes most existing algorithms [10, 17, 28, 91]. We are interested in the generalization properties of this family of algorithms, that is, how well their performance on the test set is reflected in their performance on the training set. In [28], Cortes and Mohri give explicit generalization bounds for this problem that hold for all bounded loss functions and coincide with the tight classification bounds of Vapnik [102] when applied to the problem of transductive classification.

As in Chapter 2, Section 2.4, we use the notion of algorithmic stability to derive generalization bounds for transductive regression. A stability-based approach has also been used to derive generalization bounds in the inductive setting, for the problems of regression, classification [13] and ranking [1]. The advantage of stability bounds over generalization bounds based on complexity measures such as the VC-dimension is that they are algorithm-dependent. Our bound generalizes the stability bounds given by Bousquet and Elisseeff [14] for the inductive setting and extends the stability-based transductive classification bounds of El-Yaniv and Pechyony [45] to regression.

As we saw in Chapter 2, Section 2.4, deriving stability-based generalization bounds in the inductive setting essentially involves applying McDiarmid's concentration bound [69] to the random variable that is the difference of the test error and the training error of the learned hypothesis. Note that this difference

is a function of the training sample alone, which is drawn in an i.i.d. fashion in the inductive setting. In the transductive setting, standard concentration bounds cannot be readily applied because the sampling random variables are not independent, but are drawn uniformly at random without replacement from a finite set. Instead, we need a generalization of McDiarmid's bound that holds for random variables sampled without replacement. We derive such a bound in this chapter, which was independently derived by Pechyony and El-Yaniv in [45].

The results of this chapter were published in [29].

The remainder of this chapter is organized as follows. Section 3.2 gives a formal definition of the transductive regression setting and the notion of stability for transduction. Section 3.3.1 presents a general concentration bound for random variables sampled without replacement. This concentration bound is used to derive a general transductive regression stability bound in Section 3.3.2. Section 3.4 analyzes the stability of local transductive regression algorithms which are briefly described in Section 3.4.1. In Section 3.5, we analyze the stability coefficients of various other algorithms based on their closed-form solution and propose a modification to the apparently unstable algorithm that makes them stable and guarantees a non-trivial generalization bound. Finally, Section 3.6 shows the results of experiments with local transductive regression, demonstrating the benefit of our stability bounds for model selection, and in particular for determining the radius of the local neighborhood used by the algorithm. This provides a partial validation of our bounds

and analysis.

3.2 Preliminaries

3.2.1 Learning Setting

Assume that a full sample X of $m+u$ examples is given. The learning algorithm further receives the labels of a random subset S of X of size m which serves as a training sample:

$$X = \underbrace{x_1, \ldots, x_m}_{\text{labeled}}, \underbrace{x_{m+1}, \ldots, x_{m+u}}_{\text{unlabeled}}. \tag{3.1}$$

The remaining u unlabeled examples, $x_{m+1}, \ldots, x_{m+u} \in X$, serve as test data. We denote by $X = (S, T)$ a partition of X into the training set S and the test set T. The *transductive learning* problem consists of predicting accurately the labels $y_{m+1}, \ldots, y_{m+u}$ of the test examples. No other test examples will ever be considered. The specific problems where the labels are real-valued numbers, as is the case in this chapter, is that of transductive.

Notation: We denote by $c(h, x)$ the cost of an error of a hypothesis h on a point x labeled with $y(x)$. The cost function commonly used in regression is the squared loss,

$$c(h, x) = (h(x) - y(x))^2. \tag{3.2}$$

In the remainder of this chapter, we will assume a squared loss but many of the results presented generalize to other convex cost functions.

We will denote by $\widehat{R}(h)$ and $R(h)$ the training and test errors of a hypothesis h:

$$\widehat{R}(h) = \frac{1}{m}\sum_{k=1}^{m} c(h, x_k), \quad R(h) = \frac{1}{u}\sum_{k=1}^{u} c(h, x_{m+k}). \tag{3.3}$$

Let H denote the hypothesis space from which the learning algorithm selects an element.

Definition 3.1 (B-bounded hypothesis set H) *The hypothesis set H is said to be B-bounded with respect to the set of points X if for all $h \in H$ and all $x \in X$,*

$$|h(x) - y(x)| \leq B.$$

3.2.2 Transductive Stability

The generalization bounds we derive are based on the notion of transductive algorithmic stability.

Definition 3.2 (Transductive β-stability) *Let L be a transductive learning algorithm and let h denote the hypothesis returned by L on the partition (S, T) of X and h' the hypothesis returned on the partition (S', T'). L is said to be* uniformly β-stable *with respect to the cost function c if there exists $\beta \geq 0$ such that when S and S' differ in exactly one point, for all $x \in X$,*

$$\left|c(h, x) - c(h', x)\right| \leq \beta. \tag{3.4}$$

We also present the definition of β-stability in the inductive case.

Definition 3.3 (Inductive β-stability) *Let L be an inductive learning algorithm and let h denote the hypothesis returned by L on a sample S and h' the hypothesis returned on S'. L is said to be* uniformly β-stable *with respect to the cost function c if there exists $\beta \geq 0$ such that for any two sets S and S' that differ in exactly one point and for all $x \in X$*

$$\left|c(h, x) - c(h', x)\right| \leq \beta. \tag{3.5}$$

Definitions 3.2 and 3.3 require that the cost (or the error) of any point x not change by more than β when the training set is changed in exactly one point. A stronger notion of stability is *score* stability, which requires that the hypothesis scores themselves do not change much. Thus, we say that a learning algorithm has score-stability β if the conditions in Equations 3.4 and 3.5 are replaced by the condition:

$$\left|h(x) - h'(x)\right| \leq \beta. \tag{3.6}$$

In our context, the notion of score-stability is stronger than error-stability, in that if a learning algorithm is β-score stable and the hypothesis set is B-bounded, then $c(h, x) - c(h', x) = (h(x) - y(x))^2 - (h'(x) - y(x))^2 \leq 2\beta B$.

3.3 Transductive Regression Stability Bounds

3.3.1 Bound for Sampling without Replacement

Stability-based generalization bounds in the inductive setting are based on McDiarmid's inequality [69]. The main technique used is to show that under suitable conditions on the stability of the algorithm, the difference of the test error and the training error is sharply concentrated around its expected value, and that this expected value itself is small. Roughly speaking, this implies that with high probability, the test error is close to the training error. Since the points in the training sample are drawn in an i.i.d. fashion, McDiarmid's inequality [69] can be applied.

However, in the transductive setting, the sampling random variables are not drawn independently. Thus, McDiarmid's concentration bound cannot be readily used in this case. Instead, a generalization of McDiarmid's bound that holds for random variables sampled without replacement is needed. We present such a generalization in this section. We remark that this bound was independently discovered by Pechyony and El-Yaniv in [45, 46]. To derive the bound, we use the method of averaged bounded differences and the following theorem due to Azuma [7] and McDiarmid [69].

Notation: We will denote by $\mathbf{S}_i^j$ the subsequence of random variables $S_i, \ldots, S_j$ and we write $\mathbf{S}_i^j = \mathbf{x}_i^j$ as a shorthand for the event $S_i = x_i, \ldots, S_j = x_j$.

Theorem 3.1 (McDiarmid [69], 6.10) *Let* $\mathbf{S}_1^m$ *be a sequence of random*

variables, each S_i taking values in the set X, and assume that a measurable function $\phi : X^m \mapsto \mathbb{R}$ satisfies:

$$\forall i \in [m], \forall x_i, x_i' \in X, \left| \mathbb{E}_{\mathbf{S}_{i+1}^m} \left[\phi | \mathbf{S}_1^{i-1}, S_i = x_i \right] - \mathbb{E}_{\mathbf{S}_{i+1}^m} \left[\phi | \mathbf{S}_1^{i-1}, S_i = x_i' \right] \right| \leq c_i.$$

Then for all $\epsilon > 0$,

$$\Pr \left[\phi - \mathbb{E} \left[\phi \right] \geq \epsilon \right] \leq \exp \left(\frac{-2\epsilon^2}{\sum_{i=1}^m c_i^2} \right). \tag{3.7}$$

We need the following technical definition before we present the bound.

Definition 3.4 (Symmetric Function) *A function $\phi : X^m \mapsto \mathbb{R}$ is said to be symmetric if its value does not depend on the order of its arguments. That is, for any two permutations π_1 and π_2 of $\mathbf{S}_1^m$, $\phi(\pi_1) = \phi(\pi_2)$.*

Theorem 3.2 (A Bound for Sampling Without Replacement) *Let $\mathbf{x}_1^m$ be a sequence of random variables, sampled from an underlying set X of $m+u$ elements without replacement, and let $\phi : X^m \mapsto \mathbb{R}$ be a symmetric function such that for all $i \in [1, m]$ and for all $x_1, \ldots, x_m \in X$ and $x_1', \ldots, x_m' \in X$,*

$$|\phi(x_1, \ldots, x_m) - \phi(x_1, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots, x_m)| \leq c.$$

Then, for all $\epsilon > 0$,

$$\Pr \left[\phi - \mathbb{E} \left[\phi \right] \geq \epsilon \right] \leq \exp \left(\frac{-2\epsilon^2}{\alpha(m,u)c^2} \right), \tag{3.8}$$

where $\alpha(m,u) = \frac{mu}{m+u-1/2} \cdot \frac{1}{1-1/(2\max\{m,u\})}$.

Proof. Fix $i \in [m]$ and define $g(\mathbf{S}_1^{i-1})$ as follows.

$$g(\mathbf{S}_1^{i-1}) = \mathbb{E}_{\mathbf{S}_{i+1}^m}\left[\phi|\mathbf{S}_1^{i-1}, S_i = x_i\right] - \mathbb{E}_{\mathbf{S}_{i+1}^m}\left[\phi|\mathbf{S}_1^{i-1}, S_i = x_i'\right].$$

Then,

$$\begin{aligned} g(\mathbf{x}_1^{i-1}) = &\sum_{\mathbf{x}_{i+1}^m} \phi(\mathbf{x}_1^{i-1}, x_i, \mathbf{x}_{i+1}^m) \Pr[\mathbf{S}_{i+1}^m = \mathbf{x}_{i+1}^m | \mathbf{S}_1^{i-1} = \mathbf{x}_1^{i-1}, S_i = x_i] \\ &- \sum_{\mathbf{x}'^m_{i+1}} \phi(\mathbf{x}_1^{i-1}, x_i', \mathbf{x}'^m_{i+1}) \Pr[\mathbf{S}_{i+1}^m = \mathbf{x}'^m_{i+1} | \mathbf{S}_1^{i-1} = \mathbf{x}_1^{i-1}, S_i = x_i']. \end{aligned}$$

We will bound $g(\mathbf{x}_1^{i-1})$ by $c_i = \frac{u}{m+u-i}c$ in order to apply Theorem 3.1 and thereby obtain the claimed bound.

For uniform sampling without replacement, the probability terms can be written as:

$$\Pr\left[\mathbf{S}_{i+1}^m = \mathbf{x}_{i+1}^m | \mathbf{S}_1^{i-1} = \mathbf{x}_1^{i-1}, S_i = x_i\right] = \prod_{k=i}^{m-1} \frac{1}{m+u-k} = \frac{u!}{(m+u-i)!}.$$

Thus,

$$g(\mathbf{x}_1^{i-1}) = \frac{u!}{(m+u-i)!}\Big[\sum_{\mathbf{x}_{i+1}^m} \phi(\mathbf{x}_1^{i-1}, x_i, \mathbf{x}_{i+1}^m) - \sum_{\mathbf{x}'^m_{i+1}} \phi(\mathbf{x}_1^{i-1}, x_i', \mathbf{x}'^m_{i+1})\Big].$$

To compute $\sum_{\mathbf{x}_{i+1}^m} \phi(\mathbf{x}_1^{i-1}, x_i, \mathbf{x}_{i+1}^m) - \sum_{\mathbf{x}'^m_{i+1}} \phi(\mathbf{x}_1^{i-1}, x_i', \mathbf{x}'^m_{i+1})$, we divide the set of permutations $\{\mathbf{x}'^m_{i+1}\}$ into two sets, those that contain the element x_i

and those that do not. If a permutation $\mathbf{x}'^{m}_{i+1}$ contains x_i we can write it as $\mathbf{x}'^{k-1}_{i+1} x_i \mathbf{x}'^{m}_{k+1}$, where k is such that $x'_k = x_i$. We then match it up with the permutation $x_i \mathbf{x}'^{k-1}_{i+1} \mathbf{x}'^{m}_{k+1}$ from the set $\{x_i \mathbf{x}^m_{i+1}\}$. These two permutations contain exactly the same elements, and since the function ϕ is symmetric in its arguments, the difference in the value of the function on the permutations is zero.

In the other case, if a permutation $\mathbf{x}'^{m}_{i+1}$ does not contain the element x_i, then we simply match it up with the same permutation in $\{\mathbf{x}^m_{i+1}\}$. The matching permutations appearing in the summation are then $x_i \mathbf{x}'^{m}_{i+1}$ and $x'_i \mathbf{x}'^{m}_{i+1}$ which clearly only differ with respect to x_i. The difference in the value of the function ϕ in this case can be bounded by c. The number of such permutations can be counted as follows: it is the number of permutations of length $m-i$ from the set X of $m+u$ elements that do not contain any of the elements of $\mathbf{x}_1^{i-1}$, x_i and x'_i, which is equal to $\frac{(m+u-i-1)!}{(u-1)!}$. This leads us to the following upper bound on $\sum_{\mathbf{x}^m_{i+1}} \phi(\mathbf{x}_1^{i-1}, x_i, \mathbf{x}^m_{i+1}) - \sum_{\mathbf{x}'^{m}_{i+1}} \phi(\mathbf{x}_1^{i-1}, x'_i, \mathbf{x}'^{m}_{i+1})$:

$$\sum_{\mathbf{x}^m_{i+1}} \phi(\mathbf{x}_1^{i-1}, x_i, \mathbf{x}^m_{i+1}) - \sum_{\mathbf{x}'^{m}_{i+1}} \phi(\mathbf{x}_1^{i-1}, x'_i, \mathbf{x}'^{m}_{i+1}) \leq \frac{(m+u-i-1)!}{(u-1)!} c,$$

which implies that $|g(\mathbf{x}_1^{i-1})| \leq \frac{u!}{(m+u-i)!} \cdot \frac{(m+u-i-1)!}{(u-1)!} c \leq \frac{u}{m+u-i} c$. We need to bound $\sum_{i=1}^{m} \left(\frac{u}{m+u-i} c\right)^2$ in order to apply the bound from Theorem 3.1. To this end, note that

$$\sum_{i=1}^{m} \frac{1}{(m+u-i)^2} = \sum_{j=u}^{m+u-1} \frac{1}{j^2} \leq \int_{u-1/2}^{m+u-1/2} \frac{dx}{x^2} = \frac{m}{m+u-1/2} \cdot \frac{1}{u-1/2}.$$

Applying Theorem 3.1 then yields:

$$\Pr\left[\phi - \mathbb{E}\left[\phi\right] \geq \epsilon\right] \leq \exp\left(\frac{-2\epsilon^2}{\alpha_u(m,u)c^2}\right),$$

where $\alpha_u(m,u) = \frac{mu}{m+u-1/2} \cdot \frac{1}{1-1/(2u)}$. The function ϕ is symmetric in m and u in the sense that selecting one of the sets uniquely determines the other set. The statement of the theorem then follows obtaining a similar bound with $\alpha_m(m,u) = \frac{mu}{m+u-1/2} \cdot \frac{1}{1-1/(2m)}$ and taking the tighter of the two bounds. □

3.3.2 Transductive Stability Bound

We apply the concentration bound presented in the previous section (Theorem 3.2) to the random variable that is the difference of the training error and the test error. We call this random variable $\phi(S) = R(h) - \widehat{R}(h)$. Note that the elements of the training set S uniquely determine the hypothesis h[1], which in turn determines the test error $R(h)$ and the training error $\widehat{R}(h)$. In order to apply Theorem 3.2, we need to bound $\mathbb{E}_S\left[\phi(S)\right]$, where S is a random subset of X of m elements, and $|\phi(S) - \phi(S')|$ where S and S' are samples differing on exactly one point. The stability parameter β of the transductive learning algorithm (Definition 3.2) is needed in bounding both these quantities.

Lemma 3.1 *Let H be a B-bounded hypothesis set. Let L be a β-stable algorithm and let S and S' be two training sets of size m that differ in exactly*

[1]We limit ourselves to deterministic learning algorithms that always produce the same hypothesis on the same training/test partitions X.

one point. Let $h \in H$ be the hypothesis returned by L when trained on S and $h' \in H$ the one returned when L is trained on S'. Then,

$$|\phi(S) - \phi(S')| \leq 2\beta + \frac{m+u}{mu} B^2. \tag{3.9}$$

Proof. By the definition of S', there exist $i \in [1, m]$ and $j \in [1, u]$ such that $S' = S \setminus \{x_i\} \cup \{x_{m+j}\}$. $\phi(S) - \phi(S')$ can written as follows:

$$\begin{aligned}\phi(S) - \phi(S') = \frac{1}{u} \sum_{k=1, k\neq j}^{u} [c(h, x_{m+k}) - c(h', x_{m+k})] + \frac{1}{m} \sum_{k=1, k\neq i}^{m} [c(h', x_k) - c(h, x_k)] \\ + \frac{1}{u} [c(h, x_{m+j}) - c(h', x_i)] + \frac{1}{m} [c(h', x_{m+j}) - c(h, x_i)] .\end{aligned}$$

Since H is B-bounded, the squared-loss $c(\cdot, \cdot)$ is such that $c(h, x) \leq B^2$ for all $x \in X, h \in H$. Thus,

$$|\phi(S) - \phi(S')| \leq \frac{(u-1)\beta}{u} + \frac{(m-1)\beta}{m} + \frac{B^2}{u} + \frac{B^2}{m} \leq 2\beta + B^2 \left(\frac{1}{u} + \frac{1}{m}\right).$$

□

Lemma 3.2 *Let h be the hypothesis returned by a β-stable algorithm L. Then, $|\mathbb{E}_S [\phi(S)] | \leq \beta$.*

Proof. By the definition of $\phi(S)$,

$$\begin{aligned}\mathbb{E}_S [\phi(S)] &= \mathbb{E}_S [R(h)] - \mathbb{E}_S \left[\widehat{R}(h)\right] \\ &= \frac{1}{u} \sum_{k=1}^{u} \mathbb{E}_S [c(h, x_{m+k})] - \frac{1}{m} \sum_{k=1}^{m} \mathbb{E}_S [c(h, x_k)]\end{aligned}$$

$\mathbb{E}_S[c(h, x_{m+k})]$ is the same for all $1 \leq k \leq u$, and similarly, $\mathbb{E}_S[c(h, x_k)]$ is the same for all $1 \leq k \leq m$. Let $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, u\}$. Then,

$$\begin{aligned}
\mathbb{E}_S[\phi(S)] &= \mathbb{E}_S[c(h, x_{m+j})] - \mathbb{E}_S[c(h, x_i)] \\
&= \mathbb{E}_{S' \sim X}[c(h', x_i)] - \mathbb{E}_{S \sim X}[c(h, x_i)] \\
&= \mathbb{E}_{S, S' \sim X}[c(h', x_i) - c(h, x_i)] \\
&\leq \beta.
\end{aligned}$$

□

Theorem 3.3 *Let H be a B-bounded hypothesis set and L a β-stable algorithm. Let S be a random subset of labeled points of size m drawn from X and let h be the hypothesis returned by L on the corresponding partition of X. Then, for any $\delta > 0$, with probability at least $1 - \delta$,*

$$R(h) \leq \widehat{R}(h) + \beta + \left(2\beta + \frac{B^2(m+u)}{mu}\right)\sqrt{\frac{\alpha(m,u)\ln\frac{1}{\delta}}{2}}, \tag{3.10}$$

where $\alpha(m,u) = \frac{mu}{m+u-1/2} \cdot \frac{1}{1-1/(2\max\{m,u\})}$.

Proof. Recall that $\phi = R(h) - \widehat{R}(h)$. By Lemma 3.1, for sets S and S' that differ in one point, $|\phi(S) - \phi(S')| \leq 2\beta + \frac{m+u}{mu}B^2$. By Lemma 3.2, $\mathbb{E}_S[\phi] \leq \beta$.

Thus Theorem 3.2 yields:

$$\begin{aligned}\Pr_S\left[R(h) - \widehat{R}(h) - \beta \geq \epsilon\right] &\leq \exp\left(\frac{-2\epsilon^2}{\alpha(m,u)\left(2\beta + \left(\frac{m+u}{mu}\right)B^2\right)^2}\right) \\ &= \exp\left(\frac{-2\epsilon^2 m^2 u^2}{\alpha(m,u)\left(2\beta mu + (m+u)B^2\right)^2}\right)\end{aligned}$$

Setting the upper bound to δ yields

$$\epsilon = \left(2\beta + \frac{B^2(m+u)}{mu}\right)\sqrt{\frac{\alpha(m,u)\ln\frac{1}{\delta}}{2}}.$$

Thus, with probability $1-\delta$, $R(h)$ satisfies the bound in Equation 3.10. □

This is a general bound that applies to *any* transductive algorithm. To apply it, the stability coefficient β, which depends on m and u, needs to be determined. In subsequent sections, we derive bounds on β for a number of transductive regression algorithms [8, 28, 105, 106, 107].

3.4 Stability of Local Transductive Regression Algorithms

This section describes and analyzes a general family of local transductive regression algorithms (`LTR`) generalizing the algorithm of Cortes and Mohri [28].

3.4.1 Local Transductive Regression Algorithms

LTR algorithms can be viewed as a generalization of the so-called kernel regularization-based learning algorithms to the transductive setting. The objective function that they minimize is of the form:

$$F(h,S) = \|h\|_K^2 + \frac{C}{m}\sum_{k=1}^{m} c(h,x_k) + \frac{C'}{u}\sum_{k=1}^{u} \widetilde{c}(h,x_{m+k}), \tag{3.11}$$

where $\|\cdot\|_K$ is the norm in the reproducing kernel Hilbert space (RKHS) with associated kernel K, $C \geq 0$ and $C' \geq 0$ are trade-off parameters, and $\widetilde{c}(h,x) = (h(x) - \widetilde{y}(x))^2$ is the error of the hypothesis h on the unlabeled point x with respect to a pseudo-target $\widetilde{y}$. The objective function contains three terms: a regularization term based on $\|\cdot\|_K$, a term corresponding to the empirical error as in standard kernel-based regularization algorithms, and a third term corresponding to the error with respect to the pseudo-targets.

Pseudo-targets are obtained from neighborhood labels $y(x)$ by a local weighted average. Neighborhoods can be defined as a ball of radius r around each point in the feature space. We will denote by β_{loc} the score-stability (see Equation 3.6) of the algorithm used to determine pseudo-targets, that is the maximal amount by which the two hypotheses differ on an given point, when trained on samples disagreeing on one point.

In this section, we make the bounded-labels assumption, that is for all $x \in S$, $|y(x)| \leq M$. Then, for any $x \in S$ the pseudo-target $\tilde{y}(x)$ assigned by the local estimator to the unlabeled examples satisfies $|\tilde{y}(x)| \leq M$. This

assumption is quite mild and is satisfied when the pseudo-targets are computed by Nadaraya-Watson estimators or by the other methods suggested in [28], such as kernel ridge regression.

We also assume that for any $x \in X$, $K(x,x) \leq \kappa^2$. We will use the following bound based on the reproducing property and the Cauchy-Schwarz inequality valid for any hypothesis $h \in H : \forall x \in X$ [89],

$$|h(x)| = |\langle h, K(x,\cdot)\rangle| \leq \|h\|_K \sqrt{K(x,x)} \leq \kappa \|h\|_K. \tag{3.12}$$

Lemma 3.3 *Let h be the hypothesis minimizing (3.11). Assume that for any $x \in X$, $K(x,x) \leq \kappa^2$. Then, for any $x \in X$, $|h(x)| \leq \kappa M \sqrt{C + C'}$.*

Proof. The proof is a straightforward adaptation of the technique of [14] to `LTR` algorithms. By Equation 3.12, $|h(x)| \leq \kappa \|h\|_K$. Let $\mathbf{0} \in \mathbb{R}^{m+u}$ be the hypothesis assigning label zero to all examples. By the definition of h,

$$F(h,S) \leq F(\mathbf{0},S) \leq (C + C')M^2.$$

Using the fact that $\|h\|_K \leq \sqrt{F(h,S)}$ yields the statement of the lemma. □

Since $|h(x)| \leq \kappa M \sqrt{C + C'}$, this immediately gives us a bound on $|h(x) - y(x)| \leq M(1 + \kappa\sqrt{C + C'})$ and we are in a position to apply Theorem 3.3 with $B = AM$, $A = 1 + \kappa\sqrt{C + C'}$.

Let h be a hypothesis obtained by training on S and h' by training on S'. To determine the stability coefficient β, we must upper-bound $|c(h,x) - c(h',x)|$.

Let $\Delta h = h - h'$. Then, for all $x \in X$,

$$\begin{aligned} |c(h,x) - c(h',x)| &= \Big|\Delta h(x)\left[(h(x) - y(x)) + (h'(x) - y(x))\right]\Big| \\ &\leq 2M(1 + \kappa\sqrt{C + C'})|\Delta h(x)|. \end{aligned}$$

As in Inequality 3.12, for all $x \in X$, $|\Delta h(x)| \leq \kappa \|\Delta h\|_K$, thus for all $x \in X$,

$$|c(h,x) - c(h',x)| \leq 2M(1 + \kappa\sqrt{C + C'})\kappa\|\Delta h\|_K. \tag{3.13}$$

It remains to bound $\|\Delta h\|_K$. Our approach towards bounding $\|\Delta h\|_K$ is similar to the one used by Bousquet and Elisseeff [13], and relies on the convexity of $h \mapsto c(h,x)$. Note however that in the case of $\widetilde{c}$, the pseudo-targets may depend on the training set S. This dependency matters when we wish to apply convexity with two hypotheses h and h' obtained by training on different samples S and S'. For convenience, for any two such fixed hypotheses h and h', we extend the definition of $\widetilde{c}$ as follows. For all $t \in [0,1]$,

$$\widetilde{c}(th + (1-t)h', x) = \big((th + (1-t)h')(x) - (t\widetilde{y} + (1-t)\widetilde{y}')\big)^2. \tag{3.14}$$

This allows us to use the same convexity property for $\widetilde{c}$ as for c for any two fixed hypotheses h and h' as verified by the following lemma.

Lemma 3.4 *Let h be a hypothesis obtained by training on S and h' by training*

on S'. Then, for all $t \in [0,1]$,

$$t\widetilde{c}(h,x) + (1-t)\widetilde{c}(h',x) \geq \widetilde{c}(th + (1-t)h', x). \tag{3.15}$$

Proof. Let $\widetilde{y} = \widetilde{y}(x)$ be the pseudo-target value at x when the training set is S and $\widetilde{y}' = \widetilde{y}'(x)$ when the training set is S'. For all $t \in [0,1]$,

$$\begin{aligned}
&tc(h,x) + (1-t)c(h',x) - c(th + (1-t)h', x) \\
&= t(h(x) - \widetilde{y})^2 + (1-t)(h'(x) - \widetilde{y}')^2 - \left[(th(x) + (1-t)h'(x) - (t\widetilde{y} + (1-t)\widetilde{y}')\right]^2 \\
&= t(h(x) - \widetilde{y})^2 + (1-t)(h'(x) - \widetilde{y}')^2 - \left[t(h(x) - \widetilde{y}) + (1-t)(h'(x) - \widetilde{y}')\right]^2 .
\end{aligned}$$

The statement of the lemma follows directly by the convexity of the function $x \mapsto x^2$ defined over $\mathbb{R}$. $\square$

Recall that β_{loc} denotes the score-stability of the algorithm that produces the pseudo-targets. In Lemma 3.6 we present an upper-bound $\|\Delta h\|_K$, which can then be plugged into Equation 3.13 to determine the stability of `LTR`.

Lemma 3.5 *Assume that for all $x \in X$, $|y(x)| \leq M$. Let S and S' be two samples differing by exactly one point. Let h be the hypothesis returned by the algorithm minimizing the objective function $F(h,S)$, h' be the hypothesis obtained by minimization of $F(h,S')$ and let $\widetilde{y}$ and $\widetilde{y}'$ be the corresponding*

pseudo-targets. Then for all $i \in \{1, \ldots, m\}$,

$$\begin{aligned}
&\frac{C}{m}\left[c(h', x_i) - c(h, x_i)\right] + \frac{C'}{u}\left[\widetilde{c}(h', x_i) - \widetilde{c}(h, x_i)\right] \\
&\leq 2AM\left(\kappa\|\Delta h\|_K\left(\frac{C}{m} + \frac{C'}{u}\right) + \beta_{loc}\frac{C'}{u}\right), \qquad (3.16)
\end{aligned}$$

where $\Delta h = h' - h$ *and* $A = 1 + \kappa\sqrt{C + C'}$.

Proof. From Equation 3.13, we know that:

$$|c(h', x_i) - c(h, x_i)| \leq 2M(1 + \kappa\sqrt{C + C'})\kappa\|\Delta h\|_K.$$

It remains to bound $|\widetilde{c}(h', x_i) - \widetilde{c}(h, x_i)|$.

$$\begin{aligned}
\widetilde{c}(h', x_i) - \widetilde{c}(h, x_i) &= (h'(x) - \widetilde{y}'(x))^2 - (h(x) - \widetilde{y}(x))^2 \\
&= \left((h'(x) - \widetilde{y}'(x)) + (h(x) - \widetilde{y}(x))\right)\left(\Delta h(x) - (\widetilde{y}'(x) - \widetilde{y}(x))\right) \\
&\leq 2M(1 + \kappa\sqrt{C + C'})\left(\kappa\|\Delta h\|_K + \beta_{loc}\right)
\end{aligned}$$

Here, we are using score-stability β_{loc} of the local algorithm in $|\widetilde{y}'(x) - \widetilde{y}(x)| \leq \beta_{loc}$ (see Equation 3.6) and that $|h(x) - \widetilde{y}(x)| \leq M(1 + \kappa\sqrt{C + C'})$ when $|\widetilde{y}(x)| \leq M$ (by Lemma 3.3).

Plugging the bounds for $|c(h', x_i) - c(h, x_i)|$ and $|\widetilde{c}(h', x_i) - \widetilde{c}(h, x_i)|$ into the LHS of Equation 3.16 yields the statement of the lemma. □

Lemma 3.6 *Assume that for all* $x \in X$, $|y(x)| \leq M$. *Let* S *and* S' *be two samples differing by exactly one point. Let* h *be the hypothesis returned by the*

algorithm minimizing the objective function $F(h, S)$, h' the hypothesis obtained by minimization of $F(h, S')$ and let $\widetilde{y}$ and $\widetilde{y}'$ be the corresponding pseudo-targets. Then, for any $i \in [1, m]$, $j \in [1, u]$,

$$\|\Delta h\|_K^2 \leq 2AM \left(\kappa \|\Delta h\|_K \left(\frac{C}{m} + \frac{C'}{u} \right) + \beta_{loc} \frac{C'}{u} \right), \tag{3.17}$$

where $\Delta h = h' - h$ and $A = 1 + \kappa\sqrt{C + C'}$.

Proof. By the definition of h and h', we have

$$h = \operatorname*{argmin}_{h \in H} F(h, S) \quad \text{and} \quad h' = \operatorname*{argmin}_{h \in H} F(h, S').$$

Let $t \in [0, 1]$. Then $h + t\Delta h$ and $h' - t\Delta h$ satisfy $F(h, S) - F(h + t\Delta h, S) \leq 0$ and $F(h', S') - F(h' - t\Delta h, S') \leq 0$.

For notational ease, let $h_{t\Delta}$ denote $h + t\Delta h$ and $h'_{t\Delta}$ denote $h' - t\Delta h$. Summing these two inequalities yields

$$\begin{aligned}
&\frac{C}{m} \sum_{k=1}^{m} [c(h, x_k) - c(h_{t\Delta}, x_k)] + \frac{C'}{u} \sum_{k=1}^{u} [\widetilde{c}(h, x_{m+k}) - \widetilde{c}(h_{t\Delta h}, x_{m+k})] + \\
&\frac{C}{m} \sum_{k=1, k \neq i}^{m} [c(h', x_k) - c(h'_{t\Delta}, x_k)] + \frac{C'}{u} \sum_{k=1, k \neq j}^{u} [\widetilde{c}(h', x_{m+k}) - \widetilde{c}(h'_{t\Delta}, x_{m+k})] + \\
&\frac{C}{m} [c(h', x_{m+j}) - c(h'_{t\Delta}, x_{m+j})] + \frac{C'}{u} [\widetilde{c}(h', x_i) - \widetilde{c}(h'_{t\Delta}, x_i)] + \\
&\|h\|_K^2 - \|h_{t\Delta h}\|_K^2 + \|h'\|_K^2 - \|h'_{t\Delta}\|_K^2 \quad \leq 0.
\end{aligned}$$

By the convexity of $c(h, \cdot)$ in h, it follows that for all $k \in [1, m+u]$

$$c(h, x_k) - c(h_{t\Delta h}, x_k) \geq t\left[c(h, x_k) - c(h + \Delta h, x_k)\right], \tag{3.18}$$

and

$$c(h', x_k) - c(h'_{t\Delta}, x_k) \geq t\left[c(h', x_k) - c(h' - \Delta h, x_k)\right]. \tag{3.19}$$

By Lemma 3.4, similar inequalities hold for $\widetilde{c}$. These observations lead to:

$$\begin{aligned}
&\frac{Ct}{m}\sum_{k=1}^{m}\left[c(h, x_k) - c(h', x_k)\right] + \frac{C't}{u}\sum_{k=1}^{u}\left[\widetilde{c}(h, x_{m+k}) - \widetilde{c}(h', x_{m+k})\right] + \\
&\frac{Ct}{m}\sum_{k=1,k\neq i}^{m}\left[c(h', x_k) - c(h, x_k)\right] + \frac{C't}{u}\sum_{k=1,k\neq j}^{u}\left[\widetilde{c}(h', x_{m+k}) - \widetilde{c}(h, x_{m+k})\right] + \\
&\frac{Ct}{m}\left[c(h', x_{m+j}) - c(h, x_{m+j})\right] + \frac{C't}{u}\left[\widetilde{c}(h', x_i) - \widetilde{c}(h, x_i)\right] + \\
&\|h\|_K^2 - \|h_{t\Delta h}\|_K^2 + \|h'\|_K^2 - \|h'_{t\Delta}\|_K^2 \quad \leq 0.
\end{aligned}$$

Let E denote $\|h\|_K^2 - \|h_{t\Delta h}\|_K^2 + \|h'\|_K^2 - \|h'_{t\Delta}\|_K^2$. Simplifying the previous inequality leads to:

$$\begin{aligned}
E \;\leq\; &\frac{Ct}{m}\left[c(h', x_i) - c(h, x_i) + c(h, x_{m+j}) - c(h', x_{m+j})\right] - \\
&\frac{C't}{u}\left[\widetilde{c}(h', x_i) - \widetilde{c}(h, x_i) + \widetilde{c}(h, x_{m+j}) - \widetilde{c}(h', x_{m+j})\right].
\end{aligned}$$

Let $A = 1 + \kappa\sqrt{C + C'}$. Using Lemma 3.5 twice (with x_i and x_{m+j}), the

expression above can be bounded by

$$E \leq 4AMt\left(\kappa\|\Delta h\|_K\left(\frac{C}{m}+\frac{C'}{u}\right)+\beta_{loc}\frac{C'}{u}\right), \tag{3.20}$$

. Finally, since $\|h\|_K^2 = \langle h, h\rangle_K$ for any $h \in H$, it is not hard to show that:

$$\|h\|_K^2 - \|h + t\Delta h\|_K^2 + \|h'\|_K^2 - \|h' - t\Delta h\|_K^2 = 2t\|\Delta h\|_K^2(1-t). \tag{3.21}$$

Using Equation 3.21 in Equation 3.20, it follows that:

$$\|\Delta h\|_K^2(1-t) \leq 2AM\left(\kappa\|\Delta h\|_K\left(\frac{C}{m}+\frac{C'}{u}\right)+\beta_{loc}\frac{C'}{u}\right).$$

Taking the limit as $t \to 0$ yields the statement of the lemma. □

Theorem 3.4 *Assume that for all $x \in X$, $|y(x)| \leq M$ and there exists κ such that $\forall x \in X$, $K(x,x) \leq \kappa^2$. Further, assume that the local estimator has uniform stability coefficient β_{loc}. Let $A = 1 + \kappa\sqrt{C + C'}$. Then,* `LTR` *is uniformly β-stable with*

$$\beta \leq 2(AM)^2\kappa^2\left[\frac{C}{m}+\frac{C'}{u}+\sqrt{\left(\frac{C}{m}+\frac{C'}{u}\right)^2+\frac{2C'\beta_{loc}}{AM\kappa^2 u}}\right].$$

Proof. From Lemma 3.6, we know that

$$\|\Delta h\|_K^2 \leq 2AM\left(\kappa\|\Delta h\|_K\left(\frac{C}{m}+\frac{C'}{u}\right)+\beta_{loc}\frac{C'}{u}\right),$$

where $\Delta h = h' - h$ and $A = 1 + \kappa\sqrt{C + C'}$. This implies that $\|\Delta h\|_K$ is bounded by the non-negative root of the second-degree polynomial which gives

$$\|\Delta h\|_K \leq AM\kappa \left[\left(\frac{C}{m} + \frac{C'}{u} \right) + \sqrt{\left(\frac{C}{m} + \frac{C'}{u} \right)^2 + \frac{2C'\beta_{loc}}{AM\kappa^2 u}} \right].$$

Using the above bound on $\|\Delta h\|_K$ in Equation 3.13 yields the desired bound on the stability coefficient of `LTR` and completes the proof. □

Our experiments with local transductive regression in Section 3.6 will show the benefit of this bound for model selection.

3.5 Stability Based on Closed-Form Solution

3.5.1 Unconstrained Regularization Algorithms

In this section, we consider a family of transductive regression algorithms that can be formulated as the following optimization problem:

$$\min_{\mathbf{h}} \mathbf{h}^\top \mathbf{Q}\mathbf{h} + (\mathbf{h} - \mathbf{y})^\top \mathbf{C}(\mathbf{h} - \mathbf{y}). \tag{3.22}$$

$\mathbf{Q} \in \mathbb{R}^{(m+u)\times(m+u)}$ is a symmetric regularization matrix, $\mathbf{C} \in \mathbb{R}^{(m+u)\times(m+u)}$ is a symmetric matrix of empirical weights (in practice it is often a diagonal matrix), $\mathbf{y} \in \mathbb{R}^{(m+u)\times 1}$ are the target values of the m labeled points together with (possibly) the pseudo-target values of the u unlabeled points (in some formulations, the pseudo-target value is 0), and $\mathbf{h} \in \mathbb{R}^{(m+u)\times 1}$ is a column

vector whose ith row is the predicted target value for x_i. In addition to the bounded labels assumption, in this section, we also assume that the hypothesis values are bounded. That is, for all $x \in X, |h(x)| \leq M$. These assumptions are standard [8]. The closed-form solution of (3.22) is given by

$$\mathbf{h}^* = (\mathbf{C}^{-1}\mathbf{Q} + \mathbf{I})^{-1}\mathbf{y}. \tag{3.23}$$

The formulation (3.22) is quite general and includes as special cases the algorithms of [8, 105, 106, 107]. In this section, we present a general framework for bounding the stability coefficient of these algorithms. Then we examine specific algorithms and use the general results derived in this section to bound the stability coefficient of each algorithm in turn. One point to note is that the objective function in Equation 3.22 is distinct from the one in Equation 3.11 in that the prediction for a test point is no longer linear in an underlying Hilbert space.

For a symmetric matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ we will denote by $\lambda_M(\mathbf{A})$ its largest and by $\lambda_m(\mathbf{A})$ its smallest eigenvalue. Then, for any $\mathbf{v} \in \mathbb{R}^{n\times 1}$, $\lambda_m(\mathbf{A})\|\mathbf{v}\|_2 \leq \|\mathbf{A}\mathbf{v}\|_2 \leq \lambda_M(\mathbf{A})\|\mathbf{v}\|_2$. In the proof of the following theorem, we will also use the fact that for symmetric matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$, $\lambda_M(\mathbf{AB}) \leq \lambda_M(\mathbf{A})\lambda_M(\mathbf{B})$.

Theorem 3.5 *Let* $\mathbf{h}^*$ *and* $\mathbf{h}'^*$ *solve (3.22), under test and training sets that differ exactly in one point and let* $\mathbf{C}, \mathbf{C}', \mathbf{y}, \mathbf{y}'$ *be the analogous empirical weight*

and the target value matrices. Then,

$$\|\mathbf{h}^* - \mathbf{h}'^*\|_\infty \leq \|\mathbf{h}^* - \mathbf{h}'^*\|_2 \leq \frac{\|\mathbf{y} - \mathbf{y}'\|_2}{\frac{\lambda_m(\mathbf{Q})}{\lambda_M(\mathbf{C})} + 1} + \frac{\lambda_M(\mathbf{Q})\|\mathbf{C}'^{-1} - \mathbf{C}^{-1}\|_2 \, \|\mathbf{y}'\|_2}{\left(\frac{\lambda_m(\mathbf{Q})}{\lambda_M(\mathbf{C}')} + 1\right)\left(\frac{\lambda_m(\mathbf{Q})}{\lambda_M(\mathbf{C})} + 1\right)} \quad (3.24)$$

Proof. Let $\Delta\mathbf{h}^* = \mathbf{h}^* - \mathbf{h}'^*$ and $\Delta\mathbf{y} = \mathbf{y} - \mathbf{y}'$. By definition,

$$\begin{aligned}
\Delta\mathbf{h}^* =& (\mathbf{C}^{-1}\mathbf{Q} + \mathbf{I})^{-1}\mathbf{y} - (\mathbf{C}'^{-1}\mathbf{Q} + \mathbf{I})^{-1}\mathbf{y}' \\
=& (\mathbf{C}^{-1}\mathbf{Q} + \mathbf{I})^{-1}\Delta\mathbf{y} + ((\mathbf{C}^{-1}\mathbf{Q} + \mathbf{I})^{-1} - (\mathbf{C}'^{-1}\mathbf{Q} + \mathbf{I})^{-1})\mathbf{y}' \\
=& (\mathbf{C}^{-1}\mathbf{Q} + \mathbf{I})^{-1}\Delta\mathbf{y} \\
&+ \left[(\mathbf{C}'^{-1}\mathbf{Q} + \mathbf{I})^{-1}\left[(\mathbf{C}'^{-1} - \mathbf{C}^{-1})\mathbf{Q}\right](\mathbf{C}^{-1}\mathbf{Q} + \mathbf{I})^{-1}\right]\mathbf{y}'.
\end{aligned}$$

Using standard linear algebra inequalities yields:

$$\|\Delta\|_2 \leq \frac{\|\Delta\mathbf{y}\|_2}{\lambda_m(\mathbf{C}^{-1}\mathbf{Q} + \mathbf{I})} + \frac{\lambda_M(\mathbf{Q})\|\mathbf{C}'^{-1} - \mathbf{C}^{-1}\|_2 \cdot \|\mathbf{y}'\|_2}{\lambda_m(\mathbf{C}'^{-1}\mathbf{Q} + \mathbf{I})\lambda_m(\mathbf{C}^{-1}\mathbf{Q} + \mathbf{I})} \quad (3.25)$$

Furthermore, $\lambda_m(\mathbf{C}^{-1}\mathbf{Q} + \mathbf{I}) \geq \frac{\lambda_m(\mathbf{Q})}{\lambda_M(\mathbf{C})} + 1$. Plugging this bound back into Equation 3.25 yields:

$$\|\Delta\|_2 \leq \frac{\|\Delta\mathbf{y}\|_2}{\frac{\lambda_m(\mathbf{Q})}{\lambda_M(\mathbf{C})} + 1} + \frac{\lambda_M(\mathbf{Q})\|\mathbf{C}'^{-1} - \mathbf{C}^{-1}\|_2 \cdot \|\mathbf{y}'\|_2}{\left(\frac{\lambda_m(\mathbf{Q})}{\lambda_M(\mathbf{C}')} + 1\right)\left(\frac{\lambda_m(\mathbf{Q})}{\lambda_M(\mathbf{C})} + 1\right)}.$$

□

The theorem helps derive score-stability bounds for various transductive regression algorithms [106, 105, 107] based on the closed-form solution for the

hypothesis. Recall that score-stability (Equation 3.6) is the maximum change in the hypothesis score on any point x as the learning algorithm is trained on two training sets that differ in exactly one point. Thus, the score-stability is bounded by $\|\mathbf{h}^* - \mathbf{h}'^*\|_\infty$.

For each of the algorithms in [106, 105, 107], an estimate of 0 is used for unlabeled points. Thus, the vector $\mathbf{y}$ has the following structure: the entries corresponding to training examples are their true labels and those corresponding to the unlabeled examples are 0. For each one of the three algorithms, we make the bounded labels assumption ($\forall x \in X, |y(x)| \leq M$ for some $M > 0$). It is then not difficult to show that $\|\mathbf{y} - \mathbf{y}'\|_2 \leq \sqrt{2}M$ and $\|\mathbf{y}'\|_2 \leq \sqrt{m}M$. Furthermore, all the derived stability bounds are based on the notion of score-stability and not on error-stability.

Consistency method (`CM`) In the `CM` algorithm [106], the regularization matrix $\mathbf{Q}$ is a normalized Laplacian of a weight matrix $\mathbf{W} \in \mathbb{R}^{(m+u)\times(m+u)}$ that captures affinity between pairs of points in the full sample X. Thus, $\mathbf{Q} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$, where $\mathbf{D} \in \mathbb{R}^{(m+u)\times(m+u)}$ is a diagonal matrix, whose (i,i)th element is the sum of the ith row in $\mathbf{W}$. Note that $\lambda_m(\mathbf{Q}) = 0$. Furthermore, matrices $\mathbf{C}$ and $\mathbf{C}'$ are identical in `CM`, both being diagonal matrices whose (i,i)th element is a positive constant $\mu > 0$. Thus $\mathbf{C}^{-1} = \mathbf{C}'^{-1}$ and using Theorem 3.5, we obtain the following bound on the score-stability of the `CM` algorithm:

$$\beta_{\texttt{CM}} \leq \sqrt{2}M. \tag{3.26}$$

Local learning regularization ($\texttt{LL}-\texttt{Reg}$) In the $\texttt{LL}-\texttt{Reg}$ algorithm [105], the regularization matrix $\mathbf{Q}$ is $(\mathbf{I}-\mathbf{A})^\top(\mathbf{I}-\mathbf{A})$, where $\mathbf{I} \in \mathbb{R}^{(m+u)\times(m+u)}$ is an identity matrix and $\mathbf{A} \in \mathbb{R}^{(m+u)\times(m+u)}$ is a symmetric weight matrix that captures the local similarity between pairs of points in the full sample X. $\mathbf{A}$ is normalized; that is, each row (and column) of $\mathbf{A}$ sums to 1. The matrix $\mathbf{A}$ is computed before obtaining the actual partition of the full sample X into training and test sets, and it is not symmetric in general. Let $C_l, C_u > 0$ be two positive constants. The matrix $\mathbf{C}$ is a diagonal matrix with the ith entry being C_l if the ith example is in the training set and C_u otherwise. Let $C_{\max} = \max\{C_l, C_u\}$ and $C_{\min} = \min\{C_l, C_u\}$. Thus, $\|\mathbf{C}'^{-1} - \mathbf{C}^{-1}\|_2 = \sqrt{2}\left(\frac{1}{C_{\min}} - \frac{1}{C_{\max}}\right)$.

Since the sum of entries of each row of $\mathbf{A}$ is 1, by the Perron-Frobenius theorem its eigenvalues lie in the interval $(-1, 1]$ and one of its eigenvalues is 1. Thus, $\lambda_m(\mathbf{I}-\mathbf{A}) = 0$ and $\lambda_M(\mathbf{I}-\mathbf{A}) = 2$. Recall that $\mathbf{Q} = (\mathbf{I}-\mathbf{A})^\top(\mathbf{I}-\mathbf{A})$. This leads to $\lambda_m(\mathbf{Q}) = 0$ and $\lambda_M(\mathbf{Q}) = 4$ and we have the following bound on the score-stability of the $\texttt{LL}-\texttt{Reg}$ algorithm:

$$\begin{aligned}\beta_{\texttt{LL}-\texttt{Reg}} &\leq \sqrt{2}M + 4\sqrt{m}M\left(\frac{1}{C_{\min}} - \frac{1}{C_{\max}}\right) \\ &\leq \sqrt{2}M + \frac{4\sqrt{m}M}{C_{\min}}. \end{aligned} \tag{3.27}$$

The Gaussian Mean Fields algorithm [107] is very similar to the $\texttt{LL}-\texttt{Reg}$ algorithm, and admits exactly the same stability coefficient.

Thus, the stability coefficients of the algorithms in [105, 106, 107] are quite large and do not allow the generalization bound of Equation 3.10 to con-

verge. Without additional assumptions on the matrix $\mathbf{Q}$, these algorithms do not appear to be stable enough for the generalization bound (Theorem 3.3) to converge. We first present and analyze these stability bounds and later detail a general approach imposing additional constraints on the hypothesis that make the bound converge. A particular example of this is the constraint $\sum_{i=1}^{m+u} h(x_i) = 0$ used by Belkin's algorithm [8]. In the next section, we present a generalization bound for Belkin's algorithm and comment on "stable" versions of the algorithms analyzed in this section.

3.5.2 Stability of Constrained Regularization Algorithms

This subsection analyzes constrained regularization algorithms such as the Laplacian-based graph regularization algorithm of Belkin et al. [8]. Given a weighted graph $G = (X, E)$ in which edge weights represent the extent of similarity between vertices, the task consists of predicting the vertex labels. The hypothesis h returned by the algorithm is the solution of the following optimization problem:

$$\min_{h \in H} \mathbf{h}^\top \mathbf{L}\mathbf{h} + \frac{C}{m} \sum_{i=1}^{m} (h(x_i) - y_i)^2$$
$$\text{subject to: } \sum_{i=1}^{m+u} h(x_i) = 0,$$

where $L \in \mathbb{R}^{(m+u)\times(m+u)}$ is a smoothness matrix, for example the graph Laplacian, and $\{y_i \mid i \in [m]\}$ are the target values of the m labeled nodes.

The hypothesis set H in this case can be thought of as a hyperplane in $\mathbb{R}^{m+u}$ that is orthogonal to the vector $\mathbf{1} \in \mathbb{R}^{m+u}$. Maintaining the notation used in [8], we let P_H denote the operator corresponding to the orthogonal projection on H. For a sample S drawn without replacement from X, define $\mathbf{I}_S \in \mathbb{R}^{(m+u)\times(m+u)}$ to be the diagonal matrix with $[\mathbf{I}_S]_{i,i} = 1$ if $x_i \in S$ and 0 otherwise. Similarly, let $\mathbf{y}_S \in \mathbb{R}^{(m+u)\times 1}$ be the column vector with $[\mathbf{y}_S]_{i,1} = y_i$ if $x_i \in S$ and 0 otherwise. The closed-form solution on a training sample S is given by [8]:

$$\mathbf{h}_S = \left(P_H\left(\frac{m}{C}\mathbf{L} + \mathbf{I}_S\right)\right)^{-1}\mathbf{y}_S. \tag{3.28}$$

Theorem 3.6 *Assume that the vertex labels of the graph $G = (X, E)$ and the hypothesis h obtained by optimizing Equation 3.28 are both bounded ($\forall x, |h(x)| \leq M$ and $|y(x)| \leq M$ for some $M > 0$). Let $A = 1 + \kappa\sqrt{C}$. Then, for any $\delta > 0$, with probability at least $1 - \delta$,*

$$R(h) \leq \widehat{R}(h) + \beta + \left(2\beta + \frac{(AM)^2(m+u)}{mu}\right)\sqrt{\frac{\alpha(m,u)\ln\frac{1}{\delta}}{2}},$$

with $\beta \leq \frac{4\sqrt{2}M^2}{\frac{m}{C}\lambda_2 - 1} + \frac{4\sqrt{2m}M^2}{\left(\frac{m}{C}\lambda_2 - 1\right)^2}$, where λ_2 is the second smallest eigenvalue of the Laplacian matrix and $\alpha(m,u) = \frac{mu}{m+u-1/2} \cdot \frac{1}{1-1/(2\max\{m,u\})}$.

Proof. The proof is similar to that of [8] but uses our general transductive regression bound instead. □

The generalization bound we just presented differs in several respects from that of Belkin et al. [8]. Our bound explicitly depends on both m and u while

theirs shows a dependency on m only. Also, our bound does not depend on the number of times a point is sampled in the training set (parameter t), thanks to our analysis based on sampling without replacement.

Contrasting the stability coefficient of Belkin's algorithm with the stability coefficient of `LTR` (Theorem 3.4), we note that it does not depend on C' and β_{loc}. This is because unlabeled points do not enter the objective function, and thus $C' = 0$ and $\widetilde{y}(x) = 0$ for all $x \in X$. However, the stability does depend on the second smallest eigenvalue λ_2 and the bound diverges as λ_2 approaches $\frac{C}{m}$. In all our regression experiments, we observed that this algorithm does not perform as well as `LTR`.

3.5.3 Making Seemingly Unstable Algorithms Stable

In Section 3.5.2, we saw that imposing additional constraints on the hypothesis, for example, $\mathbf{h} \cdot \mathbf{1} = 0$, allowed one to derive non-trivial stability bounds. This idea can be generalized and similar non-trivial stability bounds can be derived for "stable" versions of the algorithms presented in Section 3.5.1: $\texttt{CM}, \texttt{LL} - \texttt{Reg}$, and `GMF`. Recall that the stability bound in Theorem 3.5 is inversely proportional to the smallest eigenvalue $\lambda_m(\mathbf{Q})$. The main difficulty with using the theorem for these algorithms is that $\lambda_m(\mathbf{Q}) = 0$ in each case. Let $\mathbf{v}_m$ denote the eigenvector corresponding to $\lambda_m(\mathbf{Q})$ and let λ_2 be the second smallest eigenvalue of $\mathbf{Q}$. One can modify (3.22) and constrain the solution to be orthogonal to $\mathbf{v}_m$ by imposing $\mathbf{h} \cdot \mathbf{v}_m = 0$. In the case of [8], $\mathbf{v}_m = \mathbf{1}$. This modification, motivated by the algorithm of [8], is equivalent to increasing the

smallest eigenvalue to be λ_2.

As an example, by imposing the additional constraint, we can show that the stability coefficient of `CM` becomes bounded by $O(C/\lambda_2)$, instead of $\Theta(1)$. Thus, if $C = O(1/m)$ and $\lambda_2 = \Omega(1)$, it is bounded by $O(1/m)$ and the generalization bound converges as $O(1/m)$.

3.6 Experiments

3.6.1 Model Selection Based on Bound

This section reports the results of experiments using our stability-based generalization bound for model selection for the `LTR` algorithm. A crucial parameter of this algorithm is the stability coefficient $\beta_{loc}(r)$ of the local algorithm, which computes pseudo-targets $\widetilde{y}_x$ based on a ball of radius r around each point. We derive an expression for $\beta_{loc}(r)$ and show, using extensive experiments with multiple data sets, that the value r^* minimizing the bound is a remarkably good estimate of the best r for the test error. This demonstrates the benefit of our generalization bound for model selection, avoiding the need for a held-out validation set.

The experiments were carried out on several publicly available regression data sets: *Boston Housing* (Figure 3.3), *Elevators* (Figure 3.2) and *Ailerons* (Figure 3.3)[2]. For each of these data sets, we used $m = u$, guided by the observation that, all other parameters being fixed, the bound of Theorem 3.3

[2] `www.liaad.up.pt/~ltorgo/Regression/DataSets.html`.

is tightest when $m = u$. The value of the input variables were normalized to have mean zero and variance one. For the Boston Housing data set, the total number of examples was 506. For the Elevators and the Ailerons data set, a random subset of 2000 examples was used. For both of these data sets, other random subsets of 2000 samples led to similar results. The Boston Housing experiments were repeated for 50 random partitions, while for the Elevators and the Ailerons data set, the experiments were repeated for 20 random partitions of each set. Since the target values for the Elevators and the Ailerons data set were extremely small, they were scaled by a factor 1000 and 100 respectively in a pre-processing step.

In our experiments, we estimated the pseudo-target of a point $x' \in T$ as a weighted average of the labeled points $x \in N(x')$ in a neighborhood of x'. Thus, $\widetilde{y}_{x'} = \sum_{x \in N(x')} \alpha_x y(x) / \sum_{x \in N(x')} \alpha_x$. We considered two weighting approaches, as discussed in [28], defining them in terms of the inverse of the distance between $\Phi(x)$ and $\Phi(x')$ (i.e. $\alpha_x = (1 + \|\Phi(x) - \Phi(x')\|)^{-1}$), and in terms of a similarity measure $K(x, x')$ captured by a kernel K (i.e. $\alpha_x = K(x, x')$). In our experiments, the two approaches produced similar results. We report the results of kernelized weighted average with a Gaussian kernel.

Lemma 3.7 *Let $r \geq 0$ be the radius of the ball around an unlabeled point $x' \in X$ that determines the neighborhood $N(x')$ of x' and let $m(r)$ be the number of labeled points in $N(x')$. Furthermore, assume that the values of the labels are bounded (i.e. $\forall x \in X, |y(x)| \leq M$ for some $M \geq 0$ and that all the weights in (3.6.1) are non-negative (i.e. $\forall x, \alpha_x \geq 0$). Then, the stability*

coefficient of the weighted average algorithm for determining the estimate of the unlabeled point x' is bounded by:

$$\beta_{loc} \leq \frac{4\alpha_{\max} M}{\alpha_{\min} m(r)}, \tag{3.29}$$

where $\alpha_{\max} = \max_{x \in N(x')} \alpha_x$ *and* $\alpha_{\min} = \min_{x \in N(x')} \alpha_x$.

Proof. We consider the change in the estimate as a point is removed from $N(x')$ and show that this is at most $\frac{2\alpha_{\max} M}{\alpha_{\min} m(r)}$. The statement of the lemma then follows straightforwardly from the observation that changing one point is equivalent to removing one point and adding another point.

Let $N(x') = \{x_1, \ldots, x_{m(r)}\}$. For ease of notation, assume that $n = m(r)$. Consider the effect of removing x_n from the neighborhood $N(x')$. The estimate changes by:

$$\frac{\sum_{i=1}^{n} \alpha_i y_i}{\sum_{i=1}^{n} \alpha_i} - \frac{\sum_{i=1}^{n-1} \alpha_i y_i}{\sum_{i=1}^{n-1} \alpha_i}.$$

Thus, the stability β_{loc} is bounded by:

$$\begin{aligned}
\beta_{loc} &\leq \left| \frac{\sum_{i=1}^{n} \alpha_i y_i}{\sum_{i=1}^{n} \alpha_i} - \frac{\sum_{i=1}^{n-1} \alpha_i y_i}{\sum_{i=1}^{n-1} \alpha_i} \right| \\
&\leq \frac{\alpha_n |y_n|}{\sum_{i=1}^{n} \alpha_i} + \sum_{i=1}^{n-1} \alpha_i |y_i| \left(\frac{1}{\sum_{i=1}^{n-1} \alpha_i} - \frac{1}{\sum_{i=1}^{n} \alpha_i} \right) \\
&= \frac{\alpha_n |y_n|}{\sum_{i=1}^{n} \alpha_i} + \frac{\sum_{i=1}^{n-1} \alpha_i |y_i|}{\sum_{i=1}^{n-1} \alpha_i} \cdot \frac{\alpha_n}{\sum_{i=1}^{n} \alpha_i} \\
&\leq \frac{2\alpha_n M}{\sum_{i=1}^{n} \alpha_i} \leq \frac{2\alpha_{\max} M}{\alpha_{\min} n} \leq \frac{2\alpha_{\max} M}{\alpha_{\min} m(r).}
\end{aligned}$$

□

Corollary 3.1 *Using the notation of Lemma 3.7, the stability coefficient of the kernelized weighted average algorithm with a Gaussian kernel K with parameter σ is bounded by:*

$$\beta_{loc} \leq \frac{4M}{m(r)e^{-2r^2/\sigma^2}}.$$

Proof. This follows directly from Lemma 3.7 using the observation that for a Gaussian kernel K, $K(x, x') \leq 1$, and for x, x' such that $\|x\| \leq r$ and $\|x'\| \leq r$, $\|x - x'\| \leq 2r$. Thus, $K(x, x') \geq e^{-2r^2/\sigma^2}$. □

Corollary 3.2 *Using the notation of Lemma 3.7, the stability coefficient of the weighted average algorithm, where weights are determined by the inverse of the distance in the feature space, i.e. $\alpha_x = (1 + \|\Phi(x) - \Phi(x')\|)^{-1}$ is bounded by:*[3]

$$\beta_{loc} \leq \frac{(2r+1)2M}{m(r)}.$$

Proof. Follows directly from Lemma 3.7 using the observation that $\forall x \in N(x')$,

$$0 \leq \|\Phi(x) - \Phi(x')\| \leq 2r.$$

□

To estimate β_{loc}, one needs an estimate of $m(r)$, the number of samples in a ball of radius r from an unlabeled point x'. In our experiments, we estimated

[3] 1 is added to the weight to make the weights between 0 and 1.

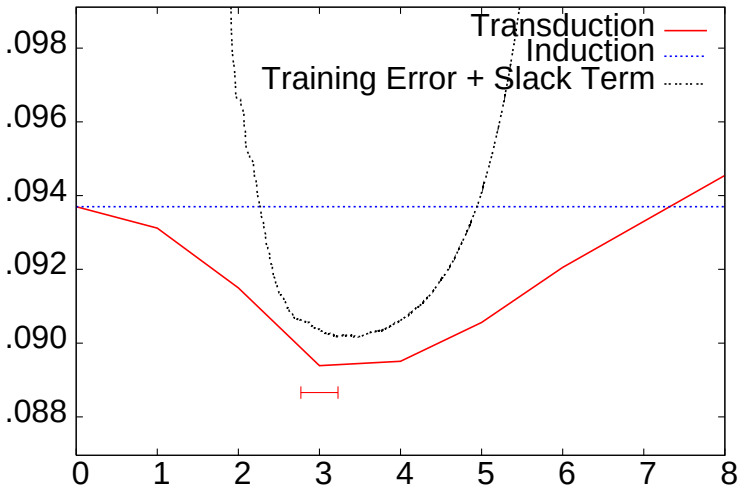


Figure 3.2: The mean-squared error (MSE) against the radius r of `LTR` for the Ailerons data set. The small horizontal bar indicates the location (mean $\pm$ one standard deviation) of the minimum of the empirically determined r.

$m(r)$ as the number of samples in a ball of radius r from the origin. Since all features are normalized to mean zero and variance one, the origin is also the centroid of the set X.

We implemented a dual solution of `LTR` and used Gaussian kernels, for which the parameter σ was selected using cross-validation on the training set. Experiments were repeated across 36 different pairs of values of (C, C'). For each pair, we varied the radius r of the neighborhood used to determine estimates from zero to the radius of the ball containing all points.

Figure 3.2 shows the mean values of the test mean-squared error of our experiments on the Ailerons data set for typical values of C and C' as the radius r of the neighborhood that determines pseudo-target values on the unlabeled

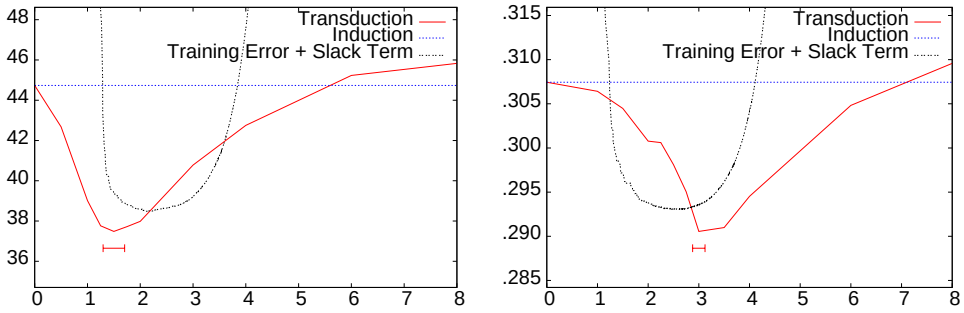


Figure 3.3: MSE against the radius r of LTR for two other data sets: (a) Boston Housing. (b) Elevators.

points is varied. As mentioned before, the objective of the experiment is to validate the stability-based generalization bound derived in Theorem 3.3. Recall that Theorem 3.3 tells us that with high probability (at least $1-\delta$),

$$R(h) \leq \widehat{R}(h) + \beta + \left(2\beta + \frac{B^2(m+u)}{mu}\right)\sqrt{\frac{\alpha(m,u)\ln\frac{1}{\delta}}{2}}, \tag{3.30}$$

where $\alpha(m,u) = \frac{mu}{m+u-1/2} \cdot \frac{1}{1-1/(2\max\{m,u\})}$, B is an upper-bound on $|h(x)-y(x)|$, β is the stability of the learning algorithm and m and u are the number of labeled and unlabeled points in X respectively. The curve labeled "training error plus slack term" corresponds to the expression on the RHS of Inequality 3.30, where β is the stability coefficient of LTR as determined in Theorem 3.4. The curve corresponding to "Transduction" (solid line) is $R(h)$. Only two terms depend upon the choice of the radius r: $\widehat{R}(h)$ and β_{loc}. Thus, keeping all other parameters fixed, the theoretically optimal radius r^* is the one that minimizes

the training error plus the slack term. Our experiments confirm that r^* is close to the empirically optimal one.

Figures 3.3(a)-(b) show similar results for the Boston Housing and Elevators data sets. For the sake of comparison, we also report results for induction. The relative standard deviations on the MSE are not indicated, but were typically on the order of 10%. `LTR` generally achieves a significant improvement over induction. The figures also include plots of the training error combined with the complexity term, appropriately scaled. The empirical minimization of the radius r coincides with or is close to $r*$. The optimal r based on test MSE is indicated with error bars.

3.7 Conclusion

We presented a comprehensive analysis of transductive regression algorithms based on the notion of stability. Since they are algorithm-dependent, our bounds are often tighter than those based on complexity measures such as the VC-dimension. Our experiments further show the effectiveness of our bounds for model selection and the good performance of `LTR` algorithms.

Chapter 4

An Analysis of Discrete Markets

4.1 Introduction

Economists have long been explaining some of the core workings of market economies via general equilibrium theory. In his book *An Inquiry into the Nature and Causes of the Wealth of Nations* [94], arguably the first modern work in the field of economics, Scottish economist Adam Smith suggested that the pursuit of self-interest tends to lead toward economic well-being and prosperity. Smith coined the expression *invisible hand* to refer to the market's ability to discover socially desirable outcomes. The first mathematical treatment seeking to model the evolution of prices in a market was made by the French economist Léon Walras in his seminal book, *Elements of Pure Economics* [104]. Since then, mathematical economists have made impressive progress in determining the conditions under which an equilibrium is guaranteed to exist [4]. Although

economists have sought to model the dynamics that lead markets to equilibrium prices, their emphasis has primarily been on establishing convergence. The complexity of the processes through which this happens (the efficiency of convergence), an important consideration in computer science, has remained largely unaddressed.

An equilibrium is provided by prices that balance supply and demand in all goods simultaneously. In order to reach equilibrium, prices need to adjust to demand imbalances. Walras suggested they do so by tatonnement: if there is too little demand for a good its price drops, if there is too much demand its price rises. He also proposed a formal model in which this could occur: the *auctioneer* model. There is an auctioneer who sets the prices. To do this, she announces current prices, receives the resulting demands, compares them to supplies, adjusts the prices (by tatonnement) and iterates. Subsequently, several studies [86, 100] sought to understand the dynamics of tatonnement-based convergence of prices to equilibrium, though the focus was merely to establish *some* convergence. These studies of pricing updates implicitly or explicitly take the view that there is a virtual price setter for each good that encapsulates the collective actions of individual buyers and sellers in the market. We too follow this viewpoint in this chapter.

A significant motivation for the recent algorithmic study of equilibrium theory is the viewpoint, first articulated by Papadimitriou, that equilibria, or at least approximate equilibria, need to be efficiently computable for equilibrium theory to be meaningful. This also suggests, as pointed out by Cole

and Fleischer [25], that equilibria need to be efficiently computable *within the market itself*.

If equilibria are to be discovered within the market itself, then it is reasonable to assume that updates to prices are responses to excess demand, for it seems plausible that buyers reveal their demand, but not the underlying utilities driving the demand. We call this demand-driven pricing. Under demand driven pricing, goods are allocated to buyers only if they demand the good (i.e. there is no substitution of second best choices). This captures algorithms which assume a demand oracle, for example the tatonnement algorithms such as [5, 50, 52], but excludes algorithms where full knowledge of the buyer's utilities is needed, such as [55, 56, 59].

The standard formulation of a market is called an Arrow-Debreu market, named after the nobel-prize winning economists Kenneth Arrow and Gerard Debreu. In Arrow-Debreu markets each agent is provided with an initial allocation of goods. The problem is to find prices at which every agent can trade its initial allocation for an optimal bundle in such a way that supply and demand balance exactly. In this chapter, we limit ourselves to the special case of Fisher markets. In these markets, there are two groups of agents: buyers, who initially have only money, and sellers, who have only goods. Each seller's goal is to sell all its goods. Each buyer's goal is to utilize its money to obtain an optimal bundle of goods. In fact, we use a small variant of the Fisher model in which buyers may attach utility to money, and thus an optimal allocation

may leave them with non-zero money at the end.[1] This strikes us as reflecting actual behavior, and was previously considered by Devanur and Vazirani who proposed putting limits on spending and also incorporating a desire for money in the context of utility functions that are linear step functions [41]. The possible inclusion of money in the utility function has received attention in the economics literature also, e.g. [15].

In the divisible setting, where all goods are infinitely divisible and prices can take on any real value, under modest assumptions, equilibrium prices always exist [4]. This need not be the case in an indivisible setting. The task then becomes one of finding prices that support a near-minimally unbalanced allocation (a near-equilibrium allocation, for short).

Seeking to understand how actions of the price setters and buyers in the market lead them to discover (near-)equilibrium prices bears some resemblance to the following more general problem: how do selfish independent agents in a multiplayer noncooperative game adjust their strategies to converge to a Nash equilibrium [6].

This chapter studies the interplay of three issues: indivisibility of goods and money, the hardness of finding approximate equilibrium prices, and demand-driven pricings and allocations. This then brings up two issues that arise due to indivisibility, namely how to measure the approximation quality, and the fact that a very standard assumption on the behavior of demand as prices

[1] Of course, all buyers could continue to have no desire for money; thus this is a generalization of the standard model.

vary (the weak gross substitutes property, defined later) appears to be unduly restrictive in the indivisible setting.

There are two issues when faced with computing an approximate equilibrium. The first is to determine prices and the second is to allocate goods, for given that the equilibrium is only approximate, there will not be a perfect allocation. Deng, Papadimitriou and Safra [40] showed that the allocation problem is APX-hard. Given our interest in what a market can do, we believe that the critical question is how hard it is to set prices, for this is presumably what the markets adjust based on the supply and demand imbalances, whereas it seems less plausible that the market optimizes among the allocations at a given price. So, by contrast with Deng et al. [40], we show that choosing optimal prices and not just optimal allocations is NP-hard. Note that in the construction of [40], optimal prices are easily computed.

There has also been a considerable body of work giving polynomial time algorithms to compute exact and approximate equilibria for divisible markets [18, 19, 21, 23, 40, 50, 56, 57, 58]. Of course, all markets are necessarily indivisible. While one anticipates large numbers of copies of each good, and prices with a minimum adjustment that is a small fraction of the price, it remains the case that both these quantities are bounded. In our view, an implicit assumption in studying the divisible problem is that the discreteness present in actual markets has only a small effect. As in practice the number of copies of a good need not be much greater than the number of interested buyers, it does not seem that the assumption is sound in general. Nonetheless, we show that

in the markets we consider, the hardness moderates as the amount of money increases (intuitively, as money becomes more divisible).

Deng et al. [40] also considered algorithms for indivisible markets. They gave an exhaustive algorithm for computing an approximate equilibrium in polynomial time for markets with a constant number of distinct goods.

In [25], Cole and Fleischer focus on tatonnement-style price update protocols that yield rapid convergence in divisible markets. In this chapter, we focus on the discrete versions of what they call the *One-Time* market, which is essentially the auctioneer model of tatonnement.

Notation Let $\mathbf{p} = (p_1, \ldots, p_n)$ be a collection of prices. Whenever convenient, we write $\mathbf{p} = (p_i, \mathbf{p}_{-i})$, where $\mathbf{p}_{-i}$ is the collection of prices for all goods except good i. Further, we let $x_i(\mathbf{p})$ denote the demand for good i at $\mathbf{p}$.

For a demand driven algorithm to achieve rapid convergence it seems necessary that a small change in prices lead to only a small change in demand. This observation led Cole and Fleischer [25] to parameterize the markets they considered in terms of a bound, E, on the price elasticity of demand (see [66], page 27). It is defined as the fractional rate of of change of demand with respect to price: $\frac{\partial x_i(\mathbf{p})/\partial p_i}{x_i/p_i}$. Under the assumption of weak gross substitutes (a market obeys weak gross substitutes if when the price of one good increases, the demand for other goods does not decrease while the good's own demand only decreases), this is negative. The parameter E is an upper bound on the

absolute value of this quantity over all goods and all prices:

$$E = -\min_{i,\mathbf{p}} \frac{\partial x_i(\mathbf{p})/\partial p_i}{x_i/p_i}. \tag{4.1}$$

In general E could be unbounded (e.g., when all the buyers have identical linear utility functions). Intuitively, it is clear that the larger E is, the smaller the price adjustments should be for a given level of excess demand.

We define a discrete version of this parameter, which we called discrete bounded elasticity.

Definition 4.1 (Bounded Elasticity E) *A discrete market is said to have bounded elasticity E if for every good i, for all prices $\mathbf{p}$, for all $h > 0$ such that $x_i(p_i + h, \mathbf{p}_{-i}) \geq 1$,*

$$x_i(p_i, \mathbf{p}_{-i}) \leq \lceil x_i(p_i + h, \mathbf{p}_{-i})\,(1 + h/p_i)^E \rceil,$$

and if $x_i(p_i + 1, \mathbf{p}_{-i}) = 0$, then $x_i(p_i, \mathbf{p}_{-i}) \leq 1$.

Note that $x_i(p_i, \mathbf{p}_{-i}) \leq 2^E x_i(2p_i, \mathbf{p}_{-i})$ if $x_i(2p_i, \mathbf{p}_{-i}) \geq 1$.

Another major change in the discrete setting, at least with money being discrete, is that there are no obvious conditions guaranteeing the existence of equilibria. However, with continuous money, equilibria exist in the settings we consider. We introduce a second parameter, r, which indicates the degree of divisibility of money; it is defined later.

The issue of how to measure the quality of an approximate equilibrium is

far from clear cut. The previous literature in computer science measured it by considering each agent in turn and asking what is the ratio between its actual utility and the maximum utility it could conceivably achieve at the prices on offer; the quality of the approximation is then defined to be the minimum of these ratios over all agents (0 is the worst possible, 1 the best). We offer two critiques of this approach.

The first is that utilities are being treated implicitly as if they were valuation functions (i.e. as if they assign dollar amounts to each bundle). But this is ascribing more meaning to the function than may be appropriate, since in general there are multiple utility functions that represent a given well behaved preference ordering. Indeed, if u is a utility function and f is a strictly increasing function, then $f(u)$ provides an alternative utility function giving the same preference ordering.[2] This is disconcerting, for a $(1-\epsilon)$-approximation under one utility function may be a much better or worse quality approximation for another equivalent utility function. Example 4.1 illustrates these two issues.

Our second critique concerns how the utilities are combined. In our view, the quality of the approximation ought to correspond to how dissatisfied the market participants are collectively; surely participants with large resources will have a larger impact. This suggests an approach other than minimizing relative discontent. Instead, we seek to combine the discontent of each of the participants. But this necessitates expressing discontent in a common unit.

[2]When studying choice under risk (for example in auctions), the concavity of u defines agents' risk aversion. In such cases, the utility function may be unique. However, this is not the case in a market setting, where we concern ourselves with deterministic outcomes.

We do this using the notion of compensatory payments from welfare economics: essentially, this asks what is the difference in value to the agent between the agent's optimal allocation at the current prices and the allocation the agent receives. We call this quantity, measured in the unit of money, the agent's *discontent*.

The following example illustrates the shortcomings of measuring the quality of an approximate equilibrium by considering the utility ratios of the most dissatisfied participant.

Example 4.1 *Consider the following scenario. In a market with 3 agents, at an announced set of (non-equilibrium) prices, sub-optimal allocations with utilities as specified in the following table are made.*

	Actual Utility	*Optimal Utility*	*"Happiness"*
Agent 1, $u_1(\cdot)$	*80*	*100*	*80%*
Agent 2, $u_2(\cdot)$	*20*	*50*	*40%*
Agent 3, $u_3(\cdot)$	*90*	*100*	*90%*

The most dissatisfied participant is 40% "happy", so the current prices are 0.4-close to equilibrium.

Now suppose that the same set of buyers have utilities $u_i^2(\cdot)$ *instead of* $u_i(\cdot)$, $i \in \{1, 2, 3\}$. *Their preference among different allocations is the same as before.*

	Actual Utility	*Optimal Utility*	*"Happiness"*
Agent 1, $u_1^2(\cdot)$	*6400*	*10000*	*64%*
Agent 2, $u_2^2(\cdot)$	*400*	*2500*	*16%*
Agent 3, $u_3^2(\cdot)$	*8100*	*10000*	*81%*

The most dissatisfied participant is 16% "happy" now, and the current prices are 0.16-close to equilibrium, even though neither the prices, nor the allocations to individual market participants have changed.

By calculating the discontent of the individual market participants appropriately, we show how to put this on a sound footing and use the term efficiency for the resulting measure.

Nonetheless, in a demand driven setting, where demands are only indirectly based on utilities, and are not influenced by the degree of disutility, it does not appear possible to determine efficiency exactly based on excess demands alone (as opposed, for example, to obtaining loose upper bounds). As a result, we also consider the simpler measures of relative prices and misspending as was done by Cole and Fleischer.

We review the definition of the Weak Gross Substitutes (WGS) property. Recall that a market obeys WGS if when the price of one good increases, the demand for other goods only increases (i.e. stays the same or increases) while its demand only decreases. As we will see, in the discrete setting, this property is not observed by many natural utility functions including linear and CES utility functions, leading us to introduce a generalized relaxed WGS property.

Summary of results. Our main results are:

- In a market with bounded elasticity E satisfying the relaxed WGS property it is NP-hard to find prices at which there is a demand-driven allocation achieving a $1 - \Theta((E-1)/r)$-approximation (in a sense made precise later). The hardness lies in finding the prices, not the allocation. To the best of our knowledge, this is the first result demonstrating that it is hard to find correct prices as opposed to a correct allocation. We also remark that in the present setting it seems much more delicate to obtain the hardness result for prices.

- A complementary P-time algorithm that achieves $1-O(E/r)$-approximation. The algorithm assumes a polynomial time oracle which, given the prices for every good, returns the demand for each good. The algorithm makes $O(Etn \log w)$ calls to the oracle, where n is the number of distinct goods in the market, w is the total buyer wealth, and t is the total number of copies of goods in the market; the oracle calls dominate the overall running time. (If $t \gg r$, the running time can be made linear in r rather than t. The running time can also be reduced proportionately if a less good approximation is sought). An appealing aspect of the algorithm is its simplicity.

- A partial analysis of the convergence of a tatonnement-based local, distributed price update protocol in which the price of each good is updated asynchronously and independently. Here, we show that the prices rapidly

converge to an interval around near-equilibrium prices.

Note that the average wealth r intuitively captures the extent of divisibility of money in the market, while E is the elasticity. For the rest of this chapter, we assume that $r \geq E$.

As mentioned earlier, a market with bounded elasticity E is one in which the rate of change of demand for a good with respect to its own price is bounded. Assuming that the demand function for every good has a bounded partial derivative with respect to every price, Codenotti et al. [23] gave strongly polynomial time algorithms for finding approximate equilibria using a procedure similar to tatonnement. The algorithm of Codenotti et al. uses the ellipsoid method as a subroutine; this contrasts with the simplicity of our method. Of course the algorithms are not directly comparable as they are for different settings, and as we discuss below, use different approximation measures. Another point to keep in mind is that a rounded optimal or near-optimal solution for the divisible setting need not be a near-optimal solution in the indivisible setting (assuming analogous problems can be defined). This can occur, for example, when there are goods for which buyers have $o(1)$ demand, even if there are many copies of the good.

Bounded elasticity constrains the behavior of the overall demand in the market. Without some constraints, equilibria need not be unique and can be hard to compute [24]. Most previous algorithmic work in the divisible setting overcame this by constraining individual utility functions (e.g. to satisfy the WGS property, or to be linear [23, 21]). However, this can still result in some

quite unintuitive behavior. For example, if all the buyers had identical linear utility functions, a minimal change in prices can shift the demand from being all for one good to all for another good. Bounded elasticity allows linear utilities, but it limits the amount of a pair of goods that are equally desired at a given set of prices (thereby avoiding large fluctuations in demand with small changes in price), and so in particular precludes all buyers having identical linear utilities.

Indivisible markets have been studied by mathematical economists also. Ausubel, Gul and Stacchetti [5, 52] restrict the utility functions being allowed so as to ensure that equilibria exist. In fact, they suppose the buyers have valuation functions that are integer valued, i.e. each basket of goods has an associated dollar value. They show that if these functions also satisfy an "individual substitutes" property, then there is a Walrasian equilibrium. The individual substitutes condition is a further restriction of the well known gross substitutes condition. It requires separate prices for distinct copies of the same good and that individual copies of the same good also obey the gross substitutes property. Given these restrictions, they show that the minima of a certain potential function (a Lyapunov function) correspond to equilibrium prices. They propose a tatonnement algorithm that updates prices at discrete time steps in the direction of decreasing potential function values, which is found by exhaustively testing every price point in a unit neighborhood of the current price. The algorithm converges to equilibrium prices, but only in exponential time.

In [71] Milgrom and Strulovici consider the above setting, replacing the individual substitutes constraint with the standard WGS constraint. As an equilibrium may then not exist they propose a notion of pseudo-equilibrium. The authors argue that a pseudo-equilibrium price is also an *approximate* equilibrium price by showing that the excess demand (which should be zero at equilibrium prices) can be bounded by a function of the number of goods, the number of buyers and the largest *gap* in demand for each good among optimal bundles at any price. They also give an exhaustive algorithm. This approach is in contrast with our work which seeks to estimate individual discontent in terms of money and aggregate the overall discontent in the unit of money.

In the divisible setting, it is well known that the market obeys WGS when buyers in the market possess linear, Cobb-Douglas or CES utility functions with $0 < \rho < 1$. A CES utility function orders a buyer's preferences over bundles of goods $\mathbf{x}$ according to the value $u(\mathbf{x}) = (\sum_i \alpha_i x_i^\rho)^{1/\rho}$, with $-\infty < \rho < 1, \rho \neq 0$. The linear utility function and the Cobb-Douglas utility function arise as special cases of the CES utility function when $\rho = 1$ and $\rho \to 0$ respectively. However the WGS property need not hold in the indivisible setting and the indivisibility can result in quite unintuitive behavior, as we illustrate later via an example. To account for such effects, we propose a relaxation of the WGS property. Our algorithm and the hardness result assume that the overall market obeys the relaxed WGS property.

Next, we discuss how to measure the quality of an approximate equilibrium. It is natural to ask what is the relationship between an individual's discontent

in our measure and that given by comparing utility values. We show that in the divisible setting, for concave utility functions, for a given allocation, the approximation factor is always at least as large in our measure, and can be arbitrarily worse. In other words, for each individual, a given approximation factor is at least as hard to achieve in our measure as in the utility measure.

Once one has a way to measure the individual discontent of each participant, the next issue is to aggregate individual discontents to represent the overall discontent of the market. Here, the fact that our measure of individual discontent is in the common unit of money allows us to simply combine discontents additively.

This can be viewed as an L_1 norm; our hardness results carry over to any L_p norm, $p > 1$, and to the L_∞ or maximum norm, but with the approximation factor $1 - \Theta((E-1)^{1/p}/r)$. Loosely speaking, the L_∞ norm is analogous to but not the same as the previous approximation measure. The algorithmic bound remains at $1 - O(E/r)$; a bound of $1 - O(E^{1/p}/r)$ applies if we can assume that for each good any unsold copies are evenly spread among E or more sellers. The fact that trivial changes to the distribution of unsold goods can sharply change the discontent in L_p norms, $p > 1$, suggests to us that the L_1 norm is the most robust of these measures. We remark that the Deng et al. inapproximability bound does not appear to extend to the norms considered here.

We conclude the introduction by summarizing our contributions. An important part of the study lies in the definitions it introduces: bounded elastic-

ity, discontent and relaxed WGS. In our view, bounded elasticity provides a natural way to specify well-behaved markets, and the definition of discontent supports a more robust notion of approximation of equilibria than the previous approach. We also characterize the complexity of computing equilibria in these markets via an NP-hardness result and a complementary simple P-time algorithm that demonstrates that our hardness result is tight. We then present some preliminary results on the convergence of a local distributed tatonnement algorithm. This analysis is similar in spirit to the one presented by Cole and Fleischer [25] for the continuous case.

In Section 4.2, we provide formal definitions. Section 4.3 gives the hardness result and Section 4.4 the algorithm. Finally, in Section 4.6, we comment on the relationship between our proposed definition of discontent and ϵ-closeness in utility.

4.2 Definitions

Definition 4.2 (Utility Function) *Let G denote the set of goods present in the market. Then $u : \mathbb{R}^G \to \mathbb{R}^+$ is said to be a utility function, if, for $A_1, A_2 \subseteq G$, the allocation A_1 is strictly preferred to A_2 exactly when $u(A_1) > u(A_2)$.*

Comment 4.1 *In the auction literature, utility is often defined to be the difference between valuation and price; this is* not *the meaning intended here.*

Definition 4.3 (The Market) *The market consists of a set G of n goods in supply* $\mathbf{s}$ *(s_g copies of good g for $g \in G$), a set B of m buyers, buyer wealth* $\mathbf{w}$

(an initial amount w_b for $b \in B$). Each buyer b possesses a utility function u_b over the basket of goods. $(G, \mathbf{s}, B, \mathbf{w}, (u_1, \ldots, u_m))$ denotes an instance of the market.

Comment 4.2 *In addition to the indivisibility of goods, we also impose indivisibility of money in the market. Thus, money is no longer* fluid*, and prices can be set only at indivisible integer values. Henceforth, for specificity, we take the unit of money to be a dollar.*

Given a market M, the problem is to find prices, called market-clearing prices, such that at these prices each buyer receives its optimal allocation and no goods are left unsold (i.e. all sellers are also optimally happy). Formally, this can be written as:

Indivisible extended Fisher market problem. Given a market M, determine prices $\mathbf{p}$ such that there exists a partitioning of goods to buyers, with buyer b receiving A_b, $A_b \cdot \mathbf{p} \leq w_b$, $\bigcup_b A_b = G$, and for each b, A_b maximizes b's utility: for all A'_b such that $A'_b \cdot \mathbf{p} \leq w_b, u_b(A_b, w_b - A_b \cdot \mathbf{p}) \geq u_b(A'_b, w_b - A'_b \cdot \mathbf{p})$.[3]

Given a set of prices, for each buyer there is a basket of goods which maximizes its utility. An optimal allocation is one which maximizes every buyer's utility simultaneously and leaves no goods unsold. In the indivisible setting, in general, there need not be any price collection at which an optimal

[3]In computing utility, we view money as just another good. However, to conform with the usual perspective, we list it separately from the other goods.

allocation exists, which raises the question of how far from optimal a given allocation is. To this end, we define a notion of individual discontent.

Our approach is based on the notion of compensatory payments from welfare economics. It asks what payment an agent a needs to receive to compensate for a non-optimal allocation. Previously, this was defined for the divisible setting, as follows. Let A^{opt} be an optimal allocation, A^{act} the actual allocation, and A^l a least cost allocation with $u(A^l) = u(A^{act})$. Note that the choice of A^l depends on the prices and on agent a's utility function. Then the compensation agent a needs is defined to be $\text{cost}(A^{opt}) - \text{cost}(A^l)$. We call $\text{cost}(A^l)$ the *value* of A^{act} for agent a at the current prices. Figure 4.1 illustrates this notion of value.

Our approach is analogous. For each agent a (buyer or seller), for each amount of money, \$1, \$2, ..., \$$i$, ..., we consider an optimal allocation A_a^i of goods and m_a^i of money, its utility $u_a(A_a^i, m_a^i)$ and its cost for agent a, $A_a^i \cdot \mathbf{p} + m_a^i = i$ at the given prices. It may be that $A_a^i = A_a^{i+1}$ and $u_a(A_a^i, m_a^i) = u_a(A_a^{i+1}, m_a^{i+1})$ for some i (this can occur only if agent a has no utility for money, and there is nothing useful to purchase with the last dollar). To find the *value* of an allocation A with money m_a (written as (A, m_a)) to agent a, we find the least i such that either $u_a(A_a^i, i - A_a^i \cdot \mathbf{p}) = u_a(A, m_a)$ or $u_a(A_a^i, i - A_a^i \cdot \mathbf{p}) < u_a(A, m_a) < u_a(A_a^{i+1}, i + 1 - A_a^{i+1} \cdot \mathbf{p})$. In the former case, the value of (A, m_a) is \$$i$, whereas in the latter case, the value of (A, m_a) is defined to lie between \$$i$ and \$$i + 1$. In the second case, given that money is indivisible, it does not appear possible to define the value more precisely; in this situation \$$i$ is called

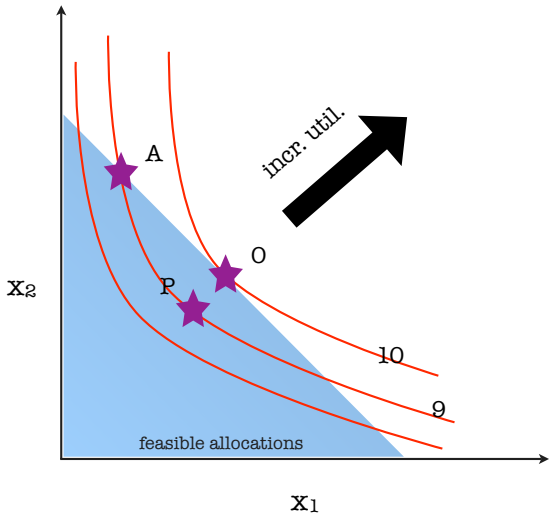


Figure 4.1: An illustration of the notion of value. Consider a buyer interested in two goods. Equal utility contours as a function of the buyer's allocation, (x_1, x_2), are shown. The shaded region denotes the set of allocations that can be afforded by the buyer at the announced set of prices. The optimal allocation, O, yields a utility of 10. The value of a sub-optimal allocation A that has a utility of 9 is the cost of the cheapest allocation yielding the same utility, i.e. the cost of allocation P.

the *lower-value* and $\$i + 1$ the *upper-value* of (A, m_a) at prices $\mathbf{p}$. When the agent is a buyer (say buyer b with wealth w_b), we let $v_b((A, m_b), \mathbf{p})$ denote the *upper-value* of allocation A at prices $\mathbf{p}$, defined to be the upper value of allocation A together with leftover money $m_b := w_b - A \cdot \mathbf{p}$. Note that for the optimal allocation, the upper value is the same as the lower value.

Henceforth, in scenarios in which a buyer has no marginal utility for money (i.e. adding money to a buyer's allocation does not improve her utility), we use the term optimal allocation to mean a maximum utility allocation of least cost.

Definition 4.4 (Discontent) *Let* $(A, w_a - A \cdot \mathbf{p})$ *be an allocation and* (A^{opt}, m^{opt}) *an optimal allocation (both including money) to agent* a *with wealth* w_a *at prices* $\mathbf{p}$. *The* upper-discontent *of* a *with allocation* $(A, w_a - A \cdot \mathbf{p})$ *at prices* $\mathbf{p}$, $ud_a((A, w_a - A \cdot \mathbf{p}), \mathbf{p})$, *is given by* $value_a((A^{opt}, m^{opt}), \mathbf{p}) - lower\text{-}value_a((A, w_a - A \cdot \mathbf{p}), \mathbf{p})$. Lower-discontent, ld_a, *is defined analogously. We use the notation* d_a *for short when no ambiguity will result.*

We observe that the seller discontent has a very simple form. First, we assume, without loss of generality, that there is a single distinct seller for each distinct good. If there are c_g copies of good g unsold at price p_g, the seller v_g for good g has a discontent of $d_{v_g}(c_g, \mathbf{p}) = c_g p_g$.

Note that upper-discontent minus lower-discontent is either \$0 or \$1. If they are equal, they are both called *discontent* for short. Further, for buyer b with wealth w_b, both lower and upper discontent are upper bounded by $v_b((A^{opt}, m^{opt}), \mathbf{p})$. Note that $m^{opt} \leq w_b - A^{opt} \cdot \mathbf{p}$, with equality when b has a desire for every remaining dollar in the leftover money. In our algorithms, we will use upper discontent as a worst case measure of discontent, while for the hardness results, we will use lower-discontent, as it provides a lower bound.

From an agent's point of view, how well it is doing, its *efficiency*, is the ratio of value achieved (lower-value say) to the value of its optimal allocation.

We define *market efficiency* as the ratio of the utility derived by all the agents compared to what appears possible individually at the offered prices. More precisely:

Definition 4.5 (Market Efficiency) *The market efficiency of an allocation A under prices $\mathbf{p}$ is defined as:*

$$1 - \frac{\sum_{a \in \mathcal{A}} d_a((A_a, w_a - A_a \cdot \mathbf{p}), \mathbf{p})}{\sum_{a \in \mathcal{A}} v_a((A_a^{opt}, m_a^{opt}), \mathbf{p})}$$

where $\mathcal{A}$ denote the set of agents in the markets (all buyers and sellers).

Another view of market efficiency is that it is simply a weighted average of the agent efficiencies, where an agent's weight is the value of its optimal allocation.

If all prices are zero, the market efficiency is not well defined (it is $1 - 0/0$). To avoid this difficulty, we limit the allowable prices to be strictly positive. Further, note that if the market has efficiency 1, then an equilibrium has been achieved, for the sellers will have sold all their goods and every buyer will have an optimal allocation. Conversely, if an equilibrium has been achieved, then the market has an efficiency of 1.

Problem 4.1 (Indivisible Market Value Problem) *Given a market M with parameters $(G, \mathbf{s}, B, \mathbf{w}, (u_1, \ldots, u_m))$, determine prices $\mathbf{p}$ and an allocation of goods to buyers that maximizes the market efficiency.*

Our results are parameterized by r, the average wealth per unit good. Formally, $r = \frac{\sum_{b \in B} w_b}{\sum_{g \in G} s_g}$.

The relaxed WGS property. As already noted, many widely studied utility functions that obey the WGS property in the divisible case do not do so in

the indivisible setting. The following example illustrates this counter-intuitive behavior.

Example 4.2 *Let B be a buyer with initial wealth* \$10 *in a market with goods G_1 and G_2, and let n_i denote B_i's demand for G_i, $i = 1, 2$. Suppose B has a linear utility $u(n_1, n_2) = 11n_1 + n_2$. At prices $p_1 = 10$, $p_2 = 1$, B's optimal bundle is $(n_1, n_2) = (1, 0)$, while at prices $p_1 = 9$, $p_2 = 1$, it is $(1, 1)$. This increase in demand for G_2 on reducing p_1 violates WGS.*

One could think of G_2 as being money in this example, though the same construction works for larger p_2 also. Similar constructions are possible for CES utility functions with $0 < \rho < 1$.

We relax the WGS property as follows to allow the demand for money to possibly increase when the price of a good drops, but this increase in demand is strictly upper bounded by the new (lower) price of the good.

Definition 4.6 (Relaxed WGS) *A market obeys the relaxed WGS property if when the price of a good G_i drops from p_i to p_i', the demand for G_i only increases, the demand for every other good apart from money only decreases, and the demand for money increases by less than p_i', i.e. $n_\$(\mathbf{p}_{-i}, p_i') < n_\$(\mathbf{p}_{-i}, p_i) + p_i'$.*

This definition could also be made w.r.t. individual buyers' demands, but in general this would give more slack in the market as a whole creating a less useful constraint.

Example 4.2 suggests that in the indivisible setting buyers may well have leftover money, and this gives a further reason to assume money has utility. For otherwise, contrary to intuition, it would seem a buyer would not have a preference for more leftover money.

A different relaxation of WGS would be to apply it to all the goods apart from money. This seems a natural approach if money has no utility, and even if money has utility, it seems defensible, for otherwise, as the price of a good G drops, spending on, and not just demand for G only increases (of course, this is what happens in the divisible setting in Fisher markets). However, as it has no effect on the algorithmic result, and is the more demanding condition for the hardness result, we choose to use Definition 4.6, above.

Finally, one might wonder how the specific assumptions affect our results.

1. If we allow arbitrary allocations in the hardness result, we can show that a $1 - \Theta(\min\{1, E-1\}/r)$-approximation is NP-hard (with a somewhat simplified construction).

2. If we replace the Relaxed WGS constraint with the stricter WGS, then it does not seem possible to combine it with bounded elasticity E.

4.3 Hardness of Computing Near-Equilibrium Prices

Our reduction will be from the balanced Max-3SAT-3 problem.

Definition 4.7 (Balanced Max-3SAT-3 problem.) Input. *A CNF formula with n variables in which each clause contains at least 2 and at most 3 literals, and every literal appears in exactly 3 clauses.*

Output. *The maximum number of clauses that can be satisfied simultaneously by some assignment to the variables.*

In the Boolean formula, we let n and m denote the number of variables and the number of clauses, respectively. Note that in a balanced 3SAT-3 formula, $m = \Theta(n)$. For ease of exposition, we assume that all clauses contain exactly 3 literals. The construction is readily modified when some of the clauses contain 2 literals instead of 3.

Given a balanced 3SAT-3 formula, we construct a market with bounded elasticity E that has $\Theta(mr^2)$ wealth, $\Theta(r)$ wealth per item, in which it is NP-hard to find if there are prices that support a market-based allocation with $O(m(E-1)r)$ discontent, while if the corresponding formula is satisfiable, there exists a pricing that supports an allocation with $O(km)$ discontent.

For each variable x in the boolean formula, we create goods $G_x, G_{\overline{x}}$ (corresponding to the positive and negative literals of x) and two helper goods F_x and D_x. When no ambiguity arises we let $G, \overline{G}, F, D$ denote $G_x, \overline{G}_x, F_x, D_x$, respectively. For the remainder of this section, we let p_E denote the price of good E; we will also let $p_x, p_{\overline{x}}$ denote the prices of $G_x, G_{\overline{x}}$, when the variable name is significant.

Our interpretation of the pricing is that $x = \mathit{True}$ if $p_G > r$ and $p_{\overline{G}} \leq r$. If both $p_{\overline{G}}, p_G > r$, this does not correspond to a truth assignment. Finally, if

both $p_{\overline{G}}, p_G \leq r$, this does not correspond to a truth assignment, but we treat it as the "assignment" in which both x and $\overline{x}$ are false. We will also need to take note of the values of p_F and p_D, but we will specify these later.

Notation: We let k denote $\lceil E \rceil$. Since we are concerned with $E > 1$, $k \geq 2$.

Our construction depends on creating two pools of discontent.

Pricing Pool: For each variable x, if the prices do not correspond to a truth assignment there will be at least $kr/8$ discontent.

Satisfying Pool: For each clause $C = a \vee b \vee c$, if $p_a, p_b, p_c \leq r$ (and all the prices of the corresponding F and D goods correspond to a truth assignment), there is at least $kr/8$ discontent.

We can now state the main result.

Lemma 4.1 *Consider a pricing. Suppose there are n_1 variables for which the pricing does not correspond to a truth assignment. Suppose there are m_2 clauses $a \vee b \vee c$ containing only literals for which the corresponding goods' prices correspond to a truth assignment of false. Then there is discontent of at least $(n_1 + m_2)kr/8$.*

Proof. This is immediate from the definition of the two pools. □

Corollary 4.1 *If any truth assignment leaves αm clauses unsatisfied for some constant $\alpha > 0$, then any pricing has discontent βmkr, with $\beta = \alpha/48$.*

Proof. Suppose there are n_1 variables for which the prices of the corresponding goods do not correspond to a truth assignment. These variables appear in at

most $6n_1$ clauses. Thus, if $n_1 \geq \alpha m/6$, the discontent is at least $n_1 kr/8 \geq \alpha mkr/48$, and if $n_1 < \alpha m/6$, it is at least $[n_1 + (\alpha m - 6n_1)]kr/8 \geq \alpha mkr/48$. □

The positive side of the construction is to show that if there is a satisfying assignment, then there exists a pricing with $O(Em)$ discontent. As the overall wealth is $O(Emr^2/(E-1))$, this yields:

Theorem 4.1 *There is a fixed $\beta > 0$, such that in a market with bounded elasticity E satisfying relaxed WGS it is NP-hard to compute prices under which a market-based allocation achieves an efficiency of $1 - \frac{\beta(E-1)}{r}$, even when there exist prices supporting a market-based allocation with efficiency $1 - \frac{\gamma(E-1)}{r^2}$, for a fixed $\gamma > 0$, for any fixed $E > 1$, where $r \geq \Theta(\max\{E, E/(E-1)\})$ is the wealth per item in the market.*

The difficulty in creating the discontent pools is that we appear to need to avoid seller discontent when p_D and p_F are "correctly" set. This implies that when $p_G = r$ (and $p_{\overline{G}} = r + 1$) there will be excess demand for G, which needs to be borne with low discontent.[4] Yet, if both $p_G = r$ and $p_{\overline{G}} = r$ there needs to be high discontent.

To this end we introduce several gadget submarkets. In particular for each variable x, we introduce the Unequal-Prices $(G, \overline{G}, F, r)$ submarket to achieve

[4]Doing this does not appear to be compatible with both demand-driven allocations and no utility for money, which may explain our inability to obtain the same hardness bound in this latter setting.

the Pricing Pool and for each clause $C = a \vee b \vee c$ we introduce the Force-Clause-True (G_a, G_b, G_c, r) submarket to achieve the Satisfying Pool.

Intuitively, these submarkets function as follows. Unequal-Prices $(G, \overline{G}, F, r)$ has two buyers, B_G and $B_{\overline{G}}$. B_G desires goods G, F, and money, while $B_{\overline{G}}$ desires $\overline{G}, F$, and money. Each buyer has r money. If $p_G \leq r$, then B_G will desire one copy of G plus leftover money. If $p_G > r$ and $p_F \leq r$, then B_G will desire one copy of F plus leftover money. $B_{\overline{G}}$'s desire is analogous. If both $p_G > r$, $p_{\overline{G}} > r$, and $p_F \leq r$, then there is a demand for two copies of F; only one will be available. The construction of the submarket will ensure this causes $\Theta(r)$ buyer discontent. This entails forcing p_F to be much less than r (we choose to force $p_F \leq 3r/4$); the "forcing" arises by causing $\Theta(r)$ discontent if $p_F > 3r/4$. In turn, the forcing uses yet another good D. Thus the submarket either creates $\Theta(r)$ discontent or yields at least one of $p_G, p_{\overline{G}} \leq r$. As it turns out, if both $p_G, p_{\overline{G}} \leq r$, the solution is only improved (i.e. discontent reduced) by setting either one of them to $r + 1$.

For each variable x, we use multiple instances of the Unequal-Prices $(G, \overline{G}, F, r)$ submarket.

Notice that if $p_G > r$, for each instance of the Unequal-Prices $(G, \overline{G}, F, r)$ submarket, there is one "leftover" copy of the corresponding good G. These will be the goods used to meet the demand of the buyers in Force-Clause-True (G_a, G_b, G_c, r) submarket. This demand will be sufficiently met so long as at least one of $p_a = r + 1$, or $p_b = r + 1$, or $p_c = r + 1$; if all of $p_a, p_b, p_c \leq r$, then too little of the demand will be met, causing $\Theta(r)$ discontent, while in

the other scenarios there will be only $O(1)$ discontent.

The net result, as we will see, is that if there is a satisfying assignment, the corresponding pricing causes $O(1)$ discontent in each instance of the above submarkets.

As we have seen, it is convenient to describe buyer demands in terms of prices, which also seems natural given the demand-driven perspective. However, we also need to ensure that the demands are reasonable in the sense that they can be generated by utility functions. For this to happen, we need the demands to observe the weak axiom of revealed preferences (WARP) ([66], page 29). The axiom states that if under a given set of prices, an agent prefers allocation A_1 among two feasible allocations A_1 and A_2, then at any other set of prices where both A_1 and A_2 are feasible, the agent will prefer A_1 over A_2. We achieve this by specifying demands at a few critical prices and leaving them otherwise unchanged as prices head away from these critical values. At first sight, this approach seems to make achieving the relaxed WGS property even harder, for these buyers do not obey the relaxed WGS property individually or in combination since their demand for money grows as prices drop from their critical values. But in fact, this approach simplifies the market-wide achievement of relaxed WGS, as well as bounded elasticity, which are met by introducing two more buyers per good, one to achieve relaxed WGS, and one to achieve bounded elasticity E (without interfering with each other or the prior demand changes between critical values).

In Section 4.3.1, we present the balanced Max-3SAT-3 problem, and prove that it is NP-hard. Subsequently, in Section 4.3.2, we provide the remaining details of the hardness construction and complete the proof of Theorem 4.1.

4.3.1 The Balanced Max-3SAT-3 problem

To prove our result, we start from the following Max-3SAT-B problem.

Input. A CNF formula with n variables in which each clause contains either 2 or 3 literals, and every variable appears in at most a bounded number (B) of clauses.

Output. The maximum number of clauses can be satisfied simultaneously by some assignment to the variables.

Arora et al. [3] prove the following:

Theorem 4.2 ([3]) *For some fixed $\epsilon > 0$, it is NP-hard to distinguish between satisfiable 3CNF-B formulas, and 3CNF-B formulas in which at most a $(1-\epsilon)$-fraction of the clauses can be satisfied simultaneously.*

Theorem 4.3 The balanced Max-3SAT-3 problem is APX-Hard: *For some fixed $\delta > 0$, it is NP-hard to distinguish between satisfiable 3CNF-3 formulas, and 3CNF-3 formulas in which at most a $(1-\delta)$-fraction of the clauses can be satisfied simultaneously.*

Proof. Given an instance ψ of a Max-3SAT-B problem, we reduce it to an instance ϕ of the balanced Max-3SAT-3 problem in polynomial time.

Consider any variable x and let b be the number of occurrences of x in ψ. Let b_+ and b_- denote the number of positive and negative occurrences of x, respectively. The first step of the reduction is to add $|b_+ - b_-|$ *balance* clauses, where each balance clauses reduces the imbalance between the positive and negative literals by 1. Without loss of generality, suppose that $b_+ > b_-$; we add $b_+ - b_-$ clauses of the form $(\overline{x} \vee \overline{x} \vee x)$. The addition of these clauses causes the boolean formula to be balanced.

Suppose that there are b_x occurrences of x (and therefore of $\overline{x}$) in the new formula. Pair each occurrence of x with an occurrence of $\overline{x}$. Since the formula is balanced, all occurrences are paired. Replace the occurrence of x in the ith pair with a fresh variable x_i, for $0 \leq i \leq b_x - 1$, and add $2b_x$ clauses $(x_i \vee \overline{x}_{i+1}), (\overline{x}_i \vee x_{i+1})$, where $i+1$ is computed mod b_x. The new clauses are satisfied if and only if $x_i = x_{i+1}$ for every i. Now each literal appears exactly 3 times, and no literal appears more than once in the same clause. The above reduction has the following properties:

- The reduction takes polynomial time.
- If ψ is satisfiable, then so is ϕ.
- The number of clauses increases by at most a constant multiplicative factor. Let m, m_1, m_2 be the number of clauses in the formula ψ, after the first step, and in the final formula ϕ respectively. We have $m_2 \leq m_1 + 2\sum_x b_x \leq 2m_1$, and $m_1 \leq 4m = O(m)$. Thus $m_2 = O(m)$ as desired.

- The number of unsatisfiable clauses decreases by at most a constant multiplicative factor. We need to argue that if for each truth assignment to ψ, ϵm clauses in ψ are not satisfied, then for each truth assignment to ϕ, δm_2 clauses in ϕ are not satisfied, for a suitable constant $\delta > 0$. We prove the contrapositive. Assume that under the assignment σ_ϕ fewer than δm_2 clauses in ϕ fail to be satisfied. Then we construct an assignment σ_ψ from assignment σ_ϕ as follows: for a boolean variable x in ψ, if each one of $x_i, 0 \leq i \leq b_x - 1$ has the same truth value in σ_ϕ, then set x to that value. If however, some of the x_i are true while others are false, pick the majority of the two as the value of x. Each unsatisfied clause in ϕ yields at most B unsatisfied clauses in ψ.

The last two properties imply that if a ϵ-fraction of the clauses of ψ are not satisfiable, then an $\frac{\epsilon m}{Bm_2}$-fraction of the clauses of ϕ are not satisfiable; setting $\delta = \frac{\epsilon m}{Bm_2}$ suffices. □

4.3.2 Details of the Hardness Construction

We introduce seven gadget submarkets: the already mentioned Unequal-Prices $(G, \overline{G}, F, r)$ and Force-Clause-True (G_a, G_b, G_c, r), plus Force-Price, Strong-Force-Price, Liberator, Buffer, and Big-Buyer. We will explain their roles and functioning in turn. The constructions use a suitably small constant $\gamma > 0$; $\gamma = 1/2er^2$ suffices, where e is another constant we will introduce later.

We begin by showing how to achieve the discontent needed for the dis-

content pool, without obtaining either bounded elasticity E or relaxed WGS. This uses the first four of the above gadget submarkets.

For variable x, the pricings correspond to a truth assignment if $p_F \leq 3r/4$, $p_D \leq r/2$, either $p_G \leq r$ or $p_{\overline{G}} \leq r$ or both. We will show:

Discontent Properties:

1. If $p_F > 3r/4$ there is at least $kp_F > 3kr/4$ seller discontent (Claim 4.3).
2. If $p_D > r/2$ and $p_F \leq 3r/4$ there is $kp_D > kr/2$ seller discontent (Claim 4.2).
3. If $p_F \leq 3r/4$, $p_D \leq r/2$, in each Strong-Force-Price submarket for which the buyer does not receive a copy of F, the buyer will have discontent at least $r/4$ (Claim 4.1).
4. If $p_F \leq 3r/4$, $p_D \leq r/2$, $p_G \geq r+1$, $p_{\overline{G}} \geq r+1$, there is at least $kr/8$ buyer discontent among the buyers in the Strong-Force-Price and Unequal-Prices $(G, \overline{G}, F, r)$ submarkets (Claims 4.5 and 4.19).

Discontent Properties 1, 2 and 4 yield the Pricing Pool discontent of at least $kr/8$ if at least one of $p_F > 3r/4$, $p_D > r/2$, or both $p_G, p_{\overline{G}} \geq r+1$ holds. Finally, we show that:

5. For a clause $C = a \vee b \vee c$, if $p_a, p_b, p_c \leq r$ and all the corresponding $p_F \leq 3r/4$, $p_D \leq r/2$, then there is at least $kr/8$ discontent among the buyers of G_x for $x \in \{a, b, c\}$. (Claims 4.9 and 4.20).

Discontent property 5 yields the Satisfying Pool discontent of at least $kr/8$.

The core of our construction comprises the following submarkets:

1. For each variable x:

   (a) $6k$ instances of Unequal-Prices $(G, \overline{G}, F, r)$.

   (b) $7k$ instances of Strong-Force-Price $(F, D, 3r/4)$.

   (c) k instances of Force-Price $(D, r/2)$.

2. For each clause $C = a \vee b \vee c$:

   (a) $2k$ instances of Force-Clause-True (G_a, G_b, G_c, r).

This construction uses k copies of D, $13k$ copies of F and $6k$ copies each of G and $\overline{G}$. It is helpful to think of having one copy each of $G, \overline{G}$ and F associated with each instance of Unequal-Prices $(G, \overline{G}, F, r)$, one copy of F with each instance of Strong-Force-Price $(F, D, 3r/4)$, one copy of D with each instance of Force-Price $(D, r/2)$.

We begin by describing Force-Price and Strong-Force-Price so as to obtain Discontent Properties 1-3. Recall that $\gamma > 0$ is a small constant, suitably chosen.

1. Force-Price $(D, r/2)$: This gadget is used to ensure that $p_D \leq r/2$ or there is an excess supply of one copy of D.

   The submarket has one buyer, B_{FP}, that has $r/2$ units of money. B_{FP} desires one copy of D, if affordable, and then leftover money. The following

utility suffices.

$$u(n_D, n_\$) = \begin{cases} r/2 + 1 + \gamma(n_D - 1) + n_\$ & \text{if } n_D \geq 1 \\ n_\$ & \text{if } n_D = 0 \end{cases}$$

2. Strong-Force-Price $(F, D, 3r/4)$: This gadget is used to ensure that $p_F \leq 3r/4$ and that when $p_F \leq 3r/4$, if the expected allocation of goods does not occur (as specified shortly) there is $3r/4 - p_D$ buyer discontent.

   It has one buyer, B_{SFP}, that has $3r/4$ units of money. B_{SFP} desires one copy of F, if affordable, alternatively one copy of D as a second preference, and then leftover money. The following utility suffices.

$$u(n_F, n_D, n_\$) = \begin{cases} 3r/2 + 1 + \gamma(n_F - 1 + n_D) + n_\$ & \text{if } n_F \geq 1 \\ 3r/4 + 1 + \gamma(n_D - 1) + n_\$ & \text{if } n_F = 0 \text{ and } n_D \geq 1 \\ n_\$ & \text{if } n_F = 0 \text{ and } n_D = 0 \end{cases}$$

Claim 4.1 *In the Strong-Force-Price $(F, D, 3r/4)$ submarket, if $p_F \leq 3r/4$, $p_D \leq r/2$, and B_{SFP} does not receive a copy of F, B_{SFP} has discontent at least $3r/4 - p_D \geq r/4$.*

Proof. At $p_F \leq 3r/4$, B_{SFP} desires one copy of F together with left-over money. Because a buyer can receive only goods it demands, if B_{SFP} does not receive a copy of F, it remains with $3r/4$ money. But $u_{\text{SFP}}(0, 1, 0) > u_{\text{SFP}}(0, 0, 3r/4)$, yet the LHS allocation costs p_D, while the RHS one costs $3r/4$. □

Claim 4.2 *In submarkets* Force-Price $(D, r/2)$ *and* Strong-Force-Price $(F, D, 3r/4)$, *if* $p_{_D} > r/2$ *and* $p_{_F} \leq 3r/4$, *there is seller discontent* $p_{_D}$.

Proof. There is no demand for D in this scenario, but there is a supply of one copy. □

To obtain Discontent Property 1 above we need to know that each instance of the Unequal-Prices $(G, \overline{G}, F, r)$ submarket creates a demand for at most 2 copies of F.

Claim 4.3 *If* $p_{_F} > 3r/4$ *there is seller discontent of at least* $kp_{_F}$.

Proof. There is at most $12k$ demand (from the Unequal-Prices $(G, \overline{G}, F, r)$ submarkets), but the supply is $13k$. □

Discontent Property 4 will follow from the definition of the Unequal-Prices $(G, \overline{G}, F, r)$ submarket.

3. Unequal-Prices $(G, \overline{G}, F, r)$: The submarket has two buyers, B_G and $B_{\overline{G}}$. B_G has r money; it desires copies of G, F and money. Its first preference is for one copy of G, failing that for one copy of F, and failing that for money. It has minimal desire for additional copies of F and G. The following utility function suffices.

$$u_G(n_G, n_F, n_\$) = \begin{cases} 2r + 1 + \gamma(n_G - 1 + \ n_F) + n_\$ & \text{if } n_G \geq 1 \\ r + 1 + \gamma(n_F - 1) + n_\$ & \text{if } n_G = 0 \text{ and } n_F \geq 1 \\ n_\$ & \text{if } n_G = 0 \text{ and } n_F = 0. \end{cases}$$

The second buyer desires copies of $\overline{G}$ rather than G and has an analogous utility function.

Note that as claimed each instance of the Unequal-Prices $(G, \overline{G}, F, r)$ submarket has a demand for at most two copies of F (when $p_G, p_{\overline{G}} \geq r+1$).

Claim 4.4 *If $p_G \geq r+1$, $p_{\overline{G}} \geq r+1$, and $p_F \leq 3r/4$, there is an excess demand for one copy of F in the Unequal-Prices $(G, \overline{G}, F, r)$ submarket.*

Proof. At these prices, neither B_G nor $B_{\overline{G}}$ can afford copies of G or $\overline{G}$. So they each desire one copy of F creating an overall demand for two copies of F; but only one is available in this submarket. □

Claim 4.5 *If $p_G \geq r+1$, $p_{\overline{G}} \geq r+1$, $p_F \leq 3r/4$, $p_D \leq r/2$, there is at least $3kr/2$ buyer discontent among the buyers $B_{\mathrm{SFP}}, B_G, B_{\overline{G}}$.*

Proof. By Claim 4.4 and (implicitly) by Claim 4.1, there is a demand for $19k$ copies of F, i.e. an excess demand of $6k$. By Claim 4.1, each buyer B_{SFP} missing a copy has discontent at least $r/4$. Similarly, if a buyer B_G (or $B_{\overline{G}}$) is missing a copy it has discontent at least $r/4$; for $u_G(0,1,0) > u_G(0,0,r)$ and yet the LHS allocation costs at least $r/4$ units of money less than the RHS one. So each unit of excess demand creates at least $r/4$ buyer discontent. □

It remains to define the Force-Clause-True (G_a, G_b, G_c, r) submarkets and to show Discontent Property 5.

4. Force-Clause-True (G_a, G_b, G_c, r): This gadget has one buyer B_{FCT_C} with wealth $3r+3$. B_{FCT_C} desires one copy of each of G_a, G_b, G_c, that has price at most $r+1$. If all of $p_a, p_b, p_c \geq r+2$, B_{FCT_C} desires one copy of the least expensive if it costs at most $3r+3$, breaking ties in the order G_a, G_b, G_c. The following utility suffices:

$$u(n_a, n_b, n_c, n_\$) = \begin{cases} (3r+4) + n_\$ + \gamma(n_a+1) & \text{if } n_a \geq 1, n_b = 0, n_c = 0 \\ (3r+4) + n_\$ + \gamma(n_b) & \text{if } n_a = 0, n_b \geq 1, n_c = 0 \\ (3r+4) + n_\$ + \gamma(n_c-1) & \text{if } n_a = 0, n_b = 0, n_c \geq 1 \\ n_\$ & \text{if } n_a = 0, n_b = 0, n_c = 0 \\ 4r+5 + n_\$ + \gamma(n_a+n_b+2) & \text{if } n_a \geq 1, n_b \geq 1, n_c = 0 \\ 4r+5 + n_\$ + \gamma(n_a+n_c+1) & \text{if } n_a \geq 1, n_b = 0, n_c \geq 1 \\ 4r+5 + n_\$ + \gamma(n_b+n_c) & \text{if } n_a = 0, n_b \geq 1, n_c \geq 1 \\ 5r+6 + n_\$ + \gamma(n_a+n_b+n_c+2) & \text{if } n_a \geq 1, n_b \geq 1, n_c \geq 1. \end{cases}$$

Claim 4.6 *If $p_a, p_b, p_c \leq r+1$, and B_{FCT_C} receives only money then it has discontent at least $2r$.*

Proof. $u(1,0,0,0) > u(0,0,0,0,3r+3)$, yet the LHS allocation costs $3r+3-p_a \geq 2r+2$ less than the RHS one. □

Claim 4.7 *If at least one of p_a, p_b, or p_c equals $r+1$ and B_{FCT_C} receives at least one of the goods priced at or below $r+1$, then B_{FCT_C} has upper discontent at most 3.*

Proof. The worst example arises when B_{FCT_C} receives one copy of G_c and $p_a = p_b = r$, $p_c = r+1$. Then as $u(1,1,0,r) < u(0,0,1,2r+2)$ and as the LHS is an optimal allocation, the lower value is at least $3r$ and hence the upper discontent is at most $3r + 3 - 3r = 3$. □

Claim 4.8 *If $p_F \leq 3r/4$ and $p_G \leq r$, and if the buyer B_G (in Unequal-Prices) does not receive a copy of G it has discontent at least $r/4$. An analogous result holds for $B_{\overline{G}}$ and $\overline{G}$.*

Proof. By the allocation rule, at the given set of prices, B_G demands only G, and on not receiving a copy of G, it remains with r money. But $u_G(0,1,0) > u_G(0,0,r)$, yet the LHS allocation is at least $r/4$ cheaper than the RHS one so its discontent is at least $r/4$. □

Claim 4.9 *Let $C = a \vee b \vee c$ be a clause. Suppose that $p_x \leq r, p_{F_x} \leq 3r/4, p_{D_x} \leq r/2$ for $x \in \{a,b,c\}$. Then B_{FCT_C} and B_{G_x} (from the Unequal-Prices submarket) have discontent at least $kr/2$.*

Proof. Consider the $6k$ buyers B_{G_x}. Between them they desire $6k$ copies of G_x. Suppose they receive $6k - l_x$ copies. By Claim 4.8, this creates at least $rl_x/4$ discontent. This results in the buyers in the $2k$ Force-Clause-True (G_a, G_b, G_c, r) submarkets receiving at most $l_a + l_b + l_c$ copies of G_a, G_b and G_c. By Claim 4.6, the $2k$ buyers B_{FCT_C} have discontent at least $2r(2k - l_a - l_b - l_c)$; the total discontent is at least $2kr/4 \geq kr/2$. □

It is readily seen that the submarkets we have formed obey WGS on every good except money, for each buyer wants (at the available amount of money)

at most one copy of each good, and demand for a good may increase only when its price drops or other prices rise.

The challenge is to achieve relaxed WGS on money, for as a price drops, typically the retained money increases. To offset this, we introduce the Buffer submarket.

5. Buffer (H, r', r'', f): This submarket has one buyer B_{BF} with $f(r'-1)+\delta$ money, where $0 \le \delta < r''$ and $f(r'-r'')+\delta = (r''-1) \bmod r''$. The assumption we make in using Buffer is that when p_H decrements from $p+1 \le r'$ to p, the money retained by buyers buying H in the other submarkets increases by at most f. Then at price p, B_{BF} seeks to spend up to $f(r'-p)+\delta$ on copies of H. Loosely speaking, this is f more spending than at price $p+1$. Finally, we will be seeking to have $p_H = r''$, so we create another $\left\lfloor \frac{f(r'-r'')+\delta}{r''} \right\rfloor$ copies of H.

   Let p_i be the largest price at which B_{BF} desires at least i copies of H (note that p_i is a non-increasing function of i). Then the following utility suffices:

$$u_{\mathrm{BF}}(n_H, n_\$) = \sum_{i=1}^{n_H} (p_i + 1/2) + n_\$.$$

Claim 4.10 *For $i = x(r'') - h$, $p_i \ge r'' + \frac{hr''}{3f}$ if $r'' \ge r'/2$, where $x(p)$ denotes the demand for H in Buffer (H, r', r'', f) when $p_H = p$.*

Proof. When $p_H = r''$, the spending on H is $f(r'-r'')+\delta-(r''-1)$. At $p_H = r''+\Delta$, the cost increases by at most $\Delta x(r'') \le \Delta f(r'-r'')/r''$, and the spending reduces by at most $(f+1)\Delta$.

If $(f+1)\Delta + \Delta f(r'-r'')/r'' < hr''$, the demand has reduced by less than h, that is, if $\Delta < \frac{hr''}{fr'/r''+1}$, the demand drops by less than h. Since $r'/r'' \leq 2$, it suffices that $\Delta < \frac{hr''}{2f+1}$. For $f \geq 1$, $\Delta < \frac{hr''}{3f}$ suffices.

So to reduce demand by h, implies the price increases by at least $\frac{hr''}{3f}$. □

Claim 4.11 *If $p_H \leq r''$ and B_{BF} has $x(r'') - h$ copies of H, its discontent is at least $\frac{(h-1)hr''}{6f}$.*

Proof. The following is a higher utility allocation: h more copies of H and with spending including retained money reduced by

$$\begin{aligned}\left\lfloor \sum_{i=x(r'')-h+1}^{x(r'')-1} (p_i - r'') \right\rfloor &\geq \sum_{j=1}^{h-1} \frac{jr''}{3f} \qquad \text{(by Claim 4.10)} \\ &= \frac{(h-1)hr''}{6f}.\end{aligned}$$

□

Claim 4.12 *The Buffer submarket together with the rest of the market obey relaxed WGS for money and for good H with respect to changes in p_H, for $p_H \leq r'$.*

Proof. The claim for H is straightforward as the demand of B_{BF} only increases as p_H decreases, and we already observed the rest of the market obeys WGS for H.

To see the result for money, we compare the demand for money at prices p and $p + h \leq r'$. At the lower price, the rest of the market demands at most

hf more money, by assumption. The Buffer buyer at price p seeks $f(r'-1) - f(r'-p) + [f(r'-p) + \delta \bmod p]$ money, and at price $p+h$, $f(r'-1) - f(r'-p-h) + [f(r'-p-h) + \delta \bmod p+h]$. Thus, (demand for money at $p_H = p+h$) minus (demand for money at $p_H = p$) is at least $fh - (p-1) - fh \geq -(p-1)$. This is exactly the definition of relaxed WGS. □

To achieve relaxed WGS on goods D, F and G, we introduce one instance each of Buffer $(D, 3r/4, r/2, 14k)$, Buffer $(F, r, 3r/4, 19k)$ and Buffer $(G, r+1, r, 12k)$. We also need to confirm that Discontent Properties 1-5 above continue to hold; we will return to this issue when we have assured that the market has bounded elasticity E.

To enforce bounded elasticity E, we use the Big-Buyer submarket.

6. Big-Buyer (H, r', r''): In this gadget there is one buyer B_{BB} with money $er'r'' + (r'-1)$. This buyer has a high desire to spend all but $2r'-2$ of its money on H. The remaining money is balanced between a slight preference for H and a preference for being unspent, the exact balance being chosen to ensure relaxed WGS for money with respect to changes in p_H.

   Let $\text{rem}(p)$ be the available money unspent by B_{BB} at price p:

   $$\text{rem}(p) = f(r'-p) + \delta \bmod p.$$

   Then B_{BB} has demand $\left\lfloor \frac{er'r'' + \text{rem}(p)}{p} \right\rfloor$ for H when $p \leq r'$, and $\left\lfloor \frac{er'r''}{p} \right\rfloor$ when $p > r'$.

Let q_i be the largest price at which B_{BB} desires i copies of H. To ensure q_i only decreases as i increases, it suffices that

$$\left\lfloor \frac{er'r''}{p} \right\rfloor + 1 \leq \left\lfloor \frac{er'r''}{p-1} \right\rfloor \quad \text{for } p \leq r'$$

or that

$$\frac{er'r''}{p} + 2 \leq \frac{er'r''}{p-1}$$

or that

$$2p(p-1) \leq er'r''$$

We will always choose $r'' \geq r'/2$, and then the condition is true if $e \geq 4$.

We use the following utility function for B_{BB}:

$$u_{\text{BB}}(n_H, n_\$) = \begin{cases} \sum_{i=1}^{n_H}(q_i + \frac{1}{2}) + n_\$ & \text{if } n_\$ \leq 2r' - 2 \\ \sum_{i=1}^{n_H}(q_i + \frac{1}{2}) + 2r' - 2 + \frac{n_\$ - 2r' + 2}{2} & \text{if } n_\$ > 2r' - 2 \end{cases}$$

Claim 4.13 *Let* $y(p) = \left\lfloor \frac{er'r'' - (r'-1)}{p} \right\rfloor$. *If* B_{BB} *receives* $y(p) - h$ *copies of* H *when* $p_{_H} = p \leq r'$, *it has discontent at least* $hp_{_H}/2$, *for* $h \geq 1$.

Proof. For the Big-Buyer, we have:

$$\begin{aligned} & u_{\text{BB}}\Big\{y(p) - \lfloor h/2 \rfloor, er'r'' + r' - 1 - p\big[y(p) - \lfloor h/2 \rfloor\big] - (p/2)h\Big\} \\ & > u_{\text{BB}}\Big\{y(p) - h, er'r'' + r' - 1 - p\big[y(p) - h\big]\Big\}, \end{aligned} \tag{4.2}$$

for the LHS utility minus the RHS utility is at least:

$$p\left(h - \lfloor h/2 \rfloor\right) + \frac{1}{2}\Big[er'r'' + r' - 1 - p\left(y(p) - \lfloor h/2 \rfloor\right) - p\lfloor h/2 \rfloor\Big]$$
$$-\frac{1}{2}\Big[er'r'' + r' - 1 - p(y(p) - h)\Big] - \text{term}(h)$$

where

$$\text{term}(h) \;=\; \begin{cases} 0 & \text{when } h \text{ is even.} \\ p/2 & \text{when } h \text{ is odd} \end{cases}$$

This can be seen as follows. First, the LHS utility corresponds to $h - \lfloor h/2 \rfloor$ additional items, each of which yields a utility of at least p. This accounts for the first term. Second, for all allocated money, except the first $2r' - 2$ units, there is a utility of $1/2$ per unit. When h is even both allocations that are being compared in (4.2) have at least $2r' - 2$ money, while when h is odd, the RHS allocation has at least $2r' - 2$ money, but the LHS one could have as little as $2r' - 2 - p/2$ money.

Thus, the difference in utilities is at least:

$$\begin{aligned} & \frac{p}{2}\lceil h/2 \rceil - \begin{cases} ph/4 & \text{when } h \text{ is even} \\ p(h-1)/4 + \frac{p}{2} & \text{when } h \text{ is odd} \end{cases} \\ = \;& 0. \end{aligned}$$

But the cost of the RHS allocation minus the cost of the LHS one is at least $ph/2$. □

Comment 4.3 *Because B_{BB}'s demand for H beyond the first $y(p)$ copies may cause less discontent, we refer to this discontent as* low discontent*, and it is equal to:*

$$actual\ demand - y(p) = \left\lceil \frac{r' - 1 + rem(p)}{p} \right\rceil .$$

Claim 4.14 *The Big-Buyer, the Buffer and the rest of the market observe relaxed WGS.*

Proof. Again, B_{BB}'s demand for H only increases as p_H decreases, so all that needs to be checked is the demand for money. As in the proof of Claim 4.12, we can compute (demand for money at $p_H = p + h$) minus (demand for money at $p_H = p) \geq (p+h-1)f - fh - (p-1)f - [(er'r'' + (r'-p)f + \delta) \bmod p] \geq -(p-1)$.

For $p_H > r'$, the only source of demand is B_{BB}, and by design, its demand obeys relaxed WGS. □

Claim 4.15 *Suppose that when p_H decrements from $p+1$ to p, the demand for H in the market apart from Buffer and Big-Buyer increases by at most d for $p \leq r'$, and by zero for $p > r'$. Then the market has bounded elasticity E for $E > 1$ if $e \geq \frac{2(d+7)}{E-1}$, $r'' \geq r'/2$, $r' \geq f$, where f is as in the Buffer (H, r', r'', f) submarket.*

Proof. We begin by analyzing the case $p < r'$. We need to show that:

$$x_H(p) \leq \left\lceil x_H(p+1)(1+1/p)^E \right\rceil$$

It suffices to show:

$$d + \frac{er'r'' + f(r' - p) + \delta}{p} + 1 \leq \left(\frac{er'r'' + f(r' - p - 1) + \delta}{p + 1} - 2 \right) (1 + E/p) .$$

Multiplying the entire equation by $p(p + 1)$ and expanding LHS and RHS separately, we obtain:

$$\begin{aligned} \text{LHS} &= (d + 1)p(p + 1) + er'r''(p + 1) + \big[f(r' - p) + \delta\big](p + 1) \\ \text{RHS} &= er'r''(p + E) - 2p(p + 1) - 2E(p + 1) + \big[f(r' - p - 1) + \delta\big](p + E) \end{aligned}$$

Simplifying the inequality leads to:

$$(d + 3)p(p + 1) + 2E(p + 1) + f(p + E) \leq (E - 1)er'r'' + (E - 1)\big[f(r' - p) + \delta\big]$$

So it suffices that:

$$(d + 3)p(p + 1) + 2E(p + 1) + f(p + E) \leq (E - 1)er'r''$$

or that:

$$(d + 3)(r')^2 + 2(r')^2 + 2fr' \leq \frac{(E - 1)e(r')^2}{2}$$

or:

$$(d + 5 + 2f/r') \leq \frac{(E - 1)e}{2}$$

And so $e \geq \frac{2(d+7)}{E-1}$ suffices.

For $p \geq r'$, it suffices to show:

$$\left\lfloor \frac{er'r''}{p} \right\rfloor \leq \left\lceil \left\lfloor \frac{er'r''}{p+1} \right\rfloor (1 + E/p) \right\rceil$$

Let $er'r'' = x(p+1) + y$ with $0 \leq y \leq p$. Then it suffices to show:

$$x + \lfloor x/p + y/p \rfloor \leq \lceil x(1 + E/p) \rceil = x + \lceil xE/p \rceil$$

Let $x = up + z$, $0 < z < p$. Then it suffices to show:

$$u + \lfloor (z+y)/p \rfloor \leq u + \lceil (E-1)u/p + zE/p \rceil$$

Note that $\lfloor (z+y)/p \rfloor \leq 1$, so unless $u = z = 0$ the equation is true. But then $p + 1 \geq er'r''$ and this is outside the range we need consider. □

For each one of the goods $F, D, G, \overline{G}$, we have $f = O(E)$, and $r' = \Theta(r)$ so the condition $r' \geq f$ is implied by the condition $r = \Omega(E)$, which we make.

For D, on a price decrement by one, the demand increases by at most $7k$; this happens when p_D drops from $3r/4 + 1$ to $3r/4$. So we use one instance of Big-Buyer $(D, 3r/4, r/2)$ and need $e \geq 2(7k+7)/(E-1)$.

For F, again, the demand increases by at most $12k$; this happens when p_F drops from $r + 1$ to r. So we use one instance of Big-Buyer $(F, r, 3r/4)$ and need $e \geq 2(12k+7)/(E-1)$.

For G, the demand increases by at most $12k$; this can happen when p_G drops from $r + 1$ to r. So we use one instance of Big-Buyer $(G, r+1, r)$ and

need $e \geq 2(13k+7)/(E-1)$.

There is one more submarket that we need. It is needed to cope with the fact that p_G could legitimately take on either the value r or $r+1$, and if the pricing corresponds to a satisfying assignment, then the discontent must be low. The difficulty is that if B_{BB} does not receive essentially its full demand, there is high discontent (Claim 4.13). The increase in demand at price r is e. We introduce e new buyers that want a copy of G at price $r+1$ (and hence at price r) but will have $O(1)$ discontent on not receiving the copy at price r, using the Liberator submarket.

7. Liberator (G, r'): The purpose of this gadget is to absorb one copy of G at prices $p_G \leq r'$, as needed. When the copy is not absorbed it results in $O(r' - p_G + 1)$ buyer discontent.

   The submarket has one buyer B_{L} with wealth r' that desires one copy of G plus residual money. The following utility suffices.

$$u_L(n_G, n_\$) = \begin{cases} r' + 1 + n_\$ + \gamma(n_G - 1) & \text{if } n_G \geq 1 \\ n_\$ & \text{if } n_G = 0 \end{cases}$$

Claim 4.16 *If $p_G \leq r'$, B_{L} has upper discontent at most $r' - p_G + 1$ if it does not receive a copy of G.*

Proof. As $u(0, p_G - 1) < u(0, r')$ and $n_G = 0, n_\$ = p_G - 1$ is an optimal allocation, allocation $n_G = 0$, $n_\$ = r'$ has lower value at least $p_G - 1$, and consequently upper discontent at most $r' - p_G + 1$. □

We introduce one instance each of Liberator (G, r') for $r+1 \le r' \le r+e$.

Claim 4.17 *If $p_G = r$ and none of the buyers B_{L} receive a copy of G the resulting upper discontent is at most $e(e+3)/2 = O(1)$.*

Proof. This follows immediately from Claim 4.16. □

There is a need to reprove Claim 4.15 to account for the effect of the Liberator submarket.

Claim 4.18 *Suppose that when p_G decrements from $p+1$ to p the demand for G in the market apart from Buffer and Big-Buyer increases by at most d for $p < r+1$, and by at most 1 for $r+1 \le p \le r+e$, and by 0 for $p > r+e$. Then the market has bounded elasticity E for $E > 1$, if $e \ge \max\{2(d+7)/(E-1), 10/(E-1)\}$, $r \ge e+1$, $r \ge E$, $r+1 \ge 12k$.*

Proof. The proof of Claim 4.15 has already shown the result except for p in the range $[r+1, r+e]$. It suffices to show:

$$\frac{e(r+1)r}{p} + 1 \le \left(\frac{er(r+1)}{p+1} - 1\right)(1 + k/p)$$

or:

$$2p(p+1) + k(p+1) + er(r+1) \le ker(r+1)$$

or:

$$2(r+e)(r+e+1) + k(r+e+1) \le (k-1)er(r+1)$$

Assuming $r \geq e+1$ and $r \geq k$, it suffices that:

$$10r^2 \leq (k-1)er(r+1)$$

So $e \geq 10/(k-1)$ suffices. □

It remains to check that Discontent-Properties 1-5 still hold. To see (1) and (2) it suffices to note that the excess supply of F and D only increase with the presence of the Buffer and Big-Buyer submarkets when $p_F > 3r/4$ and $p_D > r/2$, respectively. (3) is unaffected by the Buffer and Big-Buyer submarkets. (4) results from an excess demand for F. We need to slightly modify Claim 4.5 to obtain an analogous result.

Claim 4.19 *If $p_G \geq r+1$, $p_{\overline{G}} \geq r+1$, $p_F \leq 3r/4$, $p_D \leq r/2$ there is at least $kr/8$ buyer discontent among the following buyers of F: $B_{\mathrm{SFP}}, B_G, B_{\overline{G}}, B_{\mathrm{BF}}, B_{\mathrm{BB}}$, if $r \geq 8$.*

Proof. There is an excess demand for at least $6k$ copies of F as argued in the proof of Claim 4.5. As argued there, each of the B_{SFP}, $B_G, B_{\overline{G}}$ has discontent at least $r/4$ on not receiving a copy of F. By Claim 4.11, B_{BF} has discontent at least $\frac{(h-1)hr''}{6f} = \frac{(h-1)hr}{152k}$ if it has $x(r'') - h$ copies of F (for $r'' = 3r/4$ and $f = 19k$). By Claim 4.13, B_{BB} has discontent at least $(h' - \lceil \frac{r+p_F}{p_F} \rceil)p_F/2$ if it is missing h' copies of F (as $y(p_F)$ is at most $\left\lceil \frac{r+p_F-2}{p_F} \right\rceil$ smaller than the actual demand of B_{BB}). For $p_F = r''$, the total discontent is minimized when B_{BB} is missing 3 copies and B_{BF} is missing $6k-3$; the total discontent is then at

least $(6k-4)(6k-3)r/134k \geq kr/8$.

If $p_F < r''$, the excess demand due to B_{BB} increases by at least $\frac{err''}{r''-(r''-p_F)} - er \geq (r''-p_F)er/p_F$. There is an increase in B_{BB}'s low discontent demand of $\lceil (r-2)/p_F \rceil - \lfloor (r-2)/r'' \rfloor \leq r/p_F$. For each additional missing item beyond $6k-3$, B_{BF} has discontent at least $2(6k-3)r/134k \geq r/15$. This yields an increase in discontent of at least $([e(r''-p_F)-1]r/p_F - l)p_F/2 + lr/15$, where l items of the increased demand are taken from B_{BF} and the remainder from B_{BB}. But this is always positive and hence the discontent is only higher in this case. □

Again, to show Discontent-Property 5, we need to modify Claim 4.9.

Claim 4.20 *Let $C = a \vee b \vee c$ be a clause. Suppose that $e \geq 3$, $p_x \leq r, p_{F_x} \leq 3r/4, p_{D_x} \leq r/2$ for $x \in \{a,b,c\}$. Then there is at least $kr/8$ discontent among the buyers $B_{G_x}, B_{\mathrm{FCT}_C}, B_{\mathrm{BF}_x}, B_{\mathrm{BB}_x}$ for $x \in \{a,b,c\}$.*

Proof. We begin by considering the case where $p_x = r$ for $x \in \{a,b,c\}$. One copy of these goods is needed for each of the $2k$ B_{FCT_C}. Note that at these prices the B_{BF_x} have zero demand. Likewise, each B_{BB_x} has demand $e(r+1)$ and $y(r) = e(r+1) - 1$ (see Claim 4.13). Recall that B_{BB}'s low discontent demand is defined as actual demand $- y(p)$. Thus each B_{BB_x} can miss one copy of its demand with low discontent. Subsequent missing copies each cause discontent $r/2$ by Claim 4.13. If B_{G_x} or B_{FCT_C} are missing copies they each incur discontent $r/4$ by Claims 4.6 and 4.8 respectively. Thus, overall, there is at least $(2k-3)r/4 \geq kr/8$ discontent.

For smaller $p_x = r - \Delta$, the demand of B_{BB_x} increases by at least $\frac{\Delta e(r+1)}{r-\Delta}$; the low discontent demand (see Claim 4.13) increases by at most $\left\lceil \frac{r+(r-\Delta)}{r-\Delta} \right\rceil \leq \frac{3r}{r-\Delta}$. Thus the increase in discontent is at least $\left\lceil \frac{(\Delta e-3)r}{r-\Delta} \right\rceil \cdot \min\{\frac{r-\Delta}{2}, \frac{r}{4}\}$ (using Claim 4.13, 4.6, and 4.8, according to whether additional unmet demand is allocated to B_{BB_x}, B_{G_x}, B_{FCT_C}, respectively). But this increase in discontent is at least $\frac{\Delta(e-3)r}{4} \geq 0$ if $e \geq 3$. Thus the discontent is only larger in this case. □

Lemma 4.2 *If there is a satisfying assignment there is a pricing and demand-driven allocation yielding $O(mE)$ discontent.*

Proof. For each true literal x, set $p_{D_x} = r/2$, $p_{F_x} = 3r/4$, $p_{G_x} = r + 1$, $p_{G_{\overline{x}}} = r$. The k buyers B_{FP} are each allocated one copy of D, the $7k$ buyers B_{SFP} one copy of F each, the $6k$ buyers B_x one copy of F each, the $6k$ buyers $B_{\overline{x}}$ one copy of $G_{\overline{x}}$ each; the $6k$ copies of G_x are distributed one to each of the $6k$ clausal buyers B_{FCT_C} associated with literal x. As the formula is satisfiable, each clausal buyer B_{FCT_C} receives between one and three of the goods G_a, G_b, G_c, where $C = a \vee b \vee c$. By Claim 4.7 this yields $O(1)$ discontent per clausal buyer, or $O(km) = O(km)$ discontent overall. Finally, each B_{BF} and B_{BB} receives an amount of goods equal to the supply intended for the corresponding submarkets. All the other buyers have zero discontent as do the sellers for all the goods are sold. □

This completes the proof of Theorem 4.1.

4.4 A Matching Algorithm

In this section, we present a simple algorithm that approximates the efficiency of market with bounded elasticity E to a factor $1 - O(E/r)$. We assume that an oracle for computing the aggregate excess demand is available. This assumption is fairly standard [23, 55, 22, 59]. Let $\mathbf{p} = (p_1, \ldots, p_n)$.

Recall that a market has bounded elasticity E if for every good i, for all prices $\mathbf{p}$, for all $h > 0$ such that $x_i(p_i + h, \mathbf{p}_{-i}) \geq 1$,

$$x_i(p_i, \mathbf{p}_{-i}) \leq \lceil x_i(p_i + h, \mathbf{p}_{-i})\,(1 + h/p_i)^E \rceil,$$

and if $x_i(p_i + 1, \mathbf{p}_{-i}) = 0$, then $x_i(p_i, \mathbf{p}_{-i}) \leq 1$.

Definition 4.8 (High price.) *A price p_i in a collection $\mathbf{p} = (p_1, \ldots, p_n)$ of prices is said to be* high *if there is a surplus of good i at $\mathbf{p}$. The surplus need not be strict.* (Barely high price.) *Price p_i is said to be* barely high *if there is a surplus of good i at $\mathbf{p}$, but a strict deficit of good i at $(p_i - 1, \mathbf{p}_{-i})$.*

Low and barely low prices are defined analogously.

Let w denote the total buyer wealth and s be the minimum number of copies of any one good present in the market; i.e. $w = \sum_i w_i,\ s = \min_j s_j$. Let $x_i(\mathbf{p})$ denote the demand for good i at prices $\mathbf{p}$. We aim to find a collection of prices $\mathbf{p}$ so that all goods are priced *barely high*. The algorithm in Figure 4.2 computes such prices while keeping all prices high at all times. On termination, as all the prices are barely high, the supply of each good is at least the demand,

and thus each buyer can be and is allocated all the goods it seeks. We will show that this allocation achieves an efficiency of $1 - O(E/r)$.

```
1  initialize p with p_i ← ⌈w/s_i⌉.
2  repeat  [for each i in turn]
3          if p_i is high and p_i − 1 is not strictly low
4             decrement it (p_i ← p_i − 1)
5          end if
6  until no decrement is effective
```

Figure 4.2: A simple algorithm that converges to a barely high price vector.

Claim 4.21 *A decrement of a price leaves all other high prices high.*

Proof. This is an immediate consequence of the relaxed WGS property. □

Lemma 4.3 *Algorithm 1 terminates after at most n^2w/s calls to the demand oracle.*

Proof. If a price p_i reaches 1, it is necessarily barely high. Further, by Claim 4.21 high prices remain high until decremented. If no price can be dropped, then this is the desired state, otherwise the price of some good is decremented. Since $p_i \geq 1$ for all goods i, there can be at most $(w/s)n$ decrements. Conceivably, we may have to repeatedly try all goods to find the one whose price reduces, and therefore the total number of queries is at most n^2w/s. □

Lemma 4.4 *Decrementing the price of a good whose current price is strictly high and greater than $2Es_i$ increases its demand by at most* 1.

Proof. Let p_i denote the price of such a good after the decrement. Let Δ denote $x_i(p_i, \mathbf{p}_{-i}) - x_i(p_i + 1, \mathbf{p}_{-i})$. We claim that $\Delta \leq 1$. Note that $(1 + 1/p_i)^E \leq 1 + 1/s_i$ for $p_i \geq 2Es_i$. By bounded elasticity E, we have:

$$x_i(p_i, \mathbf{p}_{-i}) \leq x_i(p_i + 1, \mathbf{p}_{-i}) \left(1 + 1/p_i\right)^E + 1.$$

Thus $x_i(p_i, \mathbf{p}_{-i}) \leq x_i(p_i + 1, \mathbf{p}_{-i})(1 + 1/s_i) + 1$. Therefore,

$$\Delta \leq x_i(p_i + 1, \mathbf{p}_{-i})/s_i + 1$$

But $x_i(p_i + 1, \mathbf{p}_{-i}) < s_i$ as $p_i + 1$ is strictly high. As Δ is an integer, $\Delta \leq 1$. □

Lemma 4.5 *The excess supply of a good whose current price p_i is barely high is at most $Es_i/p_i + 1$.*

Proof. If $p_i \leq E$ then the lemma follows trivially since the excess supply is always less than the total supply s_i and $s_i \leq Es_i/p_i + 1$ for $p_i \leq E$. Suppose that $p_i > E$. By bounded elasticity E, we have

$$\begin{aligned} x_i(p_i, \mathbf{p}_{-i}) &\geq \frac{x_i(p_i - 1, \mathbf{p}_{-i}) - 1}{\left(1 + 1/(p_i - 1)\right)^E} \\ &\geq \frac{x_i(p_i - 1, \mathbf{p}_{-i})}{\left(1 + 1/(p_i - 1)\right)^E} - 1 \\ &= x_i(p_i - 1, \mathbf{p}_{-i}) \left(1 - 1/p_i\right)^E - 1. \end{aligned}$$

Since p_i is barely high, $p_i - 1$ must be low, so $x_i(p_i - 1, \mathbf{p}_{-i}) > s_i$. Therefore $s_i - x_i(p_i, \mathbf{p}_{-i}) < s_i - s_i(1 - 1/p_i)^E + 1$. As $(1 - 1/p_i)^E \geq 1 - E/p_i$ for $p_i > E$,

the excess supply is given by:

$$s_i - x_i(p_i, \mathbf{p}_{-i}) < s_i(1 - 1 + E/p_i) + 1 = Es_i/p_i + 1.$$

□

As each buyer receives its optimal allocation, there is no buyer discontent. The only discontent is the seller discontent. Let $\mathbf{p}$ denote the set of prices computed by the algorithm. We claim that a good i whose price p_i is greater than $2Es_i$ is correctly priced. For p_i is barely high, $p_i - 1$ is low, and by Lemma 4.4, the demand for such a good increases by at most 1 at each price decrement; therefore the demand exactly matches the supply at prices $\mathbf{p}$.
Recall that the inefficiency of an allocation is defined as the ratio of the total (buyer and seller) discontent and the total (buyer and seller) value. For subsequent analysis, it will be helpful to rewrite the expression for Market Efficiency (Definition 4.5) and specify the terms for buyers and sellers separately as:

$$1 - \frac{\sum_{b \in B} d_b((A_b, w_b - A_b \cdot \mathbf{p}), \mathbf{p}) + \sum_{g \in G} c_g p_g}{\sum_{b \in B} v_b((A_b^{opt}, m_b^{opt}), \mathbf{p}) + \sum_{g \in G} s_g p_g}$$

where c_g denotes the number of unsold copies of g in the allocation, v_b denotes the upper value for buyer b and (A_b^{opt}, m_b^{opt}) is an optimal allocation (including money) for b with wealth w_b at prices $\mathbf{p}$.

Lemma 4.6 *The above allocation at the prices* $\mathbf{p}$ *computed by the algorithm in Figure 4.2 has efficiency* $1 - O(E/r)$.

Proof. For a good i whose price is less than or equal to $2Es_i$, by Lemma 4.5, there are at most $Es_i/p_i + 1$ unsold copies. The total seller discontent is therefore at most

$$\begin{aligned}\sum_{i:p_i \le 2Es_i} (Es_i/p_i + 1)\, p_i &\le \sum_{i:p_i \le 2Es_i} Es_i + \sum_{i:p_i \le 2Es_i} p_i \\ &\le E\sum_i s_i + \sum_i 2Es_i = 3E\sum_i s_i.\end{aligned}$$

Recall that the inefficiency of a market is defined as the ratio of the total (buyer and seller) discontent and the total (buyer and seller) value. The inefficiency of the allocation at prices $\mathbf{p}$ is then given by:

$$\frac{3E\sum_i s_i}{\sum_j w_j + \sum_i s_i p_i} \le \frac{3E\sum_i s_i}{\sum_j w_j} = O(E/r).$$

The efficiency is thus $1 - O(E/r)$. □

Theorem 4.4 *The algorithm in Figure 4.2 can be modified so that it makes at most $O(nEt \log w)$ demand oracle queries, where $t = \sum_i s_i$.*

Proof. Instead of considering all prices in the set $\{1, \ldots, \lceil w/s_i \rceil\}$, one can consider multiplicatively growing values of p_i from the set $\mathbb{I}_+^i = \{1, 2, \ldots, \lceil 1/\epsilon_i \rceil\} \cup \{\lfloor \frac{1}{\epsilon_i}(1+\epsilon_i)^j \rfloor : j > 0, \frac{1}{\epsilon_i}(1+\epsilon_i)^j \le \lceil w/s_i \rceil\}$. With foresight, we set $\epsilon_i = 1/(8Es_i)$. Note that if at price $\lfloor \frac{1}{\epsilon_i}(1+\epsilon_i)^{j+1} \rfloor$ there is an excess supply of good i, then at price $\lfloor \frac{1}{\epsilon_i}(1+\epsilon_i)^j \rfloor > 8Es_i$, the demand increases by at most 1 due to the following argument. Let $p_i = \lfloor \frac{1}{\epsilon_i}(1+\epsilon_i)^j \rfloor$ and $p_i + h = \lfloor \frac{1}{\epsilon_i}(1+\epsilon_i)^{j+1} \rfloor$

so that

$$\begin{aligned} h &= \left\lfloor \frac{1}{\epsilon_i}(1+\epsilon_i)^{j+1} \right\rfloor - \left\lfloor \frac{1}{\epsilon_i}(1+\epsilon_i)^{j} \right\rfloor \\ &\leq \frac{1}{\epsilon_i}(1+\epsilon_i)^{j+1} - \frac{1}{\epsilon_i}(1+\epsilon_i)^{j} + 1 \\ &= (1+\epsilon_i)^{j} + 1 \leq 2(1+\epsilon_i)^{j}. \end{aligned}$$

Finally,

$$p_i = \left\lfloor \frac{1}{\epsilon_i}(1+\epsilon_i)^{j} \right\rfloor \geq \frac{(1+\epsilon_i)^{j}}{2\epsilon_i}.$$

Thus, $h/p_i \leq 4\epsilon_i$ and $(1+h/p_i)^E \leq (1+4\epsilon_i)^E \leq 1+1/s_i$ for $4\epsilon_i \leq 1/(2Es_i)$. From this point on, an argument identical to the one used in the proof of Lemma 4.4 works. It follows that for a good i whose price is greater than $8Es_i$, upon the termination of the algorithm, the demand equals the supply and there are no unsold copies.

Since $|\mathbb{I}_+^i| \leq 8Es_i(1+\log w)$, the complexity, in oracle calls, becomes $8nE(\sum_i s_i)(1+\log w) = O(nEt\log w)$. □

The modification is to adjust prices multiplicatively: in line 4 above, $p_i \leftarrow \min\{\lceil p_i(1-\epsilon_i)\rceil, p_i - 1\}$, where $\epsilon_i = 1/(8Es_i)$.

When t is too large one can reduce the accuracy of the low test without significantly affecting the quality of the approximation.

Definition 4.9 *Price p_i is* near-low *if the excess demand for good i at $(p_i, \mathbf{p}_{-i})$ is at most s_i/r.*

The test in the algorithm (in line 3) is changed from a test for a low price to

a test for a near-low price. Further, for all i, ϵ_i is set to $1/(8Er)$. This may create a further $O(s_i/r)$ excess supply of good i, but it is clear that this does not affect the approximation quality beyond an additive $O(1/r)$ term. The complexity becomes $O(Ern^2 \log w)$ oracle calls.

4.5 A Local Tatonnement Algorithm

4.5.1 Introduction and Motivation

Section 4.4 presented a simple P-time algorithm to compute near-equilibrium prices of a discrete market obeying WGS with bounded elasticity E. This establishes that for a fixed $E > 1$, the hardness result shown in Section 4.3 (Theorem 4.1) is tight. In this section, we seek a local, distributed, tatonnement based algorithm that converges rapidly toward near-equilibrium prices. The spirit of this investigation is to elucidate the processes by which near-equilibrium prices are discovered *within the market itself*.

In the continuous setting, Cole and Fleischer [25] showed that fast convergence is possible with tatonnement-style price update protocols, methods which update each price independently. Their analysis copes with asynchrony plus data staleness and inaccuracy. In this section, we seek to develop their analysis to account for discreteness in the market so as to provide richer and hence more plausible justifications for why markets can be well-behaved.

As in Sections 4.3 and 4.4, we continue to consider the discrete market with bounded elasticity E and average wealth r. However, in addition to bounded

elasticity (which we saw is equivalent to bounding the fractional rate of change of demand with respect to fractional changes in prices), we need a lower bound on the sensitivity of demand to price changes (made precise later).

In the economics literature, in the continuous setting, a good i is said to be *normal* for agent l if its fractional rate of change of demand with respect to the agent's wealth is non-negative. That is, $\partial x_{il}(\mathbf{p}, v_l)/\partial v_l \geq 0$, where v_l is the wealth of agent l at prices $\mathbf{p}$ and $x_{il}(\mathbf{p}, v_l)$ is agent l's demand for good i at prices $\mathbf{p}$ (see [66] page 25). In their analysis of convergence in the continuous setting, Cole and Fleischer [25] assumed that the fractional rate of change of demand with respect to the wealth is lower bounded by a strictly positive value β. They called this constraint the *wealth effect*. However, in the discrete setting, it turns out that it is no longer effective to place constraints on individual agents in order to limit changes in aggregate demand when price vary. For in the discrete setting, unless individual demands never vary, no constraint can prevent a sharp change in demand on a unit change in price at a critical price – simply imagine that many or all of the individual utilities are identical.

In the continuous setting with real-valued prices, the wealth effect on demand can be expressed in the following manner.

Definition 4.10 (Wealth Effect in the Continuous Setting) *If all prices are reduced by a factor $f < 1$ from $\mathbf{p}$ to $f\mathbf{p}$, then the demand for each good*

increases by at least a factor of $(1/f)^\beta$ *for some* $\beta > 0$. *Equivalently, for all* i

$$x_i(f\mathbf{p}) \geq \frac{x_i(\mathbf{p})}{f^\beta}.$$

Analogously, if all prices are raised by a factor $g > 1$ *from* $\mathbf{p}$ *to* $g\mathbf{p}$, *then the demand for each good decreases by at least a factor of* g^β *for some* $\beta > 0$. *That is, for all* i,

$$x_i(g\mathbf{p}) \leq x_i(\mathbf{p})/g^\beta.$$

Making an analogous definition in the discrete setting is problematic because the price of each good in $f\mathbf{p}$ may not be an integer. In order to avoid this difficulty, we combine the wealth effect with bounded elasticity, and define it as in Definition 4.11.

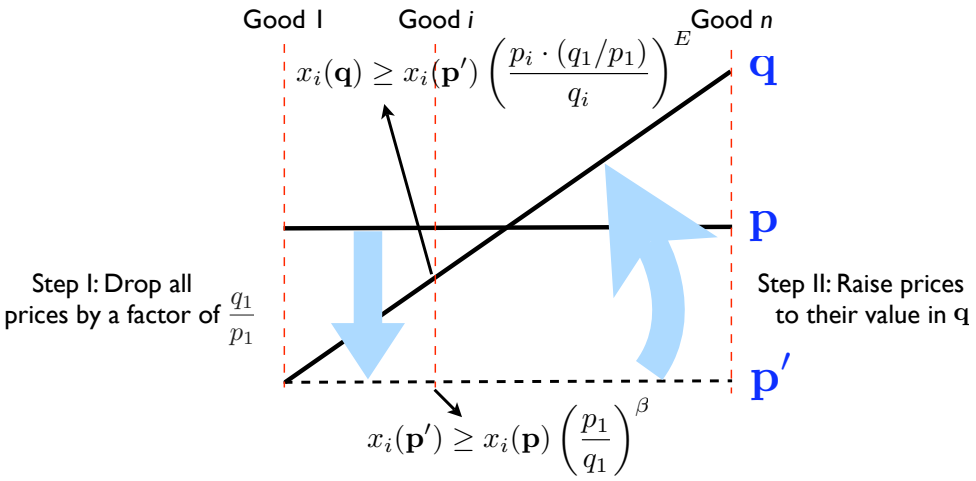


Figure 4.3: An illustration of bounded elasticity E with demand sensitivity β.

Definition 4.11 (Bounded Elasticity E and Demand Sensitivity β) *Let* $\mathbf{p}$ *and* $\mathbf{q}$ *be two price vectors. Assume that* $1 = \arg\min_i \frac{q_i}{p_i}$ *and* $n = \arg\max_i \frac{q_i}{p_i}$. *A market with bounded elasticity* E *is said to have demand sensitivity* β *if when* $\frac{q_1}{p_1} < 1$,

$$x_i(\mathbf{q}) \geq \left\lfloor x_i(\mathbf{p}) \left(\frac{p_1}{q_1}\right)^{\beta} \left(\frac{q_1/p_1}{q_i/p_i}\right)^{E} \right\rfloor$$

and when $\frac{q_n}{p_n} > 1$,

$$x_i(\mathbf{q}) \leq \left\lceil x_i(\mathbf{p}) \left(\frac{p_n}{q_n}\right)^{\beta} \left(\frac{q_n/p_n}{q_i/p_i}\right)^{E} \right\rceil,$$

for some fixed constant $\beta > 0$.

Figure 4.3 provides an intuitive explanation of bounded elasticity E and demand sensitivity β in continuous markets, for two price vectors $\mathbf{p}$ and $\mathbf{q}$ with $\min_i \frac{q_i}{p_i} < 1$. We imagine changing the price vector $\mathbf{p}$ to $\mathbf{q}$ in two steps. In the first step, all prices are dropped by a factor of $\frac{q_1}{p_1}$ (i.e. $\min_i \frac{q_i}{p_i}$) to $\mathbf{p}'$. This increases the demand for good i by at least a $\left(\frac{q_1}{p_1}\right)^{\beta}$ factor. In the second step, the prices are raised from $\mathbf{p}'$ to $\mathbf{q}$. This entails raising the price of good i from $p_i \cdot (q_1/p_1)$ to q_i, and thus by bounded elasticity E, the demand does not drop by more than a factor $\left(\frac{p_i \cdot (q_1/p_1)}{q_i}\right)^{E}$. Combining the two yields bounded elasticity E with demand sensitivity β.

We are interested in price adjustment mechanisms that are based on tatonnement, as in [25], with minor modifications to account for the discrete setting.

Let λ be a parameter of our price update protocol. Looking ahead, λ will

be at most $\frac{1}{cE}$, where E is a bound on the elasticity of demand and c is a constant, suitably chosen.

Price Update Rules: The price of good i is updated according to the following rule:

$$p_i' \leftarrow \left\lceil p_i \left(1 + \lambda \min\left\{\frac{z_i(\mathbf{p})}{s_i}, 1\right\}\right)\right\rceil \tag{4.3}$$

Note that if a good is in excess demand (i.e. $z_i(\mathbf{p}) > 0$), then its price is only increased, while if a good is in excess supply (i.e. $z_i(\mathbf{p}) < 0$), then its price is either decreased or stays the same. The following is an immediate corollary of Bounded Elasticity E and Demand Sensitivity β (Definition 4.11):

Corollary 4.2 (Bounded Elasticity E) *Fix the price of all goods other than good i to $\mathbf{p}_{-i}$. Then, for all $r_i \leq p_i \leq q_i$,*

$$\left\lfloor x_i(r_i, \mathbf{p}_{-i}) \left(\frac{r_i}{p_i}\right)^E \right\rfloor \leq x_i(\mathbf{p}) \leq \left\lceil x_i(q_i, \mathbf{p}_{-i}) \left(\frac{q_i}{p_i}\right)^E \right\rceil.$$

In Section 4.4, we described a notion of *barely high prices* (see Definition 4.8). For the analysis of the local tatonnement-style algorithm, we need the analogous *barely low prices*.

Definition 4.12 (Barely low prices) *A price vector $\mathbf{p}$ is said to be barely low if for each good i, $z_i(\mathbf{p}) \geq 0$ and $z_i(p_i + 1, \mathbf{p}_{-i}) < 0$.*

Comment 4.4 (Complexity Measure) *As is standard for asynchronous algorithms, we measure the complexity in rounds. The basic unit of time is a*

price update as specified above. A round *comprises a minimum length sequence of updates in which every price updates at least once. The rounds are specified uniquely by defining them from a fixed start time.*

4.5.2 Analysis of Convergence

Let $\mathbf{p}^H$ and $\mathbf{p}^L$ denote a pair of barely high and barely low prices, respectively. The two assumptions that we will need in the analysis are:

- the minimum number of copies of a good available in the market is $\Omega(E/\beta)$. Specifically, we require that for all goods i, $s_i \geq \frac{32E}{\beta}$.
- the barely low and the barely high price of each good is at least $\frac{8cE^2}{\beta}$, where c is a constant, suitably chosen.

The outline of our analysis is:

1. First, we show that the barely high and the barely low prices are "close" to each other. Specifically, that for all goods i, $|p_i^H - p_i^L| = O(p_i^H/E)$.
2. Next, we establish that under the price update protocol specified, prices rapidly converge, so that the price of each good i gets to be in the interval $\left[p_i^L(1 - O(1/E)), p_i^H(1 + O(1/E))\right]$.
3. Finally, we deduce that once the prices are bounded, the demand of each good is bounded. Specifically, that for all goods i, $x_i(\mathbf{p}) = O(s_i)$, and therefore the misspending, $\sum_i p_i|z_i(\mathbf{p})|$, is $O(\sum_j w_j)$, where w_j is the amount of money with buyer j in the market.

Lemma 4.7 *Let $\mathbf{p}^H$ and $\mathbf{p}^L$ denote a set of barely high and barely low prices, respectively. Let $n = \arg\max_i p_i^H/p_i^L$. For all goods i,*

$$p_i^H \leq p_i^L \left[1 + \frac{2}{cE}\right],$$

if $p_n^L, p_n^H \geq 8cE^2/\beta$.

Proof. Let $f = p_n^H/p_n^L$. We seek to upper bound f. So suppose that $f > 1$. As p_n^L is barely low, the demand at $(\mathbf{p}_{-n}^L, p_n^L + 1)$ for good n is at most $s_n - 1$. By bounded elasticity,

$$x_n(\mathbf{p}^L) \leq (s_n - 1)\left(\frac{p_n^L + 1}{p_n^L}\right)^E = (s_n - 1)\left(1 + \frac{1}{p_n^L}\right)^E.$$

If $\frac{E}{p_n^L} \leq \frac{1}{2}$, the RHS expression can be bounded as:

$$(s_n - 1)\left(1 + \frac{1}{p_n^L}\right)^E \leq (s_n - 1)\left(1 + \frac{2E}{p_n^L}\right) = s_n + \frac{2Es_n}{p_n^L} - 1 - \frac{2E}{p_n^L}.$$

Similarly, as p_n^H is barely high, the demand at $(\mathbf{p}_{-n}^H, p_n^H - 1)$ for good n is at least $s_n + 1$. By bounded elasticity,

$$x_n(\mathbf{p}^H) \geq (s_n + 1)\left(\frac{p_n^H - 1}{p_n^H}\right)^E = (s_n + 1)\left(1 - \frac{1}{p_n^H}\right)^E.$$

If $\frac{E}{p_n^H} \leq 1$, the RHS expression can be bounded as:

$$(s_n + 1)\left(1 - \frac{1}{p_n^H}\right)^E \geq (s_n + 1)\left(1 - \frac{E}{p_n^H}\right) = s_n - \frac{Es_n}{p_n^H} + 1 - \frac{E}{p_n^H}.$$

Imagine changing all prices from $\mathbf{p}^L$ to $\mathbf{p}^H$; this is an increase of at most a factor f. Hence, by demand sensitivity β (Definition 4.11), the demand for good n at $\mathbf{p}^H$ is at most

$$\left(s_n + \frac{2Es_n}{p_n^L} - 1 - \frac{E}{p_n^L}\right) f^{-\beta}.$$

We need:

$$\left(s_n + \frac{2Es_n}{p_n^L} - 1 - \frac{E}{p_n^L}\right) f^{-\beta} \geq s_n - \frac{Es_n}{p_n^H} + 1 - \frac{E}{p_n^H}.$$

Thus, the required condition on f is:

$$\begin{aligned} f &\leq \left(s_n + \frac{2Es_n}{p_n^L} - 1 - \frac{E}{p_n^L}\right)^{1/\beta} \left(s_n - \frac{Es_n}{p_n^H} + 1 - \frac{E}{p_n^H}\right)^{-1/\beta} \\ &= \left(1 + \frac{2E}{p_n^L} - \frac{1}{s_n} - \frac{E}{s_n p_n^L}\right)^{1/\beta} \left(1 - \frac{E}{p_n^H} + \frac{1}{s_n} - \frac{E}{s_n p_n^H}\right)^{-1/\beta} \\ &\leq \left(1 + \frac{4E}{\beta p_n^L}\right)\left(1 + \frac{4E}{\beta p_n^H}\right) \qquad \left(\text{if } \frac{4E}{\beta p_n^L}, \frac{4E}{\beta p_n^H} \leq 1\right) \\ &\leq 1 + \frac{8E}{\beta}\left(\frac{1}{p_n^L} + \frac{1}{p_n^H}\right). \end{aligned}$$

Since $p_n^H, p_n^L \geq 8cE^2/\beta$ by assumption, it follows that $f \leq 1 + \frac{2}{cE}$. $\square$

Let $s = \min_i s_i$, and let $p_L = \min_i p_i^L, p_H = \min_i p_i^H$. Without loss of generality, let $1 = \arg\max_i \frac{p_i^L}{p_i}$ and $n = \arg\min \frac{p_i^H}{p_i}$. For notational ease, we let z_i denote $z_i(\mathbf{p})$, i.e. the excess demand at current prices.

Lemma 4.8 *After one round of price updates, if $\lambda E(2 + \frac{1}{s}) \leq 1$, the prices $\mathbf{p}'$*

satisfy:

1. *If* $p_1 \geq p_1^L$, *then* $p_i' \geq p_i^L(1 - \lambda/s)$.

2. *If* $p_1 \leq p_1^L$, *then*

$$\frac{p_i'}{p_i^L} \geq \frac{p_1}{p_1^L}\left[1 + \lambda \min\left\{\left(\frac{p_1^L}{p_1}\right)^{\beta} - \frac{1}{s} - 1, 1\right\}\right].$$

Proof. Let $f = \frac{p_1^L}{p_i}$, $r_i = \frac{p_1^L}{p_1}\frac{p_i}{p_i^L}$.

- **Case 1:** $z_i \geq s_i$. In this case, $p_i' \geq p_i(1 + \lambda)$, and the result holds both when the condition of (1) holds and when the condition of (2) holds.

- **Case 2:** $s_i > z_i \geq 0$. If the condition of (1) holds, then the result will hold for p_i. So it suffices to consider (2). The price of good i is updated according to:

$$p_i' = \left\lceil p_i\left(1 + \lambda \min\left\{\frac{z_i}{s_i}, 1\right\}\right)\right\rceil.$$

  By demand sensitivity and bounded elasticity (Definition 4.11), $x_i \geq \lfloor s_i f^{\beta} r_i^{-E} \rfloor$.

  It suffices to show that

$$\frac{p_i}{p_i^L}\left[1 + \lambda \min\left\{f^{\beta} r_i^{-E} - \frac{1}{s_i} - 1, 1\right\}\right] \geq \frac{p_1}{p_1^L}\left[1 + \lambda \min\left\{f^{\beta} - \frac{1}{s} - 1, 1\right\}\right].$$

  Note that since $x_i < 2s_i$, it follows that $s_i f^{\beta} r_i^{-E} < 2s_i$. Thus $f^{\beta} r_i^{-E} < 2$.

For simplicity, we write $g = f^{\beta}$, $r = r_i$. It suffices to show that:

$$r\left[1 + \lambda \min\left\{gr^{-E} - \frac{1}{s} - 1, 1\right\}\right] \geq 1 + \lambda \min\left\{g - \frac{1}{s} - 1, 1\right\}.$$

Let $h(r)$ be defined as:

$$h(r) = r\left[1 + \lambda \min\left\{gr^{-E} - \frac{1}{s} - 1, 1\right\}\right].$$

If $gr^{-E} - \frac{1}{s} - 1 \leq 1$, then we show $dh/dr \geq 0$, and thus the LHS of the above equation is minimized when $r = 1$, when it is an identity; thus the condition holds when $gr^{-E} - \frac{1}{s} - 1 \leq 1$.

$$\begin{aligned}\frac{dh}{dr} &= \left[1 + \lambda\left(gr^{-E} - \frac{1}{s} - 1\right)\right] - \lambda E g r^{-E} \\ &= 1 - \lambda\left((E-1)gr^{-E} + \frac{1}{s} + 1\right).\end{aligned}$$

As $-gr^{-E} \geq -(2 + 1/s)$, we have:

$$\frac{dh}{dr} \geq 1 - \lambda\left((E-1)(2 + 1/s) + \frac{1}{s} + 1\right).$$

After simplifying and ignoring positive terms, we obtain:

$$\frac{dh}{dr} \geq 1 - \lambda E\left(2 + \frac{1}{s}\right).$$

Thus, $\frac{dh}{dr} \geq 0$ if $\lambda E(2 + 1/s) \leq 1$.

Next, consider the case when $gr^{-E} - \frac{1}{s} - 1 > 1$. In this case, it suffices that

$$r[1+\lambda] \geq 1 + \lambda \min\left\{g - \frac{1}{s} - 1, 1\right\}.$$

The RHS is at most $1+\lambda$. Clearly the condition is true in this case. Thus the result of condition (2) holds in this case.

- **Case 3:** $z_i < 0$. If the condition of (2) holds, the argument is as in Case 1. So suppose the condition of (1) holds. Then,

$$p_i' = \left\lceil p_i \left(1 + \lambda \frac{z_i}{s_i}\right)\right\rceil.$$

By demand sensitivity and bounded elasticity (Definition 4.11), $x_i \geq \lfloor s_i f^{\beta} r_i^{-E} \rfloor$. So,

$$\begin{aligned} p_i' &\geq p_i \left[1 + \lambda\left(f^{\beta} r_i^{-E} - \frac{1}{s_i} - 1\right)\right] \\ &\geq p_i^L \frac{r_i}{f}\left[1 + \lambda\left(f^{\beta} r_i^{-E} - \frac{1}{s_i} - 1\right)\right]. \end{aligned}$$

It suffices that :

$$\frac{r_i}{f}\left[1 + \lambda\left(f^{\beta} r_i^{-E} - \frac{1}{s} - 1\right)\right] \geq 1 - \frac{\lambda}{s}.$$

For simplicity, we write $r = r_i$ and let $h(r, f)$ be defined as:

$$h(r,f) = \frac{r}{f}\left[1 + \lambda\left(f^{\beta} r^{-E} - \frac{1}{s} - 1\right)\right].$$

Then

$$\begin{aligned}
\frac{dh}{dr} &= \frac{1}{f}\left[1+\lambda\left(f^{\beta}r^{-E}-\frac{1}{s}-1\right)\right]-\frac{E\lambda}{f}f^{\beta}r^{-E} \\
&= \frac{1}{f}\left[1-\lambda\left(f^{\beta}r^{-E}(E-1)+\frac{1}{s}+1\right)\right] \\
&\geq \frac{1}{f}\left[1-\lambda\left(E+\frac{1}{s}\right)\right] && (\text{as } f^{\beta}r^{-E}\leq 1) \\
&\geq 0 && \left(\text{if } \lambda\left(E+\frac{1}{s}\right)\leq 1\right)
\end{aligned}$$

So $h(r,f)$ is minimized at $r=1$. Then we want

$$h(1,f) := \frac{1}{f}\left[1+\lambda\left(f^{\beta}-\frac{1}{s}-1\right)\right] \geq 1-\frac{\lambda}{s}. \tag{4.4}$$

Let $k(f)=\frac{1}{f}\left[1+\lambda\left(f^{\beta}-\frac{1}{s}-1\right)\right]$. Then,

$$\begin{aligned}
\frac{dk}{df} &= -f^{-2}\left[1+\lambda\left(f^{\beta}-\frac{1}{s}-1\right)\right]+\lambda\beta f^{-2+\beta} \\
&= -f^{-2}\left[1-\lambda\left(1+\frac{1}{s}\right)+\lambda f^{\beta}\left(1-\beta\right)\right] \\
&< 0. && \left(\text{if } \lambda\left(1+\frac{1}{s}\right)\leq 1\right)
\end{aligned}$$

As $f\leq 1$, the LHS is minimized at $f=1$. And at $f=1$, the condition in Equation 4.4 is true.

□

Corollary 4.3 *Suppose that* $s\geq\frac{aE}{\beta}$ *for a constant* $a\geq 1$ *and* $\frac{\beta}{aE}\leq 1$. *Let* $p_i^{T^-}=p_i^L\left(1-\frac{1}{aE}\right)$. *If* $\lambda E(2+1/s)\leq 1$, *then,*

1. *If* $p_1 \geq p_1^{T^-}$, *then* $p_i' \geq p_i^{T^-}$.

2. *If* $p_1 < p_1^{T^-}$, *then,*

$$\frac{p_i'}{p_i^{T^-}} \geq \frac{p_1}{p_1^{T^-}} \left[1 + \lambda \min \left\{ \left(\frac{p_1^{T^-}}{p_1} \right)^{\beta} - 1, 1 \right\} \right].$$

Proof. Note that $p_i^{T^-} \leq p_i^L(1-\lambda/s)$ as $\frac{1}{aE} \geq \frac{\lambda\beta}{aE} \geq \frac{\lambda}{s}$; so (1) holds for $p_1 \geq p_1^L$.

First, for $p_1 < p_1^{T^-}$, we show that

$$\frac{p_i^L}{p_1^L} \left[1 + \lambda \min \left\{ \left(\frac{p_1^L}{p_1} \right)^{\beta} - \frac{1}{s} - 1, 1 \right\} \right] \geq \frac{p_i^{T^-}}{p_1^{T^-}} \left[1 + \lambda \min \left\{ \left(\frac{p_1^{T^-}}{p_1} \right)^{\beta} - 1, 1 \right\} \right].$$

It suffices to show that:

$$\begin{aligned}
\left(\frac{p_1^L}{p_1} \right)^{\beta} &\geq \left(\frac{p_1^{T^-}}{p_1} \right)^{\beta} + \frac{1}{s} \\
\left(\frac{p_1^L}{p_1} \right)^{\beta} &\geq \left(\frac{p_1^L}{p_1} \right)^{\beta} \left(1 - \frac{1}{aE} \right)^{\beta} + \frac{1}{s} \\
0 &\geq \left(\frac{p_1^L}{p_1} \right)^{\beta} \left(-\frac{\beta}{aE} \right) + \frac{1}{s} \qquad (\text{if } \beta/aE \leq 1).
\end{aligned}$$

As $p_1 \leq p_1^L$, $s \geq \frac{\beta}{aE}$ suffices.

Next, consider the case when $p_1^{T^-} \leq p_1 \leq p_1^L$. In this case, we need to show that:

$$\frac{p_i^L}{p_1^L} \left[1 + \lambda \min \left\{ \left(\frac{p_1^L}{p_1} \right)^{\beta} - \frac{1}{s} - 1, 1 \right\} \right] \geq \frac{p_i^{T^-}}{p_1^{T^-}} \left(1 - \frac{\lambda}{s} \right).$$

Thus, it suffices that the LHS expression is more than $(1-\lambda/s)^{\frac{p_i^L}{p_1^L}}$, which is true. □

Lemma 4.9 *Starting with initial prices* $\mathbf{p}^I$*, after*

$$O\left(\frac{1}{\lambda}\left(\log\max_i \frac{p_i^L}{p_i^I} + \frac{1}{\beta}\log\frac{1}{\beta} + \frac{1}{\beta}\log E\right)\right)$$

rounds, for all goods i*,* $p_i \geq p_i^L - 2(p_i^L - p_i^{T^-})$*, if* $\frac{2}{aE} \leq 1$ *and* $\lambda E(2+1/s) \leq 1$*.*

Proof. By Corollary 4.3 (2), so long as $\left(\frac{p_1^{T^-}}{p_1}\right)^\beta \geq 2$, $p_i' \geq p_i(1+\lambda)$, and so in $O(1/\lambda)$ iterations, $\max_i \frac{p_i^{T^-}}{p_i}$ halves.

Let a_h be the number of rounds needed to reduce $\max_i \frac{p_i^{T^-}}{p_i}$ from at most $(1+\frac{1}{2^h})^{\frac{1}{\beta}}$ to at most $(1+\frac{1}{2^{h+1}})^{\frac{1}{\beta}}$ for $h = 0, 1, \ldots$. To count a_h, again, by Corollary 4.3 (2), note that in each round, $\max_i \frac{p_i^{T^-}}{p_i}$ reduces at least by a $(1+\lambda/2^{h+1})$ factor. Thus, a_h can be upper-bounded using:

$$(1+\lambda/2^{h+1})^{a_h} \geq \left(\frac{1+\frac{1}{2^h}}{1+\frac{1}{2^{h+1}}}\right)^{\frac{1}{\beta}} = \left(\frac{2(2^h+1)}{2^{h+1}+1}\right)^{\frac{1}{\beta}} = \left(1+\frac{1}{2^{h+1}+1}\right)^{\frac{1}{\beta}}.$$

This yields:

$$a_h \leq \frac{1}{\beta}\frac{\log\left(\frac{2(2^h+1)}{2^{h+1}+1}\right)}{\log\left(1+\lambda/2^{h+1}\right)} \leq \frac{2\cdot 2^{h+1}}{\lambda\beta}\log\left(1+\frac{1}{2^{h+1}+1}\right) \leq \frac{2}{\lambda\beta}.$$

To increase p_i so that $p_i \geq p_i^L - 2(p_i^L - p_i^{T^-})$ for all i means that

$$\frac{p_i}{p_i^{T^-}} \geq \frac{p_i^{T^-}}{p_i^{T^-}} - \frac{p_i^L - p_i^{T^-}}{p_i^{T^-}} \geq 1 - \left[\left(1-\frac{1}{aE}\right)^{-1} - 1\right] \geq 1 - \frac{2}{aE} \qquad \left(\text{as } \frac{1}{aE} \leq \frac{1}{2}\right)$$

Thus,

$$\left(\frac{p_i}{p_i^{T-}}\right)^{\beta} \geq 1 - \frac{2\beta}{aE} \qquad \left(\text{assuming } \frac{2}{aE} \leq 1\right).$$

By the argument of the previous paragraph, to increase $p_i \geq p_i^L - 2(p_i^L - p_i^{T-})$ for all i takes a further $O\left(\frac{1}{\lambda\beta}\log\left(\frac{aE}{\beta}\right)\right)$ rounds. □

Lemma 4.10 *After one round of price updates, if* $2\lambda E \leq 1$*, the prices* $\mathbf{p}'$ *satisfy:*

1. *If* $p_n \leq p_n^H$ *then* $\frac{p_i'}{p_i^H} \leq (1 + \lambda/s) + \frac{1}{p_i^H}$.

2. *If* $p_n \geq p_n^H$ *then*

$$\frac{p_i'}{p_i^H} \leq \frac{p_n}{p_n^H}\left[1 + \lambda\left(\left(\frac{p_n^H}{p_n}\right)^{\beta} + \frac{1}{s} - 1\right)\right] + \frac{1}{p_i^H}.$$

Proof. Let $f = \frac{p_n}{p_n^H}$ and $r_i = \frac{p_n}{p_n^H}\frac{p_i^H}{p_i}$.

1. **Case 1:** $z_i \leq 0$.

   If $p_n \leq p_n^H$ then as $p_i' \leq p_i$, the result of (1) holds for all i. So consider (2) (and $p_n \geq p_n^H$).

$$\begin{aligned} p_i' &= \left\lceil p_i\left(1 + \lambda\frac{z_i}{s_i}\right)\right\rceil \\ &\leq p_i + \min\left\{\lambda p_i\left(\frac{\lceil s_i f^{-\beta} r_i^E\rceil}{s_i} - 1\right) + 1, 0\right\} \\ &\leq p_i + \min\left\{\lambda p_i f^{-\beta} r_i^E + \frac{\lambda p_i}{s_i} - \lambda p_i + 1, 0\right\} \end{aligned}$$

So it suffices to show that for $f^{-\beta} r_i^E \leq 1$,

$$\frac{p_i}{p_i^H}\left[1+\lambda f^{-\beta} r_i^E - \lambda + \frac{\lambda}{s_i}\right] + \frac{1}{p_i^H} \leq \frac{p_n}{p_n^H}\left[1+\lambda\left(f^{-\beta}-1+\frac{1}{s}\right)\right] + \frac{1}{p_i^H}.$$

Or,

$$\frac{1}{r_i}\left[1+\lambda f^{-\beta} r_i^E - \lambda + \frac{\lambda}{s}\right] \leq 1+\lambda\left(f^{-\beta}-1+\frac{1}{s}\right).$$

Let $h(r,f)$ be defined as:

$$h(r,f) = \frac{1}{r}\left(1+\lambda f^{-\beta} r^E - \lambda + \frac{\lambda}{s}\right).$$

Then,

$$\begin{aligned} \frac{dh}{dr} &= -\frac{1}{r^2}\left[1-(E-1)\lambda f^{-\beta} r^E - \lambda + \frac{\lambda}{s}\right] \\ &\leq 0 \qquad \left(\text{if } E\lambda \leq 1 \text{ as } f^{-\beta} r^E \leq 1\right). \end{aligned}$$

So the LHS is maximized at $r=1$. Then, we want

$$1+\lambda\left(f^{-\beta}-1+\frac{1}{s}\right) \leq 1+\lambda\left(f^{-\beta}-1+\frac{1}{s}\right),$$

which is indeed true.

And we also need to show that when $f^{-\beta} r_i^E \geq 1$:

$$\frac{p_i}{p_i^H} \leq \frac{p_n}{p_n^H}\left(1+\lambda\left(f^{-\beta}-1+\frac{1}{s}\right)\right) + \frac{1}{p_i^H}. \tag{4.5}$$

It suffices that:

$$1 \leq r_i \left[1 + \lambda \left(r_i^{-E} - 1 + \frac{1}{s}\right)\right].$$

To this end, let $k(r)$ be defined as:

$$k(r) = r \left[1 + \lambda \left(r^{-E} - 1 + \frac{1}{s}\right)\right].$$

$\frac{dk}{dr}$ is given by:

$$\frac{dk}{dr} = 1 - (E-1)\lambda r^{-E} - \lambda + \frac{\lambda}{s}.$$

Thus, $\frac{dk}{dr} \geq 0$ if $\lambda E \leq 1$. So the RHS in Equation 4.5 is minimized at $r_i = 1$, and then we want $1 \leq 1 + \frac{\lambda}{s}$, which is true.

2. **Case 2:** $z_i \geq 0$. Again, suppose that $p_n \geq p_n^H$. As in Case 1, it suffices that

$$\frac{1}{r_i}\left[1 + \lambda \min\left\{2, f^{-\beta} r_i^E\right\} - \lambda + \frac{\lambda}{s}\right] \leq 1 + \lambda\left(f^{-\beta} - 1 + \frac{1}{s}\right). \quad (4.6)$$

Now we define $h(r, f)$ as:

$$h(r, f) = \frac{1}{r}\left[1 + \lambda \min\left\{2, f^{-\beta} r^E\right\} - \lambda + \frac{\lambda}{s}\right].$$

For $f^{-\beta} r^E \leq 2$, dh/dr is given by:

$$\begin{aligned}\frac{dh}{dr} &= \frac{1}{r^2}\left[(E-1)\lambda f^{-\beta} r^E - 1 + \lambda - \frac{\lambda}{s}\right] \\ &\leq \frac{1}{r^2}\left[(2E-1)\lambda - 1 - \frac{\lambda}{s}\right] \qquad \text{(since } f^{-\beta} r^E \leq 2\text{)}\end{aligned}$$

Thus, we obtain that $dh/dr \leq 0$ if $1 \geq (2E-1)\lambda$ and the LHS in Equation 4.6 is maximized at $r_i = 1$, when the inequality is an equality. So again the claim is true.

When $f^{-\beta} r_i^E \geq 2$, since the claim holds for $f^{-\beta} \overline{r}_i^E = 2$, and $\frac{1}{r_i} \leq \frac{1}{\overline{r}_i}$, the claim continues to hold.

Finally, suppose that $p_n \leq p_n^H$. Let $f = \frac{p_n^H}{p_n}$ and $r_i = \frac{p_i^H}{p_i}\frac{p_n}{p_n^H}$.

$$\begin{aligned}p_i' &= \left\lceil p_i \left(1 + \lambda \left\{ \frac{z_i}{s_i}, 1 \right\}\right) \right\rceil \\ &\leq p_i + \lambda p_i \min\left\{ \frac{\left\lceil f^\beta r_i^E s_i \right\rceil}{s_i} - 1, 1 \right\} + 1.\end{aligned}$$

So

$$\begin{aligned}\frac{p_i'}{p_i^H} &\leq \frac{p_i}{p_i^H}\left[1 + \lambda\left\{ f^\beta r_i^E + \frac{1}{s_i} - 1, 1 \right\}\right] + \frac{1}{p_i^H} \\ &\leq \frac{p_n}{p_n^H}\frac{1}{r_i}\left[1 + \lambda \min\{ f^\beta r_i^E + \frac{1}{s_i} - 1, 1\}\right] + \frac{1}{p_i^H}.\end{aligned}$$

And we want to show this is bounded by $(1 + \lambda/s) + \frac{1}{p_i^H}$.

It suffices that:

$$\frac{1}{fr_i}\left(1+\lambda\min\left\{f^{\beta}r_i^E-1,1\right\}\right)\leq 1.$$

Let $h(r,f)$ be defined as:

$$h(r,f)=\frac{1}{fr}\left[1+\lambda(f^{\beta}r^E-1)\right],\ \text{for } f^{\beta}r^E\leq 2.$$

Then

$$\begin{aligned}\frac{dh}{dr}&=-\frac{1}{fr^2}\left[1+\lambda(f^{\beta}r^E-1)\right]+\frac{E\lambda f^{\beta}r^E}{fr^2}\\&\leq -\frac{1}{fr^2}\left[1-2E\lambda\right] \qquad\qquad (\text{as } 1\leq f^{\beta}r^E\leq 2)\end{aligned}$$

Thus, $\frac{dh}{dr}\leq 0$ if $2E\lambda\leq 1$.

$h(r,f)$ is maximized at $r=1$, and

$$h(f,1)=\frac{1}{f}\left[1+\lambda(f^{\beta}-1)\right].$$

Now,

$$\frac{dh(f,1)}{df}=-\frac{1}{f^2}\left[1+\lambda(f^{\beta}-1)\right]+\frac{\beta\lambda f^{\beta}}{f^2}.$$

Thus, $\frac{dh(f,1)}{df}\leq 0$ as $\lambda\geq\lambda\beta$ and $\lambda\leq 1$.

Again, h is maximized at $f=1$, and $h(1,1)=1$. This verifies the condition for $f^{\beta}r_i^E\leq 2$.

But for $f^\beta r_i^E \geq 2$, the condition becomes

$$\frac{1}{fr_i}(1+\lambda) \leq 1.$$

This is verified by the fact that it holds for $f^\beta r_i^E = 2$.

□

Corollary 4.4 *Suppose that* $s \geq \frac{8}{\beta\Delta}$, $p_H \geq \frac{2}{\lambda\beta\Delta}$ *and* $2\beta\Delta \leq 1$. *Let* $p_i^{T+} = p_i^H(1+\Delta)$.

1. *If* $p_n \leq p_n^{T+}$, *then* $p_i' \leq p_i^{T+}$.

2. *If* $p_n \geq p_n^{T+}$, *then*

$$\frac{p_i'}{p_i^{T+}} \leq \frac{p_n}{p_n^{T+}}\left[1+\lambda\max\left\{\left(\frac{p_n^{T+}}{p_n}\right)^\beta - 1, -\frac{1}{2}\right\}\right].$$

Proof. Note that $p_i^{T+} \geq p_i^H(1+\lambda/s)+1$ so (1) is true when $p_i \leq p_i^H$.

First, for $p_n \geq p_n^{T+}$, if $\left(\frac{p_n^{T+}}{p_n}\right)^\beta \geq \frac{1}{2}$, we show that

$$p_n\frac{p_i^H}{p_n^H}\left[1+\lambda\left[\left(\frac{p_n^H}{p_n}\right)^\beta + \frac{1}{s} - 1\right]\right]+1 \;\leq\; \frac{p_n p_i^{T+}}{p_n^{T+}}\left[1+\lambda\left[\left(\frac{p_n^{T+}}{p_n}\right)^\beta - 1\right]\right],$$

Or,

$$\lambda\left(\frac{p_n^H}{p_n}\right)^\beta + \frac{\lambda}{s} + \frac{p_n^H}{p_n p_i^H} \leq \lambda\left(\frac{p_n^{T+}}{p_n}\right)^\beta.$$

As $p_n^{T+} = p_n^H(1+\Delta)$, it suffices that:

$$\lambda\left(\frac{p_n^H}{p_n}\right)^\beta + \frac{\lambda}{s} + \frac{1}{p_i^H}\frac{p_n^H}{p_n} \leq \lambda\left(\frac{p_n^H}{p_n}\right)^\beta (1+\Delta)^\beta.$$

It suffices that:

$$\frac{\lambda}{s} + \frac{1}{p_i^H}\frac{p_n^H}{p_n} \leq \lambda\beta\Delta\left(\frac{p_n^H}{p_n}\right)^\beta.$$

Thus, the following two conditions suffice:

$$\frac{\lambda}{s} \leq \frac{\lambda\beta\Delta}{2}\cdot\frac{1}{2}\left(\frac{p_n^H}{p_n^{T+}}\right)^\beta \tag{4.7}$$

$$\frac{1}{p_i^H} \leq \frac{\lambda\beta\Delta}{2}. \tag{4.8}$$

For the first of these two conditions, we have

$$\left(\frac{p_n^H}{p_n^{T+}}\right)^\beta = \left(\frac{1}{1+\Delta}\right)^\beta \leq 1 - \frac{\beta\Delta}{2}, \qquad \text{if } \beta\Delta \leq \frac{1}{2}.$$

It suffices that $\frac{\lambda}{s} \leq \frac{\lambda\beta\Delta}{8}$, or $\Delta \geq \frac{8}{\beta s}$. And for the second condition, $\Delta \geq \frac{2}{\lambda\beta p_H}$ suffices.

For $\left(\frac{p_n^{T+}}{p_n}\right)^\beta \leq \frac{1}{2}$, we need to show:

$$p_n\frac{p_i^H}{p_n^H}\left[1+\lambda\left[\left(\frac{p_n^H}{p_n}\right)^\beta + \frac{1}{s} - 1\right]\right] + 1 \leq p_n\frac{p_i^{T+}}{p_n^{T+}}\left[1-\frac{\lambda}{2}\right].$$

Or,

$$\lambda\left(\frac{p_n^H}{p_n}\right)^\beta + \frac{\lambda}{s} + \frac{p_n^H}{p_n p_i^H} \leq \frac{\lambda}{2} \qquad \text{suffices.}$$

It suffices that:

$$\begin{aligned} & \frac{\lambda}{2}(1+\Delta)^{-\beta} + \frac{\lambda}{s} + \frac{1}{p_i^H}\left(\frac{1}{2}\right)^{1/\beta} \leq \frac{\lambda}{2}. \\ \text{Or,} \quad & \frac{\lambda}{2}\left[1 - \frac{\beta\Delta}{2}\right] + \frac{\lambda}{s} + \frac{1}{2p_i^H} \leq \frac{\lambda}{2} \quad \text{suffices.} \\ \text{Or,} \quad & \frac{\lambda}{s} + \frac{1}{2p_i^H} \leq \frac{\lambda\beta\Delta}{4} \quad \text{suffices.} \end{aligned}$$

So $s \geq \frac{8}{\beta\Delta}$ and $p_i^H \geq \frac{2}{\lambda\beta\Delta}$ suffice.

Now consider $p_n^H \leq p_n \leq p_n^{T^+}$. Then we need to show:

$$\frac{p_n p_i^H}{p_n^H}\left[1 + \lambda\left[\left(\frac{p_n^H}{p_n}\right)^{\beta} + \frac{1}{s} - 1\right]\right] + 1 \leq p_i^{T^+}$$

Let $x = \frac{p_n}{p_n^H}$. Write $h(x)$ as:

$$h(x) := x p_i^H\left[1 + \lambda\left[x^{-\beta} + \frac{1}{s} - 1\right]\right] + 1$$

Then

$$\frac{dh}{dx} = p_i^H\left[1 + \lambda\left(x^{\beta} + \frac{1}{s} - 1\right)\right] - \beta\lambda p_i^H x^{\beta} \geq 0.$$

This is maximized when $x = \frac{p_n^{T+}}{p_n^H}$. The LHS becomes

$$\begin{aligned}
& \frac{p_n^{T+} p_i^H}{p_n^H} \left[1 + \lambda \left(\left(\frac{p_n^H}{p_n^{T+}} \right)^\beta + \frac{1}{s} - 1 \right) \right] + 1 \\
\geq \; & p_i^H (1+\Delta) \left(1 + \lambda \left[(1+\Delta)^{-\beta} + \frac{1}{s} - 1 \right] \right) + 1 \\
\geq \; & p_i^{T+} \left[1 + \lambda (1 - \beta\Delta + \frac{1}{s} - 1) \right] + 1 \qquad\qquad (\text{if } \beta\Delta \leq 1) \\
\geq \; & p_i^{T+} \left[1 - \lambda \left(\beta\Delta - \frac{1}{s} - \frac{1}{\lambda p_i^{T+}} \right) \right]
\end{aligned}$$

For this to be bounded by p_i^{T+}, it suffices that

$$\beta\Delta \geq \frac{1}{s} + \frac{1}{\lambda p_i^{T+}}, \qquad \text{or} \qquad s \geq \frac{2}{\beta\Delta} \text{ and } p_i^{T+} \geq \frac{2}{\lambda\beta\Delta}.$$

□

Corollary 4.5 *Suppose that* $s \geq \frac{8bE}{\beta}$ *for a constant* $b \geq 1$ *and* $p_H \geq \frac{2bE}{\lambda\beta}$, $\lambda E \leq 2$. *Let* $p_i^{T+} = p_i^H \left(1 + \frac{1}{bE}\right)$.

1. *If* $p_n \leq p_n^{T+}$, *then* $p_i' \leq p_i^{T+}$.

2. *If* $p_n \geq p_n^{T+}$, *then*

$$\frac{p_i'}{p_i^{T+}} \leq \frac{p_n}{p_n^{T+}} \left[1 + \lambda \max \left\{ \left(\frac{p_n^{T+}}{p_n} \right)^\beta - 1, -\frac{1}{2} \right\} \right].$$

Proof. The proof is immediate from Corollary 4.4 with $\Delta = \frac{1}{bE}$. □

Lemma 4.11 *Starting with initial prices* $\mathbf{p}^I$*, after*

$$O\left(\frac{1}{\lambda}\left(\log\max_i \frac{p_i^I}{p_i^H} + \frac{1}{\beta}\log\frac{1}{\beta} + \frac{1}{\beta}\log E\right)\right)$$

rounds, for all goods i*,* $p_i \le p_i^H + 2(p_i^{T+} - p_i^H)$*, if* $\frac{2}{bE} \le 1$*.*

Proof. By Corollary 4.5, so long as $\left(\frac{p_n^{T+}}{p_n}\right)^\beta \le \frac{1}{2}$, $p_i' \le p_i(1-\lambda/2)$, and so in $O(1/\lambda)$ iterations, $\max_i \frac{p_i}{p_i^{T+}}$ halves.

Let a_h be the number of rounds needed to increase $\max_i \frac{p_i^{T+}}{p_i}$ from at most $\left(1-\frac{1}{2^h}\right)^{\frac{1}{\beta}}$ to at most $\left(1-\frac{1}{2^{h+1}}\right)^{\frac{1}{\beta}}$ for $h = 1, 2, \ldots$. Using an argument similar to the one used in Lemma 4.9,

$$a_h = O\left(\frac{2^{h+1}}{\lambda\beta}\log\left(\frac{2^{h+1}-1}{2^{h+1}-2}\right)\right) = O\left(\frac{1}{\lambda\beta}\right).$$

To decrease p_i so that $p_i \le p_i^H + 2(p_i^{T+} - p_i^H)$ for all i means that

$$\frac{p_i}{p_i^{T+}} \le \frac{p_i^{T+}}{p_i^{T+}} + \frac{p_i^{T+} - p_i^H}{p_i^{T+}} \le 1 + \left[1 - \left(1 + \frac{1}{bE}\right)^{-1}\right] \le 1 + \frac{1}{2bE} \qquad \left(\text{as } \frac{1}{bE} \le \frac{1}{2}\right)$$

Thus,

$$\left(\frac{p_i^{T+}}{p_i}\right)^\beta \ge \left(1 - \frac{1}{2bE}\right)^{-\beta} \ge 1 - \frac{\beta}{4bE} \qquad \left(\text{assuming } \frac{1}{2bE} \le \frac{1}{2}\right).$$

By the argument of the previous paragraph, to decrease $p_i \le p_i^H + 2(p_i^{T+} - p_i^H)$ for all i takes a further $O\left(\frac{1}{\lambda\beta}\log\left(\frac{bE}{\beta}\right)\right)$ rounds. □

Lemmas 4.9 and 4.10 can be combined to yield the following theorem.

Theorem 4.5 *Starting with initial prices* $\mathbf{p}^I$*, after*

$$O\left(\frac{1}{\lambda}\left(\log\max_i \frac{p_i^I}{p_i^H} + \log\max_i \frac{p_i^L}{p_i^I} + \frac{1}{\beta}\log\frac{1}{\beta} + \frac{1}{\beta}\log E\right)\right)$$

rounds, if $\lambda \leq \frac{1}{cE}$*,* $s \geq \frac{32E}{\beta}$ *and* $\min_i p_i^H \geq \frac{8E}{\lambda\beta}$*, for all goods* i*,*

$$\begin{aligned} p_i &\geq p_i^L - 2(p_i^L - p_i^{T^-}) \\ p_i &\leq p_i^H + 2(p_i^{T^+} - p_i^H). \end{aligned}$$

Here, $\mathbf{p}^H$ *and* $\mathbf{p}^L$ *are a set of barely high and barely low prices, respectively, and* $p_i^{T^-} = p_i^L(1 - \frac{1}{4E})$*,* $p_i^{T^+} = p_i^H(1 + \frac{1}{4E})$*.*

Theorem 4.5 demonstrates that the price of each good i converges to the interval $\left[p_i^L\left(1 - \frac{1}{2E}\right), p_i^H\left(1 + \frac{1}{2E}\right)\right]$.

Corollary 4.6 *If for all goods* i*,*

$$p_i^L\left(1 - \frac{1}{2E}\right) \leq p_i \leq p_i^H\left(1 + \frac{1}{2E}\right),$$

and $p_i^H \leq p_i^L(1 + \frac{2}{cE})$*, then the demand of each good is bounded as* $x_i(\mathbf{p}) = O(s_i)$ *and thus the excess demand is* $O(s_i)$*, for* $c \geq 4$*,* $\lambda \leq \frac{1}{cE}$*,* $s \geq \frac{32E}{\beta}$ *and* $\min_i p_i^H \geq \frac{8E}{\lambda\beta}$*.*

Proof. Using bounded elasticity E with demand sensitivity β (Definition 4.11), we know that:

$$x_i(\mathbf{p}) \leq \left\lceil x_i(\mathbf{p}^H) \left(\frac{p_n^H}{p_n}\right)^{\beta} \left(\frac{p_n/p_n^H}{p_i/p_i^H}\right)^{E} \right\rceil.$$

Since $x_i(\mathbf{p}^H) \leq s_i$, $\frac{p_n}{p_n^H} \leq 1 + \frac{1}{2E}$ and $\frac{p_i}{p_i^H} \geq \frac{1-\frac{1}{2E}}{1+\frac{8}{cE}}$, it follows that:

$$\begin{aligned} x_i(\mathbf{p}) &\leq \left\lceil s_i \left(1 + \frac{1}{2E}\right)^{E-\beta} \left(1 + \frac{2}{cE}\right)^{E} \left(1 - \frac{1}{2E}\right)^{-E} \right\rceil \\ &\leq 8s_i + 1 \qquad \left(\text{if } \tfrac{2}{c} \leq \tfrac{1}{2}\right) \end{aligned}$$

Thus, $x_i(\mathbf{p}) = O(s_i)$. □

At prices $\mathbf{p}$, we define $|p_i z_i(\mathbf{p})|$ as the *misspending* on good i. The overall misspending in the market is simply $\sum_i |p_i z_i(\mathbf{p})|$. Recall that buyer j's wealth is denoted by w_j. The total wealth in the market is $\sum_j w_j$. If $z_i(\mathbf{p}) = O(s_i)$ for all i, the misspending is $O(\sum_i p_i s_i) = O(\sum_j w_j)$.

We conjecture that the price update protocol converges to prices with misspending $O(E \sum_i s_i)$.

4.6 Relationship Between Discontent and ϵ-closeness in Utility

In this section, we discuss the relationship between the two measures of distance from equilibria in the divisible setting. Specifically, for a fixed set of prices $\mathbf{p}$, let $\mathbf{x}^*$ denote the optimal allocation of a buyer b who has initial

wealth w. We are interested in determining the discontent of a bundle $\mathbf{x}$ that is ϵ-close to $\mathbf{x}^*$ in terms of utility; i.e. $u(\mathbf{x}) = (1-\epsilon)u(\mathbf{x}^*)$. Given a utility $\overline{u}$ (*not* a utility function), a set of prices $\mathbf{p}$, define $\psi(\overline{u}, \mathbf{p})$ as follows:

$$\psi(\overline{u}, \mathbf{p}) = \min\{\mathbf{p} \cdot \mathbf{x} \mid u(\mathbf{x}) \geq \overline{u}\}$$

In words, given a desired utility level $\overline{u}$ and set of prices $\mathbf{p}$, $\psi(\overline{u}, \mathbf{p})$ is the minimum amount of wealth required to afford a bundle with utility value $\overline{u}$. In the economics literature, $\psi(\cdot)$ is also known as the *expenditure function* (see [66], page 59).

The discontent for the bundle $\mathbf{x}$ is then given by $\psi(u(\mathbf{x}^*), \mathbf{p}) - \psi((1-\epsilon)u(\mathbf{x}^*), \mathbf{p})$. In the divisible setting, it is fairly standard to assume that $\psi(u(\mathbf{x}^*), \mathbf{p}) = w$ [5] and thus the expression for the discontent reduces to $w - \psi((1-\epsilon)u(\mathbf{x}^*), \mathbf{p})$. Define the relative discontent as

$$\epsilon_d = 1 - \frac{\psi((1-\epsilon)u(\mathbf{x}^*), \mathbf{p})}{w}$$

Theorem 4.6 *If the underlying utility function $u(\cdot)$ is concave and $\psi(0, \mathbf{p}) = 0$, then the relative discontent is no smaller than the ϵ-closeness in utility. That is $\epsilon_d \geq \epsilon$.*

Proof. It suffices to show that $\psi((1-\epsilon)u(\mathbf{x}^*), \mathbf{p}) \leq (1-\epsilon)w$. It is known that if the utility function $u(\cdot)$ is concave, then the expenditure function $\psi(\cdot)$

[5] If the buyer has utility for money, this is necessarily the case.

is convex in $\overline{u}$. Thus, for any $\epsilon \in [0,1]$, we have

$$\begin{aligned} \psi(\epsilon 0 + (1-\epsilon)u(\mathbf{x}^*), \mathbf{p}) &\leq \epsilon\psi(0,\mathbf{p}) + (1-\epsilon)\psi(u(\mathbf{x}^*),\mathbf{p}) \\ &= (1-\epsilon)w \end{aligned}$$

□

For preference orderings for which an increase in wealth by a factor f results in the optimal allocation increasing the amount of each good by the same factor f, if the utility function also grows by this factor f, then the two approximation measures are the same. Cobb-Douglas and CES utilities meet this criteria.

However, when increases in wealth result in non-proportionate spending increases on the different goods, this need not be true any more. Indeed for the two-good utility function $x^{1/s} + y^{(s-1)/s}$, given fixed prices, as the wealth w of a buyer tends to zero a decrease of w to $w(1-\epsilon)$ results in a change of optimal utility from u to a value that tends to $u(1-\epsilon/s)$, i.e. giving a multiplicative gap of up to s in the two approximation factors.

Bibliography

[1] Shivani Agarwal and Partha Niyogi. Stability and generalization of bipartite ranking algorithms. In *Conference on Learning Theory (COLT 2005)*, pages 32–47. Springer, 2005.

[2] Kostyantyn Archangelsky. Efficient algorithm for checking multiplicity equivalence for the finite $z-\sigma^*$-automata. In *Developments in Language Theory (DLT 2002)*, pages 283–289. Springer, 2002.

[3] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM (JACM)*, 45(3):501–555, 1998.

[4] Kenneth Arrow and Gerard Debreu. Existence of an equilibrium for a competitive economy. *Econometrica*, 22(3):265–290, 1954.

[5] Laurence Ausubel. Walrasian tatonnement for discrete goods. Technical report, University of Maryland, 2005.

[6] Baruch Awerbuch, Yossi Azar, and Rohit Khandekar. Fast load balancing via bounded best response. In *Symposium on Discrete Algorithms (SODA 2008)*, pages 314–322. ACM Press, 2008.

[7] Kazuoki Azuma. Weighted sums of certain dependent random variables. In *Tohoku Mathematical Journal*, volume 19, pages 357–367, 1967.

[8] Mikhail Belkin, Irina Matveeva, and Partha Niyogi. Regularization and semi-supervised learning on large graphs. In *Conference on Learning Theory (COLT 2004)*, pages 624–638. Springer, 2004.

[9] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems (NIPS 2001)*, pages 585–591. MIT Press, 2001.

[10] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization; a geometric framework for learning from examples. Technical Report TR-2004-06, University of Chicago, 2004.

[11] Christian Berg, Jens Peter Reus Christensen, and Paul Ressel. *Harmonic Analysis on Semigroups*. Springer-Verlag: Berlin-New York, 1984.

[12] Jean Berstel and Christophe Reutenauer. *Rational Series and Their Languages*. Springer-Verlag: Berlin-New York, 1988.

[13] Olivier Bousquet and André Elisseeff. Algorithmic stability and generalization performance. In *Advances in Neural Information Processing Systems (NIPS 2000)*, pages 196–202. MIT Press, 2000.

[14] Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of Machine Learning Research (JMLR)*, 2:499–526, 2002.

[15] Pascal Bridel. Patinkin, Walras and the "money in the utility function" tradition. *European Journal of the History of Economic Thought*, 9(2):268–292, 2002. Available at `http://ideas.repec.org/a/taf/eujhet/v9y2002i2p268-292.html`.

[16] Rafael C. Carrasco. Accurate computation of the relative entropy between stochastic regular grammars. *Informatique Théorique et Applications (ITA)*, 31(5):437–444, 1997.

[17] Olivier Chapelle, Vladimir Vapnik, and Jason Weston. Transductive inference for estimating values of functions. In *Neural Information Processing Systems (NIPS 1999)*, pages 421–427. MIT Press, 1999.

[18] Lihua Chen, Yinyu Ye, and Jiawei Zhang. A note on equilibrium pricing as convex optimization. In *Workshop on Internet and Network Economics (WINE 2007)*, pages 7–16. Springer, 2007.

[19] Ning Chen, Xiaotie Deng, Xiaoming Sun, and Andrew Chi-Chih Yao. Fisher equilibrium price with a class of concave utility functions. In *Eu-*

ropean Symposium on Algorithms (ESA 2004), pages 169–179. Springer, 2004.

[20] Wei Chu and Selvaraj Sathiya Keerthi. New approaches to support vector ordinal regression. In *International Conference on Machine Learning (ICML 2005)*, pages 145–152. ACM Press, 2005.

[21] Bruno Codenotti, Benton McCune, and Kasturi Varadarajan. Market equilibrium via the excess demand function. In *Symposium on Theory of Computing (STOC 2005)*, pages 74–83. ACM Press, 2005.

[22] Bruno Codenotti, Sriram Pemmaraju, and Kasturi Varadarajan. The computation of market equilibria. *SIGACT News*, 35(4):23–37, 2004.

[23] Bruno Codenotti, Sriram Pemmaraju, and Kasturi Varadarajan. On the polynomial time computation of equilibria for certain exchange economies. In *Symposium on Discrete Algorithms (SODA 2005)*, pages 72–81. ACM Press, 2005.

[24] Bruno Codenotti, Amin Saberi, Kasturi Varadarajan, and Yinyu Ye. Leontief economies encode nonzero sum two-player games. In *Symposium on Discrete Algorithms (SODA 2006)*, pages 659–667. ACM Press, 2006.

[25] Richard Cole and Lisa Fleischer. Fast-converging tatonnement algorithms for one-time and ongoing market problems. In *Symposium on Theory of Computing (STOC 2008), to appear*. ACM Press, 2008.

[26] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. The MIT Press: Cambridge, MA, 1992.

[27] Corinna Cortes and Mehryar Mohri. AUC optimization vs. error rate minimization. In *Advances in Neural Information Processing Systems (NIPS 2003)*. MIT Press, 2004.

[28] Corinna Cortes and Mehryar Mohri. On transductive regression. In *Advances in Neural Information Processing Systems (NIPS 2006)*, pages 305–312. MIT Press, 2007.

[29] Corinna Cortes, Mehryar Mohri, Dmitry Pechyony, and Ashish Rastogi. Stability of transductive regression algorithms. In *International Conference on Machine Learning (ICML 2008), to appear*. ACM Press, 2008.

[30] Corinna Cortes, Mehryar Mohri, and Ashish Rastogi. On the computation of some standard distances between probabilistic automata. In *Conference on Implementation and Application of Automata (CIAA 2006)*, pages 137–149. Springer-Verlag, 2006.

[31] Corinna Cortes, Mehryar Mohri, and Ashish Rastogi. An alternative ranking problem for search engines. In *Workshop on Experimental Algorithms (WEA 2007)*, pages 1–22. Springer, 2007.

[32] Corinna Cortes, Mehryar Mohri, and Ashish Rastogi. l_p distance and equivalence of probabilistic automata. *International Journal of Foundations of Computer Science*, 18(4):761–779, 2007.

[33] Corinna Cortes, Mehryar Mohri, and Ashish Rastogi. Magnitude-preserving ranking algorithms. In *International Conference on Machine Learning (ICML 2007)*, pages 169–176. ACM Press, 2007.

[34] Corinna Cortes, Mehryar Mohri, Ashish Rastogi, and Michael Riley. On the computation of the relative entropy of probabilistic automata. *International Journal of Foundations of Computer Science*, 19(1):219–242, 2007.

[35] Corinna Cortes, Mehryar Mohri, and Jason Weston. A general regression technique for learning transductions. In *International Conference on Machine learning (ICML 2005)*, pages 153–160. ACM Press, 2005.

[36] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., New York, 1991.

[37] Koby Crammer and Yoram Singer. Pranking with ranking. In *Advances in Neural Information Processing Systems (NIPS 2001)*, pages 641–647. MIT Press, 2001.

[38] Imre Csiszar and Janos Korner. *Information Theory: Coding Theorems for Discrete Memoryless Systems*. Akademiai Kiado, 1997.

[39] Karel Culik II and Jarkko Kari. Digital images and formal languages. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 599–616. Springer, 1997.

[40] Xiaotie Deng, Christos Papadimitriou, and Shmuel Safra. On the complexity of price equilibria. *Journal of Computer System Sciences (JCSS)*, 67:311–324, 2003.

[41] Nikhil Devanur and Vijay Vazirani. The spending constraint model for market equilibrium: algorithmic, existence and uniqueness results. In *Symposium on Theory of Computing (STOC 2004)*, pages 519–528. ACM Press, 2004.

[42] Pierre Dupont, Franois Denis, and Yann Esposito. Links between probabilistic automata and hidden markov models: probability distributions, learning models and induction algorithms. In *Pattern Recognition*, volume 38, pages 1349–1371, 2005.

[43] Richard Durbin, Sean Eddy, Anders Krogh, and Graeme Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

[44] Samuel Eilenberg. *Automata, Languages and Machines*, volume A–B. Academic Press, 1974–1976.

[45] Ran El-Yaniv and Dmitry Pechyony. Stable transductive learning. In *Conference on Learning Theory (COLT 2006)*, pages 35–49. Springer, 2006.

[46] Ran El-Yaniv and Dmitry Pechyony. Transductive rademacher complexity and its applications. *Journal of Machine Learning Research (JMLR)*, 2007.

[47] Lars Engebretsen and Jonas Holmerin. Clique is hard to approximate within $n^{1-o(1)}$. In *Proceedings of the 27th International Colloquium on Automata, Languages and Programming (ICALP 2000)*, pages 2–12, London, UK, 2000. Springer-Verlag.

[48] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. In Jude W. Shavlik, editor, *Proceedings of the 15th International Conference on Machine Learning*, pages 170–178, Madison, US, 1998. Morgan Kaufmann Publishers, San Francisco, US.

[49] Michael R. Garey and David S. Johnson. *Computers and Intractability.* Freeman and Company, New York, 1979.

[50] Rahul Garg and Sanjiv Kapoor. Auction algorithms for market equilibrium. In *Symposium on Theory of Computing*, pages 511–518, 2004.

[51] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore, 1996.

[52] Faruk Gul and Ennio Stacchetti. Walrasian equilibrium with gross substitutes. *Journal of Economic Theory*, 87(1):95–124, 1999.

[53] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[54] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Large margin rank boundaries for ordinal regression. In Smola, Bartlett, Schoelkopf, and Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 115–132. MIT Press, Cambridge, MA, 2000.

[55] Kamal Jain. A polynomial time algorithm for computing the Arrow-Debreu market equilibrium for linear utilities. In *Foundations of Computer Science (FOCS 2004)*, pages 286–294. IEEE Computer Society, 2004.

[56] Kamal Jain, Mohammad Mahdian, and Amin Saberi. Approximating market equilibria. In *Approximation, Randomization, and Combinatorial Optimization (RANDOM-APPROX 2003)*, pages 98–108, 2003.

[57] Kamal Jain and Kasturi Varadarajan. Equilibria for economies with production: constant-returns technologies and production planning constraints. In *Symposium on Discrete algorithm (SODA 2006)*, pages 688–697. ACM Press, 2006.

[58] Kamal Jain and Vijay V. Vazirani. Eisenberg-gale markets: algorithms and structural properties. In *Symposium on Theory of computing (STOC 2007)*, pages 364–373. ACM Press, 2007.

[59] Kamal Jain, Vijay V. Vazirani, and Yinyu Ye. Market equilibria for homothetic, quasi-concave utilities and economies of scale in production. In *Symposium on Discrete Algorithms (SODA 2005)*, pages 63–71, 2005.

[60] Thorster Joachims. Evaluating retrieval performance using clickthrough data. In *Workshop on Mathematical, Formal Methods in Information Retrieval*, 2002.

[61] Ronald M. Kaplan and Martin Kay. Regular models of phonological rule systems. *Computational Linguistics*, 20(3):331–378, 1994.

[62] Anders Krogh, Michael Brown, I. Saira Mian, Kimmen Sjolander, and David Haussler. Hidden markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology*, 235(5):1501–1531, 1994.

[63] Werner Kuich and Arto Salomaa. *Semirings, Automata, Languages*. Number 5 in EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Berlin, Germany, 1986.

[64] Daniel J. Lehmann. Algebraic structures for transitive closures. *Theoretical Computer Science*, 4:59–76, 1977.

[65] Michael Mandel, Graham Poliner, and Dan Ellis. Support vector machine active learning for music retrieval. *Multimedia Systems*, 12(1):3–13, 2006.

[66] Andreu Mas-Collel, Michael D. Whinston, and Jerry R. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[67] Peter McCullagh. Regression models for ordinal data. *Journal of the Royal Statistical Society B*, 42(2), 1980.

[68] Peter McCullagh and John A. Nelder. *Generalized Linear Models*. Chapman & Hall, London, 1983.

[69] Colin McDiarmid. On the method of bounded differences. In *Surveys in Combinatorics*, pages 148–188. Cambridge University Press, Cambridge, 1989.

[70] Colin McDiarmid. Concentration. In *Probabilistic Methods for Algorithmic Discrete Mathematics*, pages 195–248, 1998.

[71] Paul Milgrom and Bruno Strulovici. Concepts and properties of substitute goods. Technical Report 2006-W02, Economics Group, Nuffield College, University of Oxford, 2006.

[72] Mehryar Mohri. Compact representations by finite-state transducers. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 204–209, Morristown, NJ, USA, 1994. Association for Computational Linguistics.

[73] Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2), 1997.

[74] Mehryar Mohri. General algebraic frameworks and algorithms for shortest-distance problems. Technical Memorandum 981210-10TM, AT&T Labs - Research, 62 pages, 1998.

[75] Mehryar Mohri. Generic epsilon-removal and input epsilon-normalization algorithms for weighted transducers. *International Journal of Foundations of Computer Science*, 13(1):129–143, 2002.

[76] Mehryar Mohri. Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350, 2002.

[77] Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. Weighted automata in text and speech processing. In *Proceedings of the 12th biennial European Conference on Artificial Intelligence (ECAI-96), Workshop on Extended finite state models of language, Budapest, Hungary.* John Wiley and Sons, Chichester, 1996.

[78] Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. Weighted finite-state transducers in speech recognition. *Computer Speech and Language*, 16(1):69–88, 2002.

[79] Netflix. Netflix prize, 2006. `http://www.netflixprize.com/`.

[80] Azaria Paz. *Introduction to probabilistic automata.* Academic Press, New York, 1971.

[81] Michael Oser Rabin. Probabilistic automata. In *Information and Control*, volume 6, pages 230–245, 1963.

[82] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Joural of the ACM*, pages 267–296, 1990.

[83] Cynthia Rudin, Corinna Cortes, Mehryar Mohri, and Robert E. Schapire. Margin-based ranking meets boosting in the middle. In *Proceedings of Conference on Learning Theory 2005*, volume 3359 of *Lecture Notes in Computer Science*, pages 63–78. Springer, Heidelberg, Germany, June 2005.

[84] Rune B. Lyngsø and Christian N. S. Pederson. The consensus string problem and the complexity of comparing hidden markov models. *Journal of Computer and System Sciences*, 65(3):545–569, 2002.

[85] Arto Salomaa and Matti Soittola. *Automata-Theoretic Aspects of Formal Power Series*. Springer-Verlag, 1978.

[86] Paul Samuelson. *Foundations of Economic Analysis*. Harvard University Press, 1947.

[87] Iso J. Schoenberg. Metric spaces and positive definite functions. *Transactions of the American Mathematical Society*, 44(3):522–536, november 1938.

[88] Bernhard Schölkopf and Alex Smola. *Learning with Kernels*. MIT Press: Cambridge, MA, 2002.

[89] Bernhard Schölkopf and Alexander J. Smola. A short introduction to learning with kernels. In S. Mendelson and A. J. Smola, editors, *Machine Learning, Proceedings of the Summer School, Australian National University*, pages 41–64. Springer, 2003.

[90] Marcel-Paul Schützenberger. On the definition of a family of automata. *Information and Control*, 4, 1961.

[91] Dale Schuurmans and Finnegan Southey. Metric-based methods for adaptive model selection and regularization. *Machine Learning*, 48:51–84, 2002.

[92] Amnon Shashua and Anat Levin. Ranking with large margin principle: Two approaches. In *Advances in Neural Information Processing Systems (NIPS 2003)*, 2003.

[93] Yoram Singer and Manfred K. Warmuth. Training algorithms for hidden markov models using entropy based distance functions. In *Advances in Neural Information Processing Systems (NIPS 1997)*, pages 641–647. The MIT Press, 1997.

[94] Adam Smith. *An Inquiry into the Nature and Causes of the Wealth of Nations*. University of Chicago Press; Facsimile edition, 1776.

[95] Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS 1996)*, page 627, Washington, DC, USA, 1996. IEEE Computer Society.

[96] Larry J. Stockmeyer and Albert R. Meyer. Word problems requiring exponential time. In *Proceedings of the* 5^{th} *Annual ACM Symposium on Theory of Computing*. Association for Computing Machinery, New York, 1-9., 1973.

[97] Josh Tenenbaum, Vin de Silva, and John Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000.

[98] Flemming Topsøe. Some inequalities for information divergence and related measures of discrimination. *IEEE Trans. Inform. Theory*, 46:1602–1609, 2000.

[99] Wen-Guey Tzeng. A polynomial-time algorithm for the equivalence of probabilistic automata. *Foundations of Computer Science*, pages 216–227, 1992.

[100] Hirofumi Uzawa. Walras's tatonnement in the theory of exchange. *Review of Economic Studies*, 27:182–194, 1960.

[101] Vladimir N. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer, Berlin, 1982.

[102] Vladimir N. Vapnik. *Statistical Learning Theory.* Wiley-Interscience, New York, 1998.

[103] Grace Wahba. *Spline models for observational data.* Society for Industrial and Applied Mathematics, 1990.

[104] Léon Walras. *Elements of Pure Economics.* Harvard University Press, 1874.

[105] Mingrui Wu and Bernhard Scholkopf. Transductive classification via local learning regularization. In *Artificial Intelligence and Statistics (AISTATS 2007)*, 2007.

[106] Dengyong (Denny) Zhou, Olivier Bousquet, Thomas N. Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems (NIPS 2004)*. MIT Press, 2004.

[107] Xiaojin Zhu, Zoubin Ghahramani, and John D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *International Conference on Machine Learning (ICML 2003)*, pages 912–919. ACM Press, 2003.