SMO vs PDCO for SVM: Sequential Minimal Optimization vs Primal-Dual interior method for Convex Objectives for Support Vector Machines

Ding Ma* Michael Saunders*

Working paper, January 2015

1 Introduction

In machine learning, support vector machines (SVMs) are one of the best supervised learning models for data classification and regression analysis. Sequential minimal optimization (SMO) is an algorithm for solving the quadratic programming (QP) problem that arises in SVMs. It was invented by John Platt at Microsoft Research in 1998 [3] and is widely used for solving SVM models.

PDCO (Primal-Dual interior method for Convex Objectives) is a MATLAB primal-dual interior method for solving linearly constrained optimization problems with a convex objective function [4].

This report provides an overview of SVMs and the SMO algorithm, and a reformulation of the SVM QP problem to suit PDCO. The PDCO algorithm is then presented in detail, including the barrier subproblems arising in interior methods, the associated Newton equations, and the sparse-matrix computation used to calculate each search direction.

We compare PDCO with SMO on the data sets used in Platt's SMO paper: one real-world data set concerning Income Prediction, and two artificial data sets that simulate two extreme cases (good or poor data separation). The solver computation times are compared, with PDCO showing better scalability. We conclude with discussion of the results and possible future work.

1.1 Overview of SVM

The support vector machine was invented by Vladimir Vapnik in 1979 [5]. In the linear classifier for a binary classification case, an SVM constructs a hyperplane that separates a set of points labeled $y = -1$ from a set of points labeled $y = +1$, using a vector of features x. The hyperplane is represented by $x^T w + b = 0$, with vector w and scalar b to be determined. To achieve more confident classification, the classifier should correctly separate the positive labeled points and the negative labeled points with a bigger "gap" proportional to $1/\|w\|$. After normalizing the distances from the nearest points to the hyperplane on both sides, we have the optimization problem

$$\min_{w,b} \ \frac{1}{2}\|w\|^2 \ \text{ s.t. } \ y_i(x_i^T w + b) \geq 1, \ \forall i,$$

where (x_i, y_i) denotes the features and label of the ith training sample.

By Lagrange duality, the above optimization problem is converted to the following dual optimization problem:

$$\begin{aligned} \max_{\alpha} \quad & W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j x_i^T x_j \\ \text{s.t.} \quad & \alpha_i \geq 0, \ \forall i \\ & \sum_{i=1}^{m} \alpha_i y_i = 0. \end{aligned}$$

Once the dual problem is solved, the normal vector w and the threshold b can be determined by

$$w = \sum_{i=1}^{m} \alpha_i y_i x_i, \ b = y_j - x_j^T w \ \text{ for some } \ \alpha_j > 0.$$

For the non-separable case, l_1 regularization is applied to make the algorithm still plausible and less sensitive to outliers. The original optimization problem is reformulated as follows:

$$\min_{w,b} \ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \xi_i \ \text{ s.t. } \ y_i(x_i^T w + b) \geq 1 - \xi_i, \xi_i \geq 0, \ \forall i.$$

Again by Lagrange duality, the following dual optimization problem is formed:

$$\begin{aligned} \max_{\alpha} \quad & W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j x_i^T x_j \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, \ \forall i \\ & \sum_{i=1}^{m} \alpha_i y_i = 0. \end{aligned} \tag{1}$$

Problem (1) is solved by sequential minimal optimization (SMO).

1.2 SMO for SVM

SMO is a special application of the coordinate ascent method for the SVM dual optimization problem (1). The SMO algorithm is as follows:

*Dept of Management Science and Engineering, Stanford University, Stanford, CA, USA

Repeat until convergence {

1. Heuristically choose a pair of α_i and α_j
2. Keeping all other α's fixed, optimize $W(\alpha)$ with respect to α_i and α_j.

}

The convergence of SMO is tested by checking if the KKT complementary conditions of (1) are satisfied within some tolerance around 10^{-3}. The KKT complementary conditions are:

$$\begin{aligned} \alpha_i = 0 &\Rightarrow y_i(x_i^T w + b) \geq 1 \\ \alpha_i = C &\Rightarrow y_i(x_i^T w + b) \leq 1 \\ 0 < \alpha_i < C &\Rightarrow y_i(x_i^T w + b) = 1 \end{aligned}$$

2 SVM reformulation

The SVM dual optimization problem (1) can be reformulate in a way that is suitable for a different optimization algorithm, PDCO. We define some matrix and vector quantities as follows:

$$X = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_m^T \end{pmatrix}, \qquad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}, \qquad \alpha = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_m \end{pmatrix},$$

$$Y = \mathrm{diag}(y_i), \qquad D = \mathrm{diag}(y_i \alpha_i),$$

where X is an $m \times n$ matrix with rows representing training samples and columns corresponding to features for each sample. Then $\sum_{i=1}^m y_i \alpha_i x_i^T = e^T DX$ and $De = Y\alpha$, where e is a vector of 1's. Thus,

$$W(\alpha) = \frac{1}{2} e^T DXX^T De - e^T \alpha = \frac{1}{2} \alpha^T Y^T XX^T Y \alpha - e^T \alpha.$$

If we define $Z = X^T Y \in \mathbb{R}^{n \times m}$ and $v = -Z\alpha$, we obtain a new formulation of SVM as follows:

$$\begin{aligned} \min_{\alpha, v} \; W(\alpha, v) = \quad & -e^T \alpha + \tfrac{1}{2} v^T v && (2) \\ \text{s.t.} \quad & Z\alpha + v = 0 && (3) \\ & y^T \alpha = 0 && (4) \\ & 0 \leq \alpha \leq Ce && (5) \end{aligned}$$

Note that this formulation of the SVM problem is different from five other formulations mentioned by Ferris and Munson [1] for their large-scale interior-point QP solver.

3 PDCO

PDCO (Primal-Dual Method for Convex Objective) is a MATLAB solver for optimization problems that are nominally of the form

CO	$\min_x \; \phi(x)$ s.t. $\; Ax = b, \quad \ell \leq x \leq u,$

where $\phi(x)$ is a convex function with known gradient $g(x)$ and Hessian $H(x)$, and $A \in \mathbb{R}^{m \times n}$. The format of CO is suitable for any linear constraints. For example, a double-sided constraint $\alpha \leq a^T \widetilde{x} \leq \beta$ ($\alpha < \beta$) should be entered as $a^T \widetilde{x} - \xi = 0$, $\alpha \leq \xi \leq \beta$, where $\widetilde{x}$ and ξ are relevant parts of x.

To allow for constrained least-squares problems, and to ensure unique primal and dual solutions (and improved stability), PDCO really solves the regularized problem

CO2	$\min_{x,\, r} \; \phi(x) + \frac{1}{2}\|D_1 x\|^2 + \frac{1}{2}\|r\|^2$ s.t. $\; Ax + D_2 r = b, \quad \ell \leq x \leq u,$

where D_1, D_2 are specified positive-definite diagonal matrices. The diagonals of D_1 are typically small (10^{-3} or 10^{-4}). Similarly for D_2 if the constraints in CO should be satisfied reasonably accurately. For least-squares applications, some diagonals of D_2 will be 1. Note that some elements of ℓ and u may be $-\infty$ and $+\infty$ respectively, but we expect no large numbers in A, b, D_1, D_2. If $\|D_2\|$ is small, we would expect A to be under-determined ($m < n$). If $D_2 = I$, A may have any shape.

3.1 The barrier approach

First we introduce slack variables x_1, x_2 to convert the bounds to non-negativity constraints:

CO3	$\min_{x,\, r,\, x_1,\, x_2} \; \phi(x) + \frac{1}{2}\|D_1 x\|^2 + \frac{1}{2}\|r\|^2$ s.t. $\; Ax + D_2 r = b$, $\; x - x_1 = \ell$, $\; x + x_2 = u$, $\; x_1, x_2 \geq 0.$

Then we replace the non-negativity constraints by the log barrier function, obtaining a sequence of convex subproblems with decreasing values of μ ($\mu > 0$):

CO(μ)	$\min_{x,\, r,\, x_1,\, x_2} \; \phi(x) + \frac{1}{2}\|D_1 x\|^2 + \frac{1}{2}\|r\|^2 - \mu \sum_j \ln([x_1]_j [x_2]_j)$		
	$Ax + D_2 r =$	b	$: y$
s.t.	$x - x_1 =$	ℓ	$: z_1$
	$-x - x_2 =$	$-u,$	$: z_2$

where y, z_1, z_2 denote dual variables for the associated constraints. With $\mu > 0$, most variables are strictly positive: $x_1, x_2, z_1, z_2 > 0$. (Exceptions: If $\ell_j = -\infty$ or $u_j = \infty$, the corresponding equation is omitted and the jth element of x_1 or x_2 doesn't exist.)

The KKT conditions for the barrier subproblem involve the three *primal* equations of CO(μ), along with four *dual* equations stating that the gradient of the subproblem objective should be a linear combination of the gradients of the primal constraints:

$$\begin{aligned} Ax + D_2 r &= b \\ x - x_1 &= \ell \\ -x - x_2 &= -u \\ A^T y + z_1 - z_2 &= g(x) + D_1^2 x && : x \\ D_2 y &= r && : r \\ X_1 z_1 &= \mu e && : x_1 \\ X_2 z_2 &= \mu e, && : x_2 \end{aligned}$$

where $X_1 = \text{diag}(x_1)$, $X_2 = \text{diag}(x_2)$, and similarly for Z_1, Z_2 later. The last two equations are commonly called the perturbed complementarity conditions. Initially they are in a different form. The dual equation for x_1 is really

$$-z_1 = \nabla(-\mu \ln(x_1)) = -\mu X_1^{-1} e,$$

where e is a vectors of 1's. Thus, $x_1 > 0$ implies $z_1 > 0$, and multiplying by $-X_1$ gives the equivalent equation $X_1 z_1 = \mu e$ as stated.

3.2 Newton's method

We now eliminate $r = D_2 y$ and apply Newton's method:

$$\begin{aligned}
A(x + \Delta x) + D_2^2(y + \Delta y) &= b \\
(x + \Delta x) - (x_1 + \Delta x_1) &= \ell \\
-(x + \Delta x) - (x_2 + \Delta x_2) &= -u \\
A^T(y + \Delta y) + (z_1 + \Delta z_1) - (z_2 + \Delta z_2) &= g + H\Delta x + D_1^2(x + \Delta x) \\
X_1 z_1 + X_1 \Delta z_1 + Z_1 \Delta x_1 &= \mu e \\
X_2 z_2 + X_2 \Delta z_2 + Z_2 \Delta x_2 &= \mu e,
\end{aligned}$$

where g and H are the current objective gradient and Hessian. To solve this Newton system, we work with three sets of residuals:

$$\begin{pmatrix} \Delta x - \Delta x_1 \\ -\Delta x - \Delta x_2 \end{pmatrix} = \begin{pmatrix} r_\ell \\ r_u \end{pmatrix} \equiv \begin{pmatrix} \ell - x + x_1 \\ -u + x + x_2 \end{pmatrix}, \tag{6}$$

$$\begin{pmatrix} X_1\Delta z_1 + Z_1 \Delta x_1 \\ X_2 \Delta z_2 + Z_2 \Delta x_2 \end{pmatrix} = \begin{pmatrix} c_\ell \\ c_u \end{pmatrix} \equiv \begin{pmatrix} \mu e - X_1 z_1 \\ \mu e - X_2 z_2 \end{pmatrix}, \tag{7}$$

$$\begin{pmatrix} A\Delta x + D_2^2 \Delta y \\ -H_1 \Delta x + A^T \Delta y + \Delta z_1 - \Delta z_2 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \end{pmatrix} \tag{8}$$

$$\equiv \begin{pmatrix} b - Ax - D_2^2 y \\ g + D_1^2 x - A^T y - z_1 + z_2 \end{pmatrix}, \tag{9}$$

where $H_1 = H + D_1^2$. We use (6) and (7) to replace two sets of vectors in (8). With

$$\begin{pmatrix} \Delta x_1 \\ \Delta x_2 \end{pmatrix} = \begin{pmatrix} -r_\ell + \Delta x \\ -r_u - \Delta x \end{pmatrix}, \ \begin{pmatrix} \Delta z_1 \\ \Delta z_2 \end{pmatrix} = \begin{pmatrix} X_1^{-1}(c_\ell - Z_1 \Delta x_1) \\ X_2^{-1}(c_u - Z_2 \Delta x_2) \end{pmatrix}, \tag{10}$$

$$\begin{aligned}
H_2 &\equiv H + D_1^2 + X_1^{-1} Z_1 + X_2^{-1} Z_2 \\
w &\equiv r_2 - X_1^{-1}(c_\ell + Z_1 r_\ell) + X_2^{-1}(c_u + Z_2 r_u)
\end{aligned} \tag{11}$$

we find that

$$\begin{pmatrix} -H_2 & A^T \\ A & D_2^2 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} w \\ r_1 \end{pmatrix}. \tag{12}$$

3.3 Keeping x safe

If the objective is the entropy function $\phi(x) = \sum x_j \ln(x_j)$, for example, it is essential to keep $x > 0$. However, problems CO3 and CO(μ) treat x as having no bounds (except when $x - x_1 = \ell$ and $x + x_2 = u$ are satisfied in the limit). Thus (since April 2010), PDCO safeguards x at every iteration by setting

$$\begin{aligned}
x_j &= \ \ [x_1]_j \ \text{ if } \ell_j = 0 \quad (\text{and } \ell_j < u_j), \\
x_j &= -[x_2]_j \ \text{ if } u_j = 0 \quad (\text{and } \ell_j < u_j).
\end{aligned}$$

3.4 Solving for $(\Delta x, \Delta y)$

If $\phi(x)$ is a general convex function with known Hessian H, system (12) may be treated by direct or iterative solvers. Since it is an SQD system, sparse LDL$^{\text{T}}$ factors should be sufficiently stable under the same conditions as for regularized LO: $\|A\| \approx 1$, $\|H\| \approx 1$, and $\gamma\delta \gg 10^{-8}$, where γ and δ are the minimum values of the diagonals of D_1 and D_2. If `K` represents the sparse matrix in (12), PDCO uses the following lines of code to achieve reasonable efficiency:

```
s          = symamd(K);% sqd ordering (first iteration only)
rhs        = [w; r1];
thresh     = eps; % eps ~= 2e-16 suppresses partial pivoting
[L,U,p]    = lu(K(s,s),thresh,'vector');    % expect p = I
sqdsoln    = U\(L\rhs(s));
sqdsoln(s) = sqdsoln;
dx         = sqdsoln(1:n);
dy         = sqdsoln(n+1:n+m);
```

The `symamd` ordering can be reused for all iterations, and with row interchanges effectively suppressed by `thresh = eps`, we expect the *original* sparse LU factorization in MATLAB to do no further permutations (`p = 1:n+m`).

Alternatively a sparse Cholesky factorization $H_2 = LL^T$ may be practical, where $H_2 = H +$ diagonal terms (11), and L is a nonsingular permuted triangle. This is trivial if $\phi(x)$ is a separable function, since H and H_2 in (11) are then diagonal. System (12) may then be solved by eliminating either Δx or Δy:

$$(A^T D_2^{-2} A + H_2)\Delta x = A^T D_2^{-2} r_1 - w, \ D_2^2 \Delta y = r_1 - A\Delta x, \tag{13}$$

$$(AH_2^{-1}A^T + D_2^2)\Delta y = AH_2^{-1} w + r_1, \ \ H_2 \Delta x = A^T \Delta y - w. \tag{14}$$

Sparse Cholesky factorization may again be applicable, but if an iterative solver must be used it is preferable to regard them as least-squares problems suitable for LSQR or LSMR:

$$\min_{\Delta x} \left\| \begin{pmatrix} D_2^{-1} A \\ L^T \end{pmatrix} \Delta x - \begin{pmatrix} D_2^{-1} r_1 \\ -L^{-T} w \end{pmatrix} \right\|^2, \ D_2 \Delta y = D_2^{-1}(r_1 - A\Delta x), \tag{15}$$

$$\min_{\Delta y} \left\| \begin{pmatrix} L^{-1} A^T \\ D_2 \end{pmatrix} \Delta y - \begin{pmatrix} L^{-1} w \\ D_2^{-1} r_1 \end{pmatrix} \right\|^2, \ L^T \Delta x = L^{-1}(A^T \Delta y - w). \tag{16}$$

The right-most vectors in (15)–(16) are part of the residual vectors for the least-squares problems (and may be by-products from the least-squares solver).

3.5 Success

PDCO has been applied to some large web-traffic network problems with the entropy function $\sum x_j \ln x_j$ as objective. Search directions were obtained by applying LSQR to (16) with diagonal $H = X^{-1}$ and $L = H_2^{-1/2}$, and $D_1 = 0$, $D_2 = 10^{-3} I$. A problem with 50,000 constraints and 660,000 variables (an explicit sparse A) solves in about 3 minutes on a 2GHz PC, requiring less than 100 total LSQR iterations. At the time (2003), this was unexpectedly remarkable performance. Both the entropy function and the network matrix A are evidently amenable to interior methods.

4 BENCHMARKING PDCO

PDCO was tested here against SMO on a series of benchmarking problems used by John Platt's SMO paper: one real-world data set concerning Income Prediction, and two different artificial data sets that simulate two extreme scenarios (perfectly linearly separable and noise).

1. Income Prediction: decide whether the household income is greater than $50,000. The data used are publicly available at http://ftp.ics.uci.edu/pub/machine-learning-databases/adult/. There were 32561 examples in training set. Eight features are categorical and six are continuous. The six continuous attributes were discretized into quintiles, which yielded a total of 129 binary features, of which 14 are true. The training set size was varied, and the limiting value C is chosen to be 0.05. The results are compared in Table 1 and Figure 1.

2. Separable data: the data were randomly generated 300-dimensional binary vectors, with a 10% fraction of 1's. A 300-dimensional weight vector was generated randomly in [-1,1], to decide the labels of examples. A positive label is assigned to the input point whose dot product with the weight vector is greater than 1; and if the dot product is less than -1, the point is labeled -1; otherwise, the point is discarded. ($C = 100$) The results are shown in Table 2 and Figure 2.

3. Noise data: generated 300-dimensional random binary input points (10% 1's) and random output labels. ($C = 0.1$) In this case, SVM is fitting pure noise. The results are shown in Table 3 and Figure 3.

Specifically, for equations (2)–(5), the quantities in PDCO's problem CO2 are $x = \alpha$, $r = [v; \delta]$, $\phi(x) = -e^T\alpha$, $A = [Z; y^T] = [X^TY; y^T]$, $H = 0$, $D_1 = 10^{-3}I$, and $D_2 = I$ (except the last diagonal of D_2 is $\rho = 10^{-4}$ to represent equality constraint (4) as $y^T\alpha + \rho\delta = 0$, where $\rho\delta$ will be very small). The feasibility and optimality tolerances were set to 10^{-6}, giving an accurate solution to well-scaled data. PDCO's new CO2 are as follows:

$$\begin{aligned} \min_{\alpha,v,\delta} \quad & W(\alpha, v, \delta) = -e^T\alpha + \frac{1}{2}\begin{pmatrix} v \\ \delta \end{pmatrix}^T \begin{pmatrix} v \\ \delta \end{pmatrix} \\ \text{s.t.} \quad & \begin{pmatrix} X^TY \\ y^T \end{pmatrix}\alpha + \begin{pmatrix} I & \\ & \rho \end{pmatrix}\begin{pmatrix} v \\ \delta \end{pmatrix} = 0 \\ & 0 \le \alpha \le Ce \end{aligned}$$

The main work per iteration for PDCO lies in forming the matrix $AH_2^{-1}A^T + D_2^2$ in (14) and its sparse Cholesky factorization. The computation times for SMO and PDCO on three sets of data with varying numbers of samples are shown in Tables 1–3. The SMO times were obtained with C++ on a 266 MHz Pentium II processor, which is approximately 10 times slower than Matlab on the iMac 2.93GHz machine that we used for PDCO. To make the CPU times roughly comparable, we divided the SMO times by 10 in the Figures 1–3.

Table 1: SMO vs PDCO on Income Prediction data

Data size	PDCO time	SMO time
1605	0.187	0.4
2265	0.448	0.9
3185	0.546	1.8
4781	0.829	3.6
6414	1.15	5.5
11221	2.46	17.0
16101	3.55	35.3
22697	6.44	85.7
32561	10.35	163.6

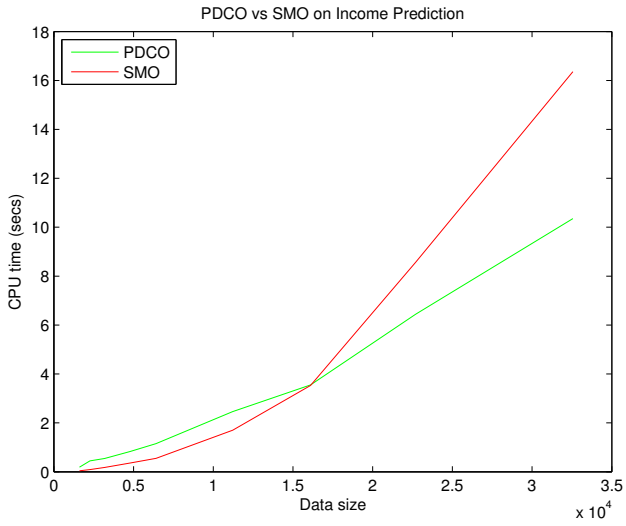

Figure 1: PDCO vs SMO on Income Prediction

5 Discussion

The tables show that the computation times for SMO scale approximately linearly on all three data sets. Plotting these against times for PDCO revealed different slopes for each method. The figures show that PDCO scales better with data size m than SMO on these examples. One reason is that the number of PDCO iterations increases slowly with m. Each iteration requires a sparse Cholesky factorization of an $n \times n$ matrix of the form X^TDX for some positive diagonal matrix D, with the number of features n being only 129 or 300 in these examples. The storage for the Cholesky factor depends on the sparsity of X and X^TX. This storage is minimal for $n \le 10,000$ (say), but may remain tolerable for much bigger n if X and X^TX remain extremely sparse. For problems exceeding the capacity of RAM, there are distributed implementations of sparse Cholesky factorization, such as MUMPS [2].

Ferris and Munson [1] have already shown that very large SVM problems can be processed by interior methods using distributed memory. The same would be true with a distributed implementation of PDCO. This project suggests that sparse interior-point methods may prove to be competitive with

Table 2: SMO vs PDCO on Separable data

Data size	PDCO time	SMO time
1000	0.470	15.3
2000	0.695	33.4
5000	1.39	103.0
10000	3.08	186.8
20000	7.39	280.0

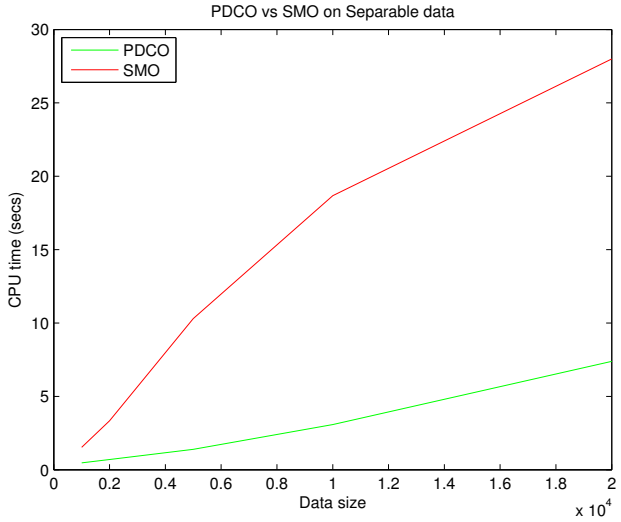

Figure 2: PDCO vs SMO on Separable data

Table 3: SMO vs PDCO on Noise data

Data size	PDCO time	SMO time
500	0.306	1.0
1000	0.399	3.5
2000	0.591	15.7
5000	1.27	67.6
10000	2.41	187.1

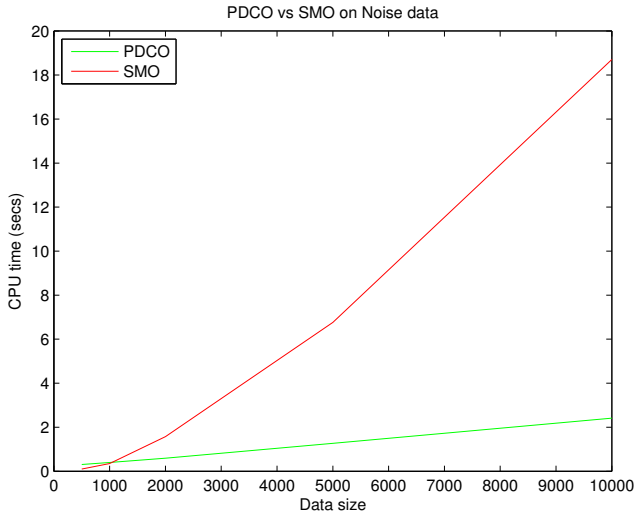

Figure 3: PDCO vs SMO on Noise data

coordinate-descent methods like SMO. We hope that future implementations will resolve this question.

References

[1] M. C. Ferris and T. S. Munson. Interior-point methods for massive support vector machines. *SIAM J. Optim.*, 13(3):783–804, 2003.

[2] MUMPS: a multifrontal massively parallel sparse direct solver. `http://mumps.enseeiht.fr/`.

[3] J. C. Platt. Sequential Minimal Optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, Microsoft Research, 1998.

[4] M. A. Saunders. PDCO convex optimization software. `http://stanford.edu/group/SOL/software/pdco/`.

[5] V. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, 1982.