A Kernel Method for Market Clearing

Sébastien Lahaie
Yahoo! Research
New York, NY 10018
lahaies@yahoo-inc.com

Abstract

The problem of market clearing in an economy is that of finding prices such that supply meets demand. In this work, we propose a kernel method to compute nonlinear clearing prices for instances where linear prices do not suffice. We first present a procedure that, given a sample of values and costs for a set of bundles, implicitly computes nonlinear clearing prices by solving an appropriately formulated quadratic program. We then use this as a subroutine in an elicitation procedure that queries demand and supply incrementally over rounds, only as much as needed to reach clearing prices. An empirical evaluation demonstrates that, with a proper choice of kernel function, the method is able to find sparse nonlinear clearing prices with much less than full revelation of values and costs. When the kernel function is not suitable to clear the market, the method can be tuned to achieve approximate clearing.

1 Introduction

It is standard in microeconomic modeling to assume that utilities are concave and costs convex. These assumptions ensure that *linear* prices exist that balance demand and supply, where one price is assigned to each item and the price of a bundle is simply the sum of item prices. But there are many realistic conditions under which such assumptions may fail to hold, such as complementary items or economies of scale. In such cases *nonlinear* prices might be needed to clear the market.

The problem of finding nonlinear clearing prices is challenging because such prices may be difficult to succinctly describe, let alone efficiently compute. In this work, we propose a kernel method for market clearing to address both of these issues. We view nonlinear prices as linear prices with respect to some high-dimensional representation of bundles. This perspective is drawn directly from the field of kernel methods in machine learning, in particular support vector machines, where nonlinear classifiers are construed as linear classifiers in high-dimensional feature space [Shawe-Taylor and Cristianini, 2004].

We consider a simple model with one buyer and one seller. In an environment with multiple buyers and sellers, our two agents would correspond to *representative* buyers and sellers that enact the aggregate demand and supply behavior of each side of the economy. Thus there is little loss of generality in considering just a buyer and a seller in this setting, and we opt for this simpler two-agent model for this initial work.

The problem at hand is that of finding prices such that the bundle that maximizes the buyer's utility simultaneously maximizes the seller's profit; it is in this sense that supply meets demand. We make no convexity assumptions on the value and cost functions of the buyer and seller. Now, no single clearing method can hope to perform universally well under such general conditions. A kernel method has the modularity to adopt an appropriate kernel function to clear any particular instance of value and cost functions.

We first provide a quadratic program that, given a sample of values and costs for a set of bundles, computes nonlinear clearing prices using only inner-products of high-dimensional bundle representations; this is the usual notion of a kernel method. Unlike typical machine learning scenarios, however, a sample of values and costs is not just given to us, it must be elicited. Thus we embed our quadratic program as a subroutine in an elicitation procedure that queries demand and supply incrementally over rounds.

Kernel methods have been applied to *single-agent* preference elicitation, where the goal is to recover or approximate an agent's preferences in order to act on its behalf or assist it on some tasks (i.e., for decision support). Domshlak and Joachims [2006] provide a method for generating an ordinal utility function given qualitative preference statements. Chappelle and Harchaoui [2005] and Evgeniou et al. [2005] apply support vector machines to conjoint analysis (preference elicitation for marketing purposes).

We are not aware of any applications of kernel methods in the context of *mutli-agent* preference elicitation, where the goal is not to recover preferences in any complete way, but rather to solve some ulterior problem involving all the agents (such as market clearing). Our elicitation procedure is similar to the work of Lahaie and Parkes [2004], who give an auction-style protocol that leads to clearing prices. They fit valuation functions to the agent's reported values using techniques from computational learning theory, and ultimately reach clearing prices on the basis of these valuations. Our method bypasses the step of fitting functions and directly computes clearing prices based on sample information of values and costs.

2 The Model

There is one buyer, one seller, and m items. A *bundle* is a subset of the items. We associate each bundle with its indicator vector, and denote the set of bundles by $X = \{0,1\}^m$. We write $x \leq x'$ to denote that bundle x is contained in bundle x' (the inequality is understood component-wise). The buyer has a value function $v : X \to \mathbf{R}_+$ denoting how much it is willing to pay for each bundle. The seller has a cost function $c : X \to \mathbf{R}_+$ denoting how much it would cost to produce each bundle. We assume that each function is monotone: $v(x) \leq v(x')$ whenever $x \leq x'$, and the same for c. We also assume $v(\mathbf{0}) = c(\mathbf{0}) = 0$.

A bundle is *efficient* if it maximizes the gains from trade: value created minus cost of production. Formally, $x \in X$ is efficient if $x \in \arg\max_{x' \in X} v(x') - c(x')$. Together with an efficient bundle, we wish to find prices $p : X \to \mathbf{R}$ such that at these prices, supply meets demand. Let

$$\begin{aligned} D(p) &= \arg\max_{x \in X}\{v(x) - p(x)\} \\ S(p) &= \arg\max_{x \in X}\{p(x) - c(x)\} \end{aligned}$$

In words, $D(p)$ is the set of bundles that maximizes the buyer's *utility*—value minus price—and similarly $S(p)$ are those bundles that maximize the seller's *profit*—revenue minus cost. Prices p are *clearing prices* if $D(p) \cap S(p) \neq \emptyset$. At clearing prices, there is some bundle that simultaneously maximizes the buyer's utility and the seller's profit; in this sense, supply meets demand and the market clears.

Clearing prices are important because they decentralize the problem of efficient trade. If $x^* \in D(p) \cap S(p)$, then for any other bundle x we have

$$\begin{aligned} v(x^*) - p(x^*) &\geq v(x) - p(x) \\ p(x^*) - c(x^*) &\geq p(x) - c(x) \end{aligned}$$

and adding these two inequalities shows that $v(x^*) - c(x^*) \geq v(x) - c(x)$. Thus x^* is efficient. In our model *nonlinear* clearing prices always exist because observe that we can simply take $p = v$ or $p = c$. As mentioned in the introduction, convexity assumptions would imply that *linear* clearing prices exist; formally, these can be described by a vector $p \in \mathbf{R}^m$ so that the price of a bundle $x \in X$ is the usual inner product $\langle p, x \rangle = \sum_{j=1}^m p_j x_j$.

We need to explain how value and cost information will be provided to our algorithms. Because the domain of bundles is exponentially large, passing an encoded description of v and c would be infeasible in general for even moderate m (such as $m = 30$). In fact, one main goal of our work is to try to find clearing prices by eliciting as little value and cost information as possible. Thus we will just assume that our algorithms have oracle access to the value and cost functions through two kinds of queries. We assume that agents respond truthfully to queries, and leave the question of incentive-compatibility to future work. On a *value query*, the buyer is presented a bundle x and replies with $v(x)$. On an *ϵ-demand query*, the buyer is presented a bundle x, prices p, and an $\epsilon \geq 0$; the buyer returns any bundle that maximizes its utility at prices p within an additive error of ϵ, breaking ties however it wishes, with the exception that if x applies then it is returned. The notions of *cost query* and *ϵ-supply query* are defined analogously for the seller. Because prices are communicated to the agents in demand and supply queries, we must ensure that they have succinct encodings for such queries to be practical. This is another goal of our work, for which we draw on techniques from kernel methods.

3 Kernels

To compute nonlinear clearing prices, we view these as linear prices in a higher-dimensional space to which we map the bundles via a mapping $\phi : X \to \mathbf{R}^M$, where $M \geq m$. Entry i in the vector $\phi(x)$ defines the value of the ith "feature" of bundle x, for $i = 1, \ldots, M$. Now, given $p \in \mathbf{R}^M$, the price of bundle x is $\langle p, \phi(x) \rangle$. The mapping ϕ therefore indirectly describes how each bundle is priced.

We might need $M \gg m$ to ensure that clearing prices exist, but with large M we cannot explicitly exhibit vectors $\phi(x)$ or prices p drawn from $\mathbf{R}^M$. The key insight of kernel methods is known as the "kernel trick". Instead of working with vectors in $\mathbf{R}^M$, which may be impractical or even infeasible, one formulates the relevant problem (e.g., classification, regression) as a mathematical program that relies only on the inner products $\langle \phi(x_i), \phi(x_j) \rangle$. What makes this practical is that, for many kinds of mappings, the inner products can be efficiently evaluated in time that does not depend on M.

A *kernel function* κ computes the inner product of the images of two bundles under a mapping $\phi : \mathbf{R}^m \to \mathbf{R}^M$, so that $\kappa(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle$. A fundamental result on kernels states that if a function $\kappa : X \times X \to \mathbf{R}$ with a countable domain is the kernel function of some map ϕ, then for any finite sample $\{x_1, \ldots, x_k\}$ from X, the $k \times k$ *kernel matrix* K with entries $K_{ij} = \kappa(x_i, x_j)$ is positive semi-definite (see, e.g., [Shawe-Taylor and Cristianini, 2004]).

Instances of useful kernel functions for various classification and regression tasks abound in the machine learning literature. By way of example, we list here those kernels that we use later for the empirical evaluation of our method.

Linear At one extreme we have the kernel $\kappa(x_i, x_j) = \langle x_i, x_j \rangle$ which simply corresponds to the identity map $\phi(x) = x$. We would use this kernel in the method developed later if we were to try to find linear clearing prices.

Identity At the other extreme, we have the kernel function $\kappa(x_i, x_j) = 1$ if $x_i = x_j$, and 0 otherwise. List all the possible bundles as $x_1, x_2, \ldots, x_{2^m}$. If we define $e_i \in \mathbf{R}^{2^m}$ to be the unit vector that has a 1 in entry i (and zeroes in the remaining), then the implied map is simply $\phi(x_i) = e_i$. This corresponds to the case were each bundle is priced explicitly, and therefore clearing prices are guaranteed to exist.

All-subsets The *all-subsets* kernel maps bundles into vectors in $\mathbf{R}^{2^m}$. The range has a dimension for each bundle $x \in X$. We have $\phi_x(x') = 1$ if $x \leq x'$ and 0 otherwise. Thus the price of a bundle is the aggregate of the prices on all its subsets. The actual kernel function is defined as $\kappa(x, x') = \prod_{i=1}^m (1 + x_i x'_i)$, which can be evaluated in linear time. With the ability to implicitly price subsets of commodities, this kernel seems well suited to market clearing.

Gaussian This is one of the most widely used kernels in machine learning. It is defined as $\kappa(x, x') =$

$\exp(-\|x - x'\|^2 / 2\sigma^2)$. (Here and everywhere else, $\|\cdot\|$ refers to the Euclidean norm.) The parameter σ controls the flexibility of the kernel. With smaller σ, it is better able to fit arbitrary functions, and the kernel matrix approaches the identity matrix (i.e., the kernel matrix of the identity kernel). It is difficult to glean any economic motivation for this kernel, but its success with classification and regression suggests it should be worth trying for market clearing as well.

4 Formulation

In this section, we formulate the market clearing problem as a quadratic program. To do this, we need to work with continuous approximations of the value and cost functions. Let $r = |X|$. Let $Y = \phi(X)$, where ϕ is the mapping under consideration, and let $y_i = \phi(x_i)$ for each $x_i \in X$. With a slight abuse of notation, the value function v can be written as a function over Y rather than X (assuming ϕ is one-to-one, but this can be dispensed with in the developments that follow). Let $\bar{Y}$ be the convex closure of Y, namely the set of all convex combinations of elements in Y. We introduce a function $\bar{v}$ over $\bar{Y}$, parametrized by $\lambda \geq 0$, with $\bar{v}(y)$ for $y \in \bar{Y}$ defined as

$$\max_{\substack{\sum_{i=1}^r \alpha_i = 1 \\ \alpha \in \mathbf{R}_+^r}} \left\{ \sum_{i=1}^r \alpha_i v(y_i) - \frac{\lambda}{2} \|\alpha\|^2 \;\middle|\; \sum_{i=1}^r \alpha_i y_i = y \right\} \quad (1)$$

We will refer to $\bar{v}$ as the buyer's *extended* value function. When $\lambda = 0$, $\bar{v}$ is the *concave extension* of v over $\bar{Y}$: the smallest concave function such that $\bar{v}(y_i) \geq v(y_i)$ for each $y_i \in Y$. When λ is large, the second term in the objective dominates. It is maximized when α is uniform: $\alpha_i = 1/k$ for each i. Thus it induces the function to consider more points in the neighborhood of y when imputing a value. The extended cost function $\bar{c}$ of c is defined similarly: above, $\min$ replaces $\max$, $c(y_i)$ replaces $v(y_i)$, and the second term in the objective is added rather than subtracted. When $\lambda = 0$, $\bar{c}$ is the *convex extension* of c over $\bar{Y}$.

Lemma 1 *The extended value (cost) function is concave (convex) for all $\lambda \geq 0$.*

We stress that for any given λ, even $\lambda = 0$, it is not necessarily the case that $\bar{v}(y_i) = v(y_i)$ for all $y_i \in Y$, and the same for $\bar{c}$ and c. The point of $\bar{v}$ and $\bar{c}$ is to approximate v and c by concave and convex functions over $\bar{Y}$, because we know that with such functions linear clearing prices exist in $\bar{Y}$.

We now formulate the clearing problem with respect to the extended value and cost functions. Explicitly, our quadratic program only tries to identify an efficient bundle, but we will see that the dual of the program gives the desired clearing prices. Note that (2) below corresponds to M different constraints.

$$\max_{\alpha,\beta \geq 0} \quad \sum_{i=1}^r \alpha_i v(x_i) - \sum_{i=1}^r \beta_i c(x_i) - \frac{\lambda}{2}\|\alpha\|^2 - \frac{\lambda}{2}\|\beta\|^2$$

$$\text{s.t.} \quad \sum_{i=1}^r \alpha_i \phi(x_i) = \sum_{i=1}^r \beta_i \phi(x_i) \quad (2)$$

$$\sum_{i=1}^r \alpha_i = 1 \;,\; \sum_{i=1}^r \beta_i = 1 \quad (3)$$

Observe that this quadratic program solves the problem of finding an efficient bundle with respect to $\bar{v}$ and $\bar{c}$, except that we have written $v(x_i)$ instead of $v(y_i)$ and $\phi(x_i)$ explicitly instead of y_i. The dual of this program is as follows.

$$\min_{\pi^s, \pi^b, p, \epsilon} \quad \pi^b + \pi^s + \tfrac{\mu}{2}\left\|\epsilon^b\right\|^2 + \tfrac{\mu}{2}\left\|\epsilon^s\right\|^2$$

$$\text{s.t.} \quad \pi^b \geq v(x_i) - \langle p, \phi(x_i)\rangle - \epsilon_i^b \quad i = 1, \ldots, r$$

$$\pi^s \geq \langle p, \phi(x_i)\rangle - c(x_i) - \epsilon_i^s \quad i = 1, \ldots, r$$

Here $\mu = 1/\lambda$. At an optimal solution, we have $\pi^b = \max_i \left\{ v(x_i) - \langle p, \phi(x_i)\rangle - \epsilon_i^b \right\}$. Thus π^b reflects the maximum utility that the buyer can derive from any bundle in X at prices p, to within a certain slack. The variable π^s has an analogous interpretation in terms of profit.

Ideally, the program would find a discrete solution, meaning that the coefficients α and β would generate a $y_i \in Y$, thus identifying a specific bundle $x_i \in X$. However, it is possible (in fact typical) that the optimal convex combinations do not result in a discrete solution. The following proposition gives a way to extract an (approximately) efficient bundle from a solution to the primal. Its proof consists of a straightforward appeal to complementary slackness.

Proposition 1 *Let (α, β) and $(\pi^b, \pi^s, p, \epsilon)$ be optimal primal and dual solutions. Assume there is an index i such that $\alpha_i > 0$ and $\beta_i > 0$. Then, letting $\delta^b = \max_j \epsilon_j^b - \epsilon_i^b$, $\delta^s = \max_j \epsilon_j^s - \epsilon_i^s$, and $\delta = \delta^b + \delta^s$, we have that*

(a) Bundle x_i is efficient to within an additive error of δ.

(b) Bundle x_i maximizes the buyer's utility (seller's profit) at prices p to within an additive error of δ^b (δ^s).

This result also clarifies the purpose of the parameter λ. Suppose that we set $\lambda = 0$ (equivalently, $\mu = \infty$) and solve the quadratic program, but that at the solution (α^*, β^*) there is no index i such that $\alpha_i^* > 0$ and $\beta_i^* > 0$. This indicates that clearing prices have not been found, and necessarily occurs if they do not exist for our choice of kernel. In this case we can increase λ—equivalently, decrease μ—to allow for approximate clearing prices in the dual. As $\lambda \to \infty$ the optimal solution tends to $\alpha_i^* = \beta_i^* = 1/k$ for all i, and thus we will reach a point where the conditions of Proposition 1 are satisfied. Therefore, increasing λ increases the chances that the quadratic program will identify a discrete solution for any given kernel function, with the caveat that the discrete solution might only be approximately efficient; correspondingly, the prices obtained in the dual will only approximately clear the market.

5 Algorithms

As formulated, our quadratic program is problematic in two respects. First, it has too many constraints. Recall that M, the dimension of the range of ϕ, is potentially very large because clearing prices may only exist in high-dimensional space. The number of variables is also too large: there is a variable α_i and a variable β_i for each $x_i \in X$, and X is of size 2^m. We deal with the first concern by using a kernel method, and then propose an elicitation procedure to address the second concern.

5.1 Method of Multipliers

The usual approach in kernel methods is to formulate the given problem as a mathematical program that only uses inner-product information (i.e., the kernel matrix), and then solve the optimization problem using general or special purpose algorithms. Our quadratic program is not formulated in terms of inner products. Instead, it is the procedure we use to solve the program that only uses information in the kernel matrix.

The procedure is known as the *method of multipliers*. (For an extensive treatment of this method, see [Bertsekas, 1996].) Despite its name, it was not originally conceived as a kernel method, but simply as a way to solve equality-constrained quadratic programs. That it only requires kernel matrix information when applied to our quadratic program was a crucial insight in this work. The procedure eliminates the problematic constraints (2) and replaces them with multiplier and penalty terms in the objective, which becomes

$$\max_{\alpha,\beta\geq 0} \quad \sum_{i=1}^{r} \alpha_i v(x_i) - \sum_{i=1}^{r} \beta c(x_i) - \frac{\lambda}{2}\left\|\alpha\right\|^2 - \frac{\lambda}{2}\left\|\beta\right\|^2$$

$$- \left\langle p, \sum_{i=1}^{r} \alpha_i \phi(x_i) - \sum_{i=1}^{r} \beta_i \phi(x_i) \right\rangle \qquad (4)$$

$$- \frac{\nu}{2} \left\| \sum_{i=1}^{r} \alpha_i \phi(x_i) - \sum_{i=1}^{r} \beta_i \phi(x_i) \right\|^2 \qquad (5)$$

Let us ignore the multiplier term (4) for the moment. Evaluating the penalty term, one finds that it only involves the kernel matrix K. As $\nu \to \infty$, the penalty term ensures that at an optimal solution, the constraints (2) are satisfied. In practice, one chooses a sufficiently large ν so that the constraints are satisfied to within an acceptable tolerance. There is a drawback, however: as ν grows large the problem becomes increasingly ill-conditioned, making it inherently more difficult to solve with standard quadratic programming algorithms. Thus one typically starts with a small initial ν^1 and increases it over several iterations until the constraints are sufficiently satisfied, in an attempt to avoid ill-conditioning. A typical update rule is $\nu^{k+1} = \tau\nu^k$ for some $\tau \in [4, 10]$, and this is the rule we used in our experiments [Bertsekas, 1996].

To speed up convergence and further alleviate ill-conditioning, one can introduce a multiplier term as in (4). For simplicity, let $h(\alpha, \beta) = \sum_{i=1}^{r} \alpha_i \phi(x_i) - \sum_{i=1}^{r} \beta_i \phi(x_i)$. The original method of multipliers (there are several variants) uses at each iteration k the update rule $p^{k+1} = p^k + \nu^k h(\alpha^k, \beta^k)$, where (α^k, β^k) is the optimal solution at the iteration. Under this update rule, the iterates p^k converge to the optimal dual solution [Bertsekas, 1996]—in this case the clearing prices, which is why we used the suggestive notation p for the multiplier. In theory and practice, using a multiplier speeds up convergence. This was borne out in our experiments. The multiplier is also essential for us because we seek to compute clearing prices, not just an efficient bundle.

If we use $p^1 = \mathbf{0}$ as the initial multiplier, then $p^{k+1} = \sum_{\ell=1}^{k} \nu^\ell h(\alpha^\ell, \beta^\ell)$, and substituting this into (4) yields a term that uses only inner-products. Thus the method of multipliers leads to a kernel method when applied to our quadratic program. At iteration k, the objective of our quadratic program becomes

$$\max_{\alpha^k,\beta^k\geq 0} \quad \sum_{i=1}^{r} \alpha_i^k v(x_i) - \sum_{i=1}^{r} \beta_i^k c(x_i) - \frac{\lambda}{2}\left\|\alpha^k\right\|^2 - \frac{\lambda}{2}\left\|\beta^k\right\|^2$$

$$-(\bar{\alpha}^k - \bar{\beta}^k)' K(\alpha^k - \beta^k)$$

$$-\frac{\nu}{2}(\alpha^k - \beta^k)' K(\alpha^k - \beta^k)$$

where $\bar{\alpha}^k = \sum_{\ell=1}^{k-1} \nu^\ell \alpha_i^\ell$ and $\bar{\beta}^k = \sum_{\ell=1}^{k-1} \nu^\ell \beta_i^\ell$. The constraints remain (3). One finds that the Hessian of the objective is negative-semidefinite from the fact that K is positive semidefinite, so we now have a straightforward quadratic maximization problem with concave objective function, for which there is a wide selection of commercial and non-commercial solvers.

At the final round k', we use the multiplier as the clearing prices given the method of multipliers' convergence properties. Assuming clearing prices have been found (i.e., the conditions of Proposition 1 hold), the price of a bundle $x \in X$ is given by

$$\langle p^{k'}, x\rangle = \sum_{i=1}^{r} (\bar{\alpha}_i^{k'} - \bar{\beta}_i^{k'}) \kappa(x_i, x). \qquad (6)$$

If the vectors $\bar{\alpha}^{k'}$ and $\bar{\beta}^{k'}$ are sparse, this gives us a sparse representation of nonlinear clearing prices.

5.2 Elicitation Procedure

The kernel method of the previous section addresses the question of computing nonlinear clearing prices given a kernel and full information of the value and cost functions, namely $v(x_i)$ and $c(x_i)$ for all $x_i \in X$. If, instead of this full information, we only know the values and costs of bundles in some limited sample S, it is still possible apply the algorithm of the previous section with respect to S rather than X. To check whether the resulting bundle y is efficient, and whether the resulting prices p are clearing, we can perform demand and supply queries with these as inputs. If y is returned in both cases, then it maximizes both the buyer's utility and seller's profit at prices p, and we are done, following the arguments of Section 2. Otherwise, we can grow our sample S by including the buyer and seller's replies.

The elicitation procedure based on these ideas is given formally as Algorithm 1. It begins with an empty sample. Throughout, prices are represented by the multiplier $(\bar{\alpha}, \bar{\beta})$, which is zero in the first round, leading to zero prices. The buyer's demand query reply at each round is recorded in x^b, initialized to $\mathbf{1}$, the bundle containing all items; the seller's reply at each round is x^s, initialized to $\mathbf{0}$, the empty bundle. The bundle y refers to the optimal bundle at each round, and δ^b and δ^s are the slacks at each round.

The elicitation proceeds as long as no bundle common to the demand and supply sets is found. At each round, the sample is updated with the latest replies. We write value(x^b) rather than $v(x^b)$ to indicate that an explicit value query is performed, and similarly for cost queries; when we write

Input: κ, λ, query access to v and c.
Output: whether clearing prices were found.
$S = \emptyset$, $p = (\mathbf{0}, \mathbf{0})$, $y = \mathbf{0}$;
$x^b = \mathbf{1}$, $x^s = \mathbf{0}$;
$\delta^b = 0$, $\delta^s = 0$;
while $x^b \neq y$ *or* $x^s \neq y$ **do**
  $S \leftarrow S \cup \{x^b, x^s\}$;
  record value(x^b), cost(x^b), value(x^s), cost(x^s);
  $(\alpha, \beta, \bar{\alpha}, \bar{\beta}) \leftarrow$ multipliers-method(S, κ, λ);
  if *there is an index* i *where* $\alpha_i > 0$ *and* $\beta_i > 0$ **then**
    $y \leftarrow x_i$;
    $p \leftarrow (\bar{\alpha}, \bar{\beta})$;
    $\delta^b = \max_{x_j \in S}[v(x_j) - p(x_j)] - [v(x_i) - p(x_i)]$;
    $\delta^s = \max_{x_j \in S}[p(x_j) - c(x_j)] - [p(x_i) - c(x_i)]$;
  else
    return `false`;
  end
  $x^b =$ demand(y, p, κ, δ^b);
  $x^s =$ supply(y, p, κ, δ^s);
end
return `true`;

Algorithm 1: Elicitation procedure to compute an efficient bundle and sparse clearing prices.

$v(x_i)$ or $c(x_i)$, it is because the algorithm has already obtained the value or cost of bundle x_i.

At each round, the method of multipliers is run with respect to S to find a candidate efficient bundle and clearing prices using the kernel κ. Specifically, the method returns the primal solution (α, β), from which we can deduce whether an efficient bundle was found, and the final multiplier $(\bar{\alpha}, \bar{\beta})$, which represents the prices according to (6). If the criterion of Proposition 1 does not hold at this point, then the procedure halts with a failure flag.

In the demand and supply queries, the kernel κ is passed along with the coefficients $p = (\bar{\alpha}, \bar{\beta})$ because it is needed to evaluate the price of bundles—recall (6). To ensure that the procedure makes progress at each round, we must have $x^b \notin S$ or $x^s \notin S$ for at least one of the replies. This is the essence of the following result. (Below, $\delta = \delta^b + \delta^s$ as before.)

Proposition 2 *Algorithm 1 converges in a finite number of rounds. Upon convergence, y is a δ-optimal bundle, and it maximizes the buyer's utility (seller's profit) at prices p to within an additive error of δ^b (δ^s).*

If $v = c$, then observe that the only possible clearing prices are $p = v = c$. In this case, any procedure that finds clearing prices must elicit the complete information of one agent. In the worst-case, this requires an exponential number of queries (in m), so there is no hope for our procedure or any other to always run in polynomial-time. (For formal communication complexity results to this effect, see [Nisan and Segal, 2006].) Nevertheless, none of this discounts the possibility that our method could perform well in practice, especially given that the modularity of the method allows one to choose a kernel function appropriate for the clearing task at hand. Therefore, we now turn to an empirical evaluation.

6 Empirical Evaluation

In this section we report on experiments run to evaluate the clearing, elicitation, and efficiency performance of our method. We used the CATS suite of distributions to generate value and cost functions for the buyer and seller [Leyton-Brown *et al.*, 2000]. CATS generates instances in the XOR language [Nisan, 2000]. An XOR representation of a value function consists of a subset of bundles $Z \subseteq X$ together with their values, in the form of *atomic bids* $(z, v(z))$ for all $z \in Z$. The value of a bundle $x \in X$ according to an XOR representation is $v(x) = \max_{\{z \leq x | z \in Z\}} v(z)$. For cost functions, we used the semantics $c(x) = \min_{\{z \geq x | z \in Z\}} c(z)$ to interpret the XOR instance. CATS allows one to specify the number of atomic bids desired in an XOR representation, as well as the number of items; in all our experiments, we used $m = 30$.

CATS was originally designed to generate valuation functions for the purpose of testing winner-determination algorithms for combinatorial auctions, not to generate pairs of value and cost functions, so ours is an unconventional use of the test suite. Nevertheless, this scheme generated nontrivial test instances. In fact, the relative difficulty of clearing the market under different CATS distributions (in terms of runtime) agreed with the relative difficulty of the distributions for winner determination in combinatorial auctions, as reported by Leyton-Brown and Shoham [2006].

Algorithm 1 was coded in C. To solve the quadratic programs within the method of multipliers, we used a noncommercial C implementation of the LOQO interior-point code [Vanderbei, 1999].[1] We used default parameters for LOQO, except for a margin of 0.25 rather than 0.95 (this amounts to a more aggressive stepsize). We used $\tau = 6$ for the updates in the method of multipliers.

CATS provides five different distributions: arbitrary, paths, matching, regions, and scheduling. The matching distribution generated exceedingly easy instances where linear clearing prices almost always existed, so we do not report on this distribution.

Clearing. To evaluate the clearing ability of our method with different kernels, we ran it on each of the four CATS distributions, varying the total number of atomic bids. Letting Z^b and Z^s be the bundles in the XOR representations of the value and cost functions, the total number of bids refers to $|Z^b| + |Z^s|$. The average number of instances cleared are reported in Table 1. As expected, the identity kernel cleared 100% of the instances for all distributions. The fact that the linear kernel regularly failed to clear the market indicates that the CATS distributions often generate interesting instances for which linear clearing prices do not exist. For the paths and scheduling distributions, the all-subsets kernel generated a kernel matrix with such large entries that the penalty term (5) swamped the linear terms in the objective, leading LOQO to ignore the latter and reach suboptimal solutions. This revealed an unanticipated limitation of the all-subsets kernel and of our method. We believe this could be remedied to an extent with proper scaling and more conservative LOQO settings.

[1] The implementation, due to Alex Smola, was retrieved at `www.kernel-machines.org/code/prloqo.tar.gz`

		arbitrary		**paths**		**regions**		**scheduling**	
κ	bids	clr.	*elc.*	clr.	*elc.*	clr.	*elc.*	clr.	*elc.*
linear	100	61	*16*	8	*32*	61	*17*	18	*15*
	300	17	*6*	6	*14*	33	*6*	16	*8*
	500	5	*4*	6	*10*	38	*4*	20	*9*
Gaussian	100	100	*25*	59	*45*	95	*24*	61	*31*
	300	89	*18*	66	*26*	78	*13*	43	*19*
	500	78	*15*	79	*22*	73	*8*	50	*15*
all-subsets	100	100	*33*	n/a	*n/a*	100	*39*	n/a	*n/a*
	300	100	*43*	n/a	*n/a*	100	*28*	n/a	*n/a*
	500	100	*43*	n/a	*n/a*	100	*21*	n/a	*n/a*
identity	100	100	*43*	100	*32*	100	*45*	100	*46*
	300	100	*44*	100	*22*	100	*41*	100	*43*
	500	100	*44*	100	*20*	100	*38*	100	*43*

Table 1: Average clearing and elicitation performance of our method over 200 instances using $\lambda = 0$ and $\sigma = 30$. Elicitation figures are averages over those instances that cleared.

Elicitation. The elicitation metric is defined as $|S|/|Z^b \cup Z^s|$, where S is the sample upon termination of Algorithm 1. The motivation for this definition is that at most $|Z^b \cup Z^s|$ bundles can be elicited when the value and cost functions are represented with XOR. Table1 shows that with all choices of kernels, the method is able to reach clearing prices with much less than full revelation on average. Importantly, elicitation often decreases as bids are scaled up; this is particularly the case for the Gaussian kernel. That the identity kernel is able to clear the market with less than 50% elicitation on average was an unexpected but welcome finding. The reason this occurs is that the kernel explicitly prices the empty bundle; this amounts to adding a constant term to the price of every other bundle, and this can lead to a much sparser way of representing clearing prices.

Sparsity. The sparsity metric we use is the number of nonzero coefficients in the final prices (6) divided by the term $|Z^b \cup Z^s|$, which is the maximum possible price size. By definition, sparsity is never more than elicitation, because prices contain one coefficient for each element of S, and some coefficients may be zero. We do not report on any sparsity statistics because we found that sparsity was always extremely close to elicitation. This indicates that the method elicits very few superfluous bundles for the purpose of clearing.

Approximation. As explained, the λ parameter can be adjusted to increase the chances that clearing prices are found, at the expense of approximate rather than exact clearing. Figure 1 illustrates its effect on the paths distribution. We see that with a modest efficiency sacrifice of around 3%, the Gaussian kernel can be made to clear almost all instances, where it was able to clear less than 70% when exact clearing was imposed. The improvement is even more dramatic for the linear kernel: a clearing improvement of almost 75% with less than a 5% sacrifice in efficiency. The trade-off was comparable on the other distributions.

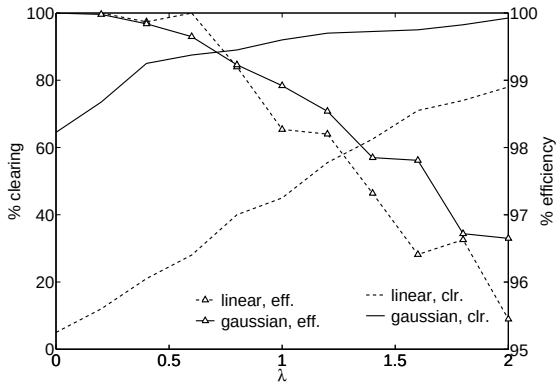

Figure 1: Clearing and efficiency performance of the linear and Gaussian kernels ($\sigma = 30$) on the paths distribution with 250 bids, varying the approximation parameter λ. Each data point is averaged over 200 instances.

References

[Bertsekas, 1996] Dimitri P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Athena Scientific, 1996.

[Chapelle and Harchaoui, 2005] Olivier Chapelle and Zaïd Harchaoui. A machine learning approach to conjoint analysis. In *Advances in Neural Information Processing Systems*, 17. MIT Press, 2005.

[Domshlak and Joachims, 2006] Carmel Domshlak and Thorsten Joachims. Unstructuring user preferences: Efficient non-parametric utility revelation. In *Proc. of the 21st Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 169–177, 2006.

[Evgeniou *et al.*, 2005] Theodoros Evgeniou, Constantinos Boussios, and Giorgos Zacharia. Generalized robust conjoint estimation. *Marketing Science*, 24(3):415–429, 2005.

[Lahaie and Parkes, 2004] Sébastien Lahaie and David C. Parkes. Applying learning algorithms to preference elicitation. In *Proc. of the 5th ACM Conference on Electronic Commerce (EC)*, pages 180–188, 2004.

[Leyton-Brown and Shoham, 2006] Kevin Leyton-Brown and Yoav Shoham. A test suite for combinatorial auctions. In *Combinatorial Auctions*, chapter 18, pages 451–478. MIT Press, 2006.

[Leyton-Brown *et al.*, 2000] Kevin Leyton-Brown, Mark Pearson, and Yoav Shoham. Towards a universal test suite for combinatorial auction algorithms. In *Proc. of the second ACM Conference on Electronic Commerce (EC)*, pages 66–76, 2000.

[Nisan and Segal, 2006] Noam Nisan and Ilya Segal. The communication requirements of efficient allocations and supporting prices. *Journal of Economic Theory*, 129(1):192–224, 2006.

[Nisan, 2000] Noam Nisan. Bidding and allocation in combinatorial auctions. In *Proc. second ACM Conference on Electronic Commerce (EC)*, pages 1–12, 2000.

[Shawe-Taylor and Cristianini, 2004] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[Vanderbei, 1999] Robert Vanderbei. LOQO user's manual–version 3.10. *Optimization Methods and Software*, 12:485–514, 1999.