Computational Optimization and Applications, , 1–11 (2004)

Convex Kernel Underestimation of Functions with Multiple Local Minima

O. L. MANGASARIAN olvi@cs.wisc.edu
Computer Sciences Department
University of Wisconsin
Madison, WI 53706
Department of Mathematics
University of California at San Diego
La Jolla, CA 92093

J. B. ROSEN jbrosen@cs.ucsd.edu
Department of Computer Science and Engineering
University of California at San Diego
La Jolla, CA 92093

M. E. THOMPSON thompson@cs.wisc.edu
Computer Sciences Department
University of Wisconsin
Madison, WI 53706

Received June 21, 2004; Revised March 14, 2005

Editor:

Abstract. A function on R^n with multiple local minima is approximated from below, via linear programming, by a linear combination of convex kernel functions using sample points from the given function. The resulting convex kernel underestimator is then minimized, using either a linear equation solver for a linear-quadratic kernel or by a Newton method for a Gaussian kernel, to obtain an approximation to a global minimum of the original function. Successive shrinking of the original search region to which this procedure is applied leads to fairly accurate estimates, within 0.0001% for a Gaussian kernel function, relative to global minima of synthetic nonconvex piecewise-quadratic functions for which the global minima are known exactly. Gaussian kernel underestimation improves by a factor of ten the relative error obtained using a piecewise-linear underestimator [11], while cutting computational time by an average factor of over 28.

keywords: *multiple minima, underestimation, convex kernels, global minimization*

1. Introduction

Recently [11] nonconvex functions with multiple local minima were underestimated by a convex piecewise-linear function in order to obtain a global minimum of the function. Aside from taking over four hours for solving a 6-dimensional problem, the method required the solution of an NP-hard problem of minimizing a concave function on a polyhedral set. In an earlier work [14] the same problem was solved by obtaining a strongly convex quadratic underestimator. However that approach also required the difficult minimization of a concave function on a quadratically constrained convex region.

The present approach avoids all the above difficulties associated with these two methods by using an underestimator that is a linear combination of convex kernel functions. The convex kernel underestimator is obtained by solving a single linear program. A global minimum of the underestimator is then easily obtained by either solving a system of linear equations or by a Newton method. Kernel (or equivalently support vector machine) methods used here constitute the method of choice for classification and approximation problems [16, 3, 18, 8, 1, 6, 10, 9, 15]. However, these methods have not been used to any significant extent for nonconvex function minimization. The nonconvex function minimization problem that we are interested in here plays a key role in protein docking [5, 12, 13] and will be used to test our method.

We briefly outline the contents of the paper now. In Section 2 we formulate the kernel underestimation problem as a linear program for both a linear-quadratic kernel and a Gaussian kernel and give two algorithms for finding a global minimum of an underestimator. In Section 3 we present our numerical results and compare them with previous ones. The global minimum value was attained to within 0.9290% relative to the exact minimum by using a linear-quadratic kernel underestimator and to within 0.0001% by using a Gaussian kernel underestimator. Section 4 concludes the paper.

A word about our notation and background material. All vectors will be column vectors unless transposed to a row vector by a prime superscript $'$. The scalar product of two vectors x and y in the n-dimensional real space R^n will be denoted by $x'y$. For $x \in R^n$ the norm $\|x\|$ will denote the 2-norm and $\|x\|_1$ will denote the 1-norm. For an $m \times n$ matrix A, A_i will denote the ith row of A, $A_{\cdot j}$ will denote the jth column of A and A_{ij} will denote the element in row i and column j. The identity matrix in a real space of arbitrary dimension will be denoted by I, while a column vector of ones of arbitrary dimension will be denoted by e. For a vector $v \in R^m$, $diag(v)$ will denote an $m \times m$ diagonal matrix with v along the diagonal. For $A \in R^{m\times n}$ and $B \in R^{n\times k}$, a *kernel* $K(A, B)$ is an arbitrary function that maps $R^{m\times n} \times R^{n\times k}$ into $R^{m\times k}$. In particular, if x and y are column vectors in R^n then, $K(x', y)$ is a real number, $K(x', A')$ is a row vector in R^m and $K(A, A')$ is an $m \times m$ matrix. The base of the natural logarithm will be denoted by ε. A frequently used kernel in nonlinear classification is the Gaussian kernel [18, 8] whose ijth element, $i = 1 \dots, m,\ j = 1 \dots, k$, is given by: $(K(A, B))_{ij} = \varepsilon^{-\mu\|A_i{}' - B_{\cdot j}\|^2}$, where $A \in R^{m\times n}$, $B \in R^{n\times k}$ and μ is a positive constant. We shall assume for simplicity that our kernels are symmetric, that is $K(x', y)' = K(y', x)$, and instead of the standard Mercer positive semidefiniteness condition on $K(A, A')$ [18, 16] we shall require that the m components of $K(x', A')$ are convex functions on R^n, which turns out to be the case for linear and quadratic kernels as well as for the *negative* of the Gaussian kernel. We shall refer to such kernels as *convex kernels*.

2. Convex Kernel Underestimation via Linear Programming

The problem we are interested in is to find a global minimum of a function $f : R^n \longrightarrow R$, given m function evaluations of $f(x)$, that is:

$$y^k = f(x^k),\ k = 1, \ldots, m. \tag{1}$$

In [14] a strictly convex quadratic underestimator:

$$q(\alpha, c, M; x) = \alpha + c'x + \frac{1}{2}x'Mx,\ M \text{ symmetric positive definite}, \tag{2}$$

is first obtained by solving the mathematical program:

$$\begin{array}{ll} \displaystyle\min_{\alpha,c,M} & \displaystyle\sum_{k=1}^{m}(y^k - q(\alpha, c, M; x^k)) \\ \text{s.t.} & \begin{array}{rcl} q(\alpha, c, M; x^k) & \leq & y^k,\ k = 1, \ldots, m, \\ M & & \text{symmetric positive definite}, \end{array} \end{array} \tag{3}$$

where $\alpha \in R$, $c \in R^n$ and $M \in R^{n\times n}$, and then minimizing $q(\alpha, c, M; x)$ over $x \in R^n$. The nonlinear positive definiteness constraint in (3) complicates this otherwise linear formulation. This constraint is converted to a convex quadratic constraint in [14] by using the representation $M = LL'$, which renders the objective of (3) a concave function that is difficult to find a global minimum for. Another possible difficulty with this approach is that a *single* strongly convex quadratic function, such as $q(\alpha, c, M; x)$, might not closely underestimate $f(x)$. To avoid this last difficulty, a piecewise-linear underestimator was proposed in [11]:

$$p(\alpha, c, A, b; x) = \alpha + c'x + \|Ax + b\|_1, \tag{4}$$

where $\alpha \in R$, $c \in R^n$, $A \in R^{\ell\times n}$ and $b \in R^{\ell}$, where ℓ is the number of linear functions generating our piecewise-linear underestimation. Note that $p(\alpha, c, A, b; x)$ is convex and piecewise-linear in x for fixed (α, c, A, b) and similarly it is convex and piecewise-linear in (α, c, A, b) for fixed x. The approximation problem for an underestimator of $f(x)$ proposed in [11] was the following concave minimization problem:

$$\begin{array}{ll} \displaystyle\min_{\alpha,c,A,b} & \displaystyle\sum_{k=1}^{m}(y^k - p(\alpha, c, A, b; x^k)) \\ \text{s.t.} & p(\alpha, c, A, b; x^k) \leq y^k,\ k = 1, \ldots, m. \end{array} \tag{5}$$

Note that unlike the mathematical program (3), there are no constraints on the matrix A. However, the objective function of this minimization problem is again concave and at best only a local minimum can be found.

It is these noted difficulties that led us to the following considerably simpler formulation that utilizes a convex kernel underestimator and requires merely a linear program for its generation. Our underestimator is the following:

$$s(a, \beta, \gamma; x) = K(x', H')a + \beta + \gamma\frac{x'x}{2} = \sum_{i=1}^{m} a_i K(x', H_i') + \beta + \gamma\frac{x'x}{2}, \tag{6}$$

where $a \in R^m$, $\beta \in R$, $x \in R^n$, $H \in R^{m\times n}$ is the matrix whose m rows consist of the m row vectors: $x^{1'}, x^{2'}, \ldots, x^{m'}$, $K(x', H')$ is a convex kernel function: $R^{1\times n} \times R^{n\times m} \longrightarrow R^{1\times m}$, as defined in the Introduction and $\gamma \in R$. A positive value for the parameter γ ensures positive definiteness of the Hessian of the underestimator for linear and quadratic kernel functions. The parameters a, β and γ defining our kernel underestimator are chosen by the following linear program:

$$\begin{array}{ll} \min\limits_{a\geq 0,\beta,\gamma\geq 0} & \sum\limits_{k=1}^{m}(y^k - s(a,\beta,\gamma;x^k)) \\ \text{s.t.} & \qquad s(a,\beta,\gamma;x^k) \ \leq\ y^k,\ k=1,\ldots,m. \end{array} \tag{7}$$

In order to ensure that the resulting underestimator is convex in x, we look at the gradient and Hessian of $s(a,\beta,\gamma;x)$ with respect to x, which are:

$$\nabla_x s(a,\beta,\gamma;x) = \sum_{i=1}^{m} a_i \nabla_x K(x', H_i{}') + \gamma x, \tag{8}$$

$$\nabla_{xx} s(a,\beta,\gamma;x) = \sum_{i=1}^{m} a_i \nabla_{xx} K(x', H_i{}') + \gamma I. \tag{9}$$

For the *linear-quadratic kernel function*:

$$lq(a,\beta,\gamma;x) = \sum_{i=1}^{m} [a_i^{(1)}(x'H_i{}') + a_i^{(2)}(x'H_i{}')^2] + \beta + \gamma\frac{x'x}{2},\ a = \begin{bmatrix} a^{(1)} \in R^m \\ a^{(2)} \in R^m \end{bmatrix}, \tag{10}$$

the gradient and Hessian are:

$$\nabla_x lq(a,\beta,\gamma;x) = \sum_{i=1}^{m} a_i^{(1)} H_i{}' + \sum_{i=1}^{m} 2a_i^{(2)}(x'H_i{}')H_i{}' + \gamma x, \tag{11}$$

$$\nabla_{xx} lq(a,\beta,\gamma;x) = \sum_{i=1}^{m} 2a_i^{(2)} H_i{}'H_i + \gamma I. \tag{12}$$

It follows immediately from (12) that the Hessian $\nabla_{xx} lq(a,\beta,\gamma;x)$ is positive semidefinite if $a_i^{(2)}$ and γ are nonnegative. Hence the linear-quadratic kernel function $lq(a,\beta,\gamma;x)$ of (10) is convex on R^n under the constraints that $a_i^{(2)}$ and γ are nonnegative which are already imposed by the linear program (7), and is strictly convex if γ is bounded below by a positive number.

For the *Gaussian kernel function*:

$$g(a,\beta,\gamma;x) = \sum_{i=1}^{m} -\ a_i \varepsilon^{-\mu\|x-H_i{}'\|^2} + \beta + \gamma\frac{x'x}{2}, \tag{13}$$

the gradient and Hessian are:

$$\nabla_x g(a,\beta,\gamma;x) = \sum_{i=1}^{m} 2a_i \varepsilon^{-\mu\|x-H_i{}'\|^2} \mu(x - H_i{}') + \gamma x, \tag{14}$$

$$\nabla_{xx}g(a,\beta,\gamma;x) = \sum_{i=1}^{m} 2\mu a_i \varepsilon^{-\mu\|x-H_i{'}\|^2}[I - 2\mu(x-H_i{'})(x-H_i{'})'] + \gamma I. \qquad (15)$$

It follows from (15) that the Hessian $\nabla_{xx}g(a,\beta,\gamma;x)$ is positive definite if μ is sufficiently small and a_i and γ are nonnegative. Hence, the Gaussian kernel function $g(a,\beta,\gamma;x)$ of (13) is strictly convex on any bounded subset of R^n under the constraints that a_i and γ are nonnegative and μ is sufficiently small. Numerically μ ranged in the interval $[0.00001, 0.1]$ and was determined experimentally.

We conclude from the above that the linear program (7) will generate a *convex* underestimator for both the linear-quadratic kernel function $lq(a,\beta,\gamma;x)$ of (10) and for the Gaussian kernel function $g(a,\beta,\gamma;x)$ of (13).

We state now our explicit algorithms for finding the global minimum of our kernel function underestimator which is obtained by solving the linear program (7). Note that for the linear-quadratic kernel function the global minimum is found by solving a system of linear equations and for the Gaussian kernel function by using a Newton method.

Algorithm 1 Global Minimum of Convex Linear-Quadratic Kernel Underestimator *Solve the linear program (7) with $s(a,\beta,\gamma;x)$ replaced by $lq(a,\beta,\gamma;x)$ of (10). Find the global minimum of $lq(a,\beta,\gamma;x)$ with respect to x by solving the linear system of equations obtained by setting $\nabla_x lq(a,\beta,\gamma;x)$ of (11) equal to zero.*

Remark 2 *The system of linear equations $\nabla_x lq(a,\beta,\gamma;x) = 0$, to be solved in Algorithm 1 above, is nonsingular whenever the Hessian $\nabla_{xx} lq(a,\beta,\gamma;x)$ of (12) is positive definite. This is the case whenever $\gamma > 0$ or the n columns of the matrix $(diag(a^2))^{\frac{1}{2}}H$ are linearly independent. In our numerical experiments it turns out that the Hessian $\nabla_{xx} lq(a,\beta,\gamma;x)$ is nonsingular because $\gamma > 0$.*

Algorithm 3 Global Minimum of Convex Gaussian Kernel Underestimator *Solve the linear program (7) with $s(a,\beta,\gamma;x)$ replaced by $g(a,\beta,\gamma;x)$ of (13). Find the global minimum of $g(a,\beta,\gamma;x)$ with respect to x by a Newton method with an Armijo stepsize obtained by setting $\nabla_x g(a,\beta,\gamma;x)$ of (14) equal to zero.*

Remark 4 *The Newton direction for solving the nonlinear equations $\nabla_x g(a,\beta,\gamma;x) = 0$ in Algorithm 3 is a descent direction whenever the Hessian $\nabla_{xx} g(a,\beta,\gamma;x)$ of (15) is positive definite, which is the case whenever $\gamma \geq 0$ and μ is small enough. This together with an Armijo stepsize ensures global quadratic convergence of Algorithm 3 [7, Appendix].*

Remark 5 *Computationally, it was found that adding a Tikhonov regularization term [17] to the minimization problem (7), with weight $\frac{1}{\nu}$, improved computational efficiency. Such regularization, commonly used in machine learning and statistics, consists of adding a term that minimizes a norm of some of the variables to be determined. In our case we added the term that minimizes the 1-norm of (a,β) as follows:*

$$\begin{array}{ll}\displaystyle\min_{a\geq 0,\beta,\gamma\geq 0} & \displaystyle\sum_{k=1}^{m}(y^k - s(a,\beta,\gamma;x^k)) + \frac{1}{\nu}\left\|\begin{bmatrix} a\\ \beta\end{bmatrix}\right\|_1 \\ s.t. & \qquad\qquad s(a,\beta,\gamma;x^k) \;\leq\; y^k,\ k=1,\ldots,m.\end{array} \tag{16}$$

Values of ν used are given in Tables 1 and 2. Note that $\left\|\begin{bmatrix} a\\ \beta\end{bmatrix}\right\|_1$ is handled by replacing it by $e't = \sum_{i=1}^{m+1} t_i$ and the added linear constraints $-t \leq \begin{bmatrix} a\\ \beta\end{bmatrix} \leq t$, that is:

$$\begin{bmatrix} -t_1\\ \vdots\\ -t_{m+1}\end{bmatrix} \leq \begin{bmatrix} a_1\\ \vdots\\ a_m\\ \beta\end{bmatrix} \leq \begin{bmatrix} t_1\\ \vdots\\ t_{m+1}\end{bmatrix}.$$

We turn now to numerical implementation and testing of the proposed algorithms.

3. Numerical Testing

We tested both Algorithms 1 and 3 on six nonconvex piecewise-quadratic functions on R^n with $n = 1, \ldots, 6$ defined as follows.

$$y(x) = \min_{j\in\{1,\ldots,r\}} h_j(x), \tag{17}$$

where $h_j(x)$, $j = 1, \ldots, r$ are arbitrary strictly convex quadratic functions, such as:

$$h_j(x) = \beta^j + {d^j}'x + \frac{1}{2}x'(0.5I + {M^j}'M^j)x,\ j = 1,\ldots,r, \tag{18}$$

Here, $\beta^j \in R$, $d^j \in R^n$ and $M^j \in R^{n\times n}$ are randomly chosen. In our testing, we used $r = 5$. An interesting feature of the piecewise-quadratic function $y(x)$ generated as described above, is that its exact global minimum solution can be computed as follows [11, Section 4, Proposition 1]. An exact global minimum of (17)-(18) is given by:

$$\min_{j\in\{1,\ldots,r\}}\min_{x\in R^n} h_j(x). \tag{19}$$

More specifically,

$$\min_{x\in R^n} y(x) = \min_{j\in\{1,\ldots,r\}} h_j(x^j), \tag{20}$$

where:

$$x^j = -(0.5I + {M^j}'M^j)^{-1}d^j,\ j = 1,\ldots,r. \tag{21}$$

We also tested our algorithms on a synthetic protein docking (SPD) problem generated from real docking data [12, 14]. For the SPD problem we used the model (17), where each $h_j(x)$ is a strictly convex quadratic function with a pre-determined minimum solution corresponding to local minima of the docking problem energy function. The protein docking problem energy function [12] is defined on R^6 with $r = 5$. The data for the SPD consisted of five local minima v_j of the docking energy function given at five specific points $x^j \in R^6$, $j = 1, \ldots, 5$. The six dimensions of the space R^6 for the SPD represent three coordinate axes and three angles that define the docking energy function. For each of the five given local minima v_j at x^j, $j = 1, \ldots, 5$, we generated a quadratic $h_j(x)$ as in (18) with the matrix $(0.5I + {M^j}'M^j)$ replaced by a diagonal matrix $D^j \in R^{6\times 6}$. The minimum value of each $h_j(x)$ was set to be v_j at x^j as follows. We chose the diagonal matrix D^j with element values ranging between 0.6 and 140, then set $d^j = -D^j x^j$ and $\beta^j = v_j - \frac{1}{2}{d^j}'x^j$.

Results of these tests are presented in Table 1 for the linear-quadratic kernel function of (10) and in Table 2 for the Gaussian kernel function of (13).

In our testing, we used a search process similar to that in [11, Section 5], refining the search region around the minimum solution of the underestimator after each iteration. An example of the refinement process using Algorithm 3 is shown in Figure 1 with the synthetic piecewise-quadratic function of (17)-(18) on R with $r = 5$. In our implementation using Algorithm 1, we found it helpful to introduce a *positive* lower bound γ_{min} on γ in (16). This caused the solution of $lq(a, \beta, \gamma; x)$ to consistently be within the search region. Furthermore, in implementing Algorithm 3, we found that γ was not needed, because μ was sufficiently small so as to render $g(a, \beta, \gamma; x)$ strictly convex, and so we conducted those tests with $\gamma = 0$. We also used the solution of the previous iterate as a starting point for our Newton method when finding the minimum of $g(a, \beta, \gamma; x)$.

We make the following observations regarding our computational results:

(i) With the exception of the SPD problem, both the computed minimum value error as a percentage of the true minimum value (*%Error in min*) and the 1-norm error in the computed solution vector as a percentage of the 1-norm of the true solution vector (*%Error in soln*) were better for the linear-quadratic kernel underestimator using Algorithm 1 than the corresponding results of the piecewise-linear underestimator of [11, Section 5, Table 1]. Computational times (*Time*) were also faster for all cases including the SPD problem.

(ii) The Gaussian kernel results of Algorithm 3 were consistently better than the linear-quadratic kernel results of Algorithm 1. That is, all quantities: *%Error in min*, *%Error in soln* and *Time* were smaller for the Gaussian kernel. An intuitive justification for the superiority of the Gaussian kernel is that it mimics the commonly used and effective nearest neighbor [2, 4] classification approach of machine learning wherein points are classified according to the class of the majority of the nearest k neighboring points. In particular, here and in classification algorithms, the Gaussian kernel term $\varepsilon^{-\mu\|x-{H_i}'\|^2}$ gives much more weight to an H_i that is closer to a given x than an H_i that is much further away from x.

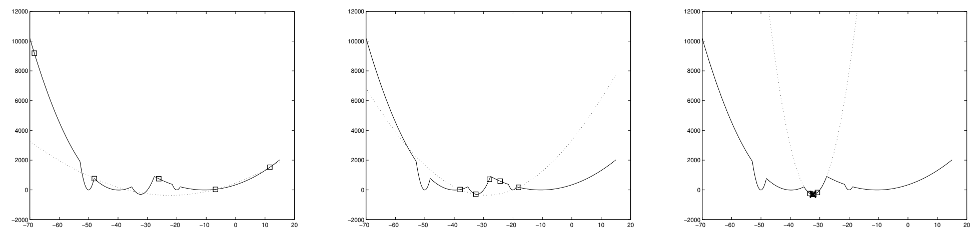
(iii) All *%Error in min* and *%Error in soln* were ten times smaller for the Gaussian kernel Algorithm 3 than those for the piecewise-linear underestimator of [11, Section 5, Table 1]. Average speedup of *Time* for the Gaussian kernel over those for the piecewise-linear underestimator was over 28.

(iv) We note that we conveniently used $m = 3^n$ sample points by taking 3 values along each coordinate axis. Because $n \leq 6$ is sufficiently small for the SPD and our other numerical problems, this poses no difficulty. To handle considerably larger dimensional problems one could use instead $m(<< 3^n)$ randomly generated points in R^n, say of the order of n^3.

(v) We give in Table 3 below a comparison of the proposed convex kernel underestimation versus the piecewise-linear underestimation of [11] on the SPD problem in R^6. The table shows the distinct superiority of the Gaussian kernel underestimator. We note that the results of the strictly convex quadratic underestimation of [14] are not directly comparable to the present results because the function being underestimated there is a perturbed strictly convex quadratic function and not a nonconvex piecewise-quadratic function such as (17).

In view of (ii), (iii) and (v) above, *the Gaussian kernel function is the underestimator of choice* for finding a global minimum of functions with multiple minima. This parallels the wide use of the Gaussian kernel in the extensive literature on kernel classification and approximation. However, quite unlike the existing kernel literature, we utilize here the *negative* of the Gaussian kernel in order to generate a convex kernel function underestimator.

As in [11], our computations were performed on machines utilizing an 800 Mhz Pentium III processor and 256MB of memory running on Redhat Linux 9, with MATLAB 6.5 installed.



Iteration 1 Iteration 3 Iteration 6

Figure 1. **Selected iterations from the refinement process using Algorithm 3 on a synthetic piecewise-quadratic function defined in (17)-(18) over R with $r = 5$.**

4. Conclusion and Outlook

We have proposed a method for finding an accurate estimate of a global minimum of a nonconvex function by underestimating the function by a linear combination of

Table 1. **Linear-quadratic kernel results on six synthetic test problems and the synthetic protein docking (SPD) problem. The number m of initial sample points used to generate our underestimator is 3^n. %Error is rounded to 4 decimal places.**

	Synthetic Problems in R^n						SPD in R^6
n	1	2	3	4	5	6	6
m	3	9	27	81	243	729	729
Refinement rate	0.5	0.5	0.5	0.5	0.5	0.25	0.25
Iterate tolerance	0.0001	0.01	0.05	0.005	0.1	0.01	0.01
ν	1e0	1e1	1e-1	1e0	1e-1	1e0	1e-2
γ_{min}	1e3	1e3	1e3	5e3	1e3	5e3	5e3
True min	-2917.9	-2758.5	-3594.0	-12421.2	-3217.6	-7023.7	-42.7
Computed min	-2917.9	-2758.5	-3594.2	-12421.4	-3221.5	-7025.1	-42.3
Error in min	0.0	0.0	-0.2	-0.2	-3.9	-1.3	0.4
%Error in min	0.0000%	-0.0004%	-0.0052%	-0.0018%	-0.1211%	-0.0187%	0.9290%
Error in soln (1-norm)	0.000	0.012	0.020	0.021	0.098	0.093	0.392
%Error in soln	0.0001%	0.0335%	0.0688%	0.0509%	0.2598%	0.2227%	0.7767%
No. refinements	27	14	15	38	41	17	12
Time (s)	0.37	0.54	2.07	23.94	372.08	1649.43	1042.33
Time per refinement	0.01	0.04	0.14	0.63	9.08	97.03	86.86

Table 2. **Gaussian kernel results on six synthetic test problems and the synthetic protein docking (SPD) problem. The number m of initial sample points used to generate our underestimator is 3^n. %Error is rounded to 4 decimal places.**

	Synthetic Problems in R^n						SPD in R^6
n	1	2	3	4	5	6	6
m	3	9	27	81	243	729	729
Refinement rate	0.5	0.5	0.25	0.5	0.5	0.25	0.5
Iterate tolerance	0.1	0.001	0.01	0.01	0.01	0.01	0.001
ν	1e-2	1e-3	1e2	1e0	1e0	1e-1	1e-3
Kernel parameter (μ)	1e-5	1e-1	1e-4	1e-4	1e-4	1e-3	1e-1
True min	-4734.1	-1675.8	-4576.2	-19824.8	-6286.7	-18431.4	-42.7
Computed min	-4734.1	-1675.8	-4576.2	-19824.8	-6286.7	-18431.4	-42.7
Error in min	0.0	0.0	0.0	0.0	0.0	0.0	0.0
%Error in min	0.0000%	0.0000%	-0.0001%	-0.0001%	0.0000%	0.0000%	0.0000%
Error in soln (1-norm)	0.000	0.002	0.003	0.009	0.011	0.004	0.004
%Error in soln	0.0000%	0.0069%	0.0071%	0.0153%	0.0209%	0.0103%	0.0085%
No. refinements	5	11	17	27	32	30	16
Time (s)	0.09	0.49	2.02	14.31	97.07	875.95	477.20
Time per refinement	0.02	0.04	0.12	0.53	3.03	29.20	29.82

convex kernel functions and then finding the global minimum of the underestimator. The method gives accurate estimates of global minima for a class of synthetic nonconvex piecewise-quadratic functions that closely model protein docking problems. An interesting problem for future consideration is that of approximating

Table 3. **Comparison of Convex Kernel and Piecewise-Linear Underestimation [11] on the SPD Problem in R^6.**

Underestimator	%Error in Soln (1-Norm)	%Error in Min	Time(Seconds)
Gaussian Kernel	0.0085%	0.0000%	477.20
Quadratic Kernel	0.7767%	0.9290%	1042.33
Piecewise-Linear	0.083%	0.014%	5846.9

nonconvex protein docking energy functions by our synthetic nonconvex piecewise-quadratic function (17)-(18). This may lead to a considerably more accurate global docking energy representation whose exact global minimum can be easily found by (20)-(21).

Acknowledgments

The research described in this Data Mining Institute Report 04-02, May 2004, was supported by National Science Foundation Grants CCR-0138308, ITR-0082146, by the Microsoft Corporation and by ExxonMobil.

References

1. V. Cherkassky and F. Mulier. *Learning from Data - Concepts, Theory and Methods*. John Wiley & Sons, New York, 1998.
2. S. Cost and S. Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10:57–58, 1993.
3. N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, MA, 2000.
4. L. Devroye and T. J. Wagner. Nearest neighbor methods in discrimination. In P. R. Krishnaiah and L. N. Kanal, editors, *Handbook of Statistics, Volume 2: Classification, Pattern Recognition and Reduction of Dimensionality*, Netherlands, 1982. North-Holland.
5. K. A. Dill, A. T. Phillips, and J. B. Rosen. CGU: An algorithm for molecular structure prediction. In L. T. Biegler et al, editor, *IMA Volumes in Mathematics and its Applications: Large Scale Optimization with Applications III: Molecular Structure and Optimization*, pages 1–22, 1997.
6. T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, New York, 2001.
7. Y.-J. Lee and O. L. Mangasarian. SSVM: A smooth support vector machine. *Computational Optimization and Applications*, 20:5–22, 2001. Data Mining Institute, University of Wisconsin, Technical Report 99-03. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/99-03.ps.
8. O. L. Mangasarian. Generalized support vector machines. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 135–146, Cambridge, MA, 2000. MIT Press. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-14.ps.
9. O. L. Mangasarian and D. R. Musicant. Robust linear and support vector regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(9):950–955, 2000. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/99-09.ps.

10. O. L. Mangasarian and D. R. Musicant. Large scale kernel regression via linear programming. *Machine Learning*, 46:255–269, 2002. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/99-02.ps.
11. O. L. Mangasarian, J. B. Rosen, and M. E. Thompson. Global minimization via piecewise-linear underestimation. Technical Report 03-03, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, June 2003. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/03-03.ps. *Journal of Global Optimization*, to appear.
12. J. C. Mitchell, A. T. Phillips, J. B. Rosen, and L. F. Ten Eyck. Coupled optimization in protein docking. In *Optimization in Computational Chemistry and Molecular Biology*, pages 191–207, Dordrecht, Netherlands, 2000. Kluwer Academic Publishers.
13. A. T. Phillips, J. B. Rosen, and K. A. Dill. Convex global underestimation for molecular stucture prediction. In P. M. Pardalos et al, editor, *From Local to Global Optimization*, pages 1–18, Dordrecht, Netherlands, 2001. Kluwer Academic Publishers.
14. J. B. Rosen and R. F. Marcia. Convex quadratic approximation. *Computational Optimization and Applications*, 28:173–184, 2004.
15. B. Schölkopf, P. Bartlett, A. Smola, and R. Williamson. Shrinking the tube: a new support vector regression algorithm. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 330–336, Cambridge, MA, 1999. MIT Press. Available at http://www.kernel-machines.org/publications.html.
16. B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.
17. A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill–Posed Problems*. John Wiley & Sons, New York, 1977.
18. V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, second edition, 2000.