基于连续过松弛方法的支持向量回归算法*

全 勇+, 杨 杰, 姚莉秀, 叶晨洲

(上海交通大学 图像处理及模式识别研究所,上海 200030)

Successive Overrelaxation for Support Vector Regression

QUAN Yong+, YANG Jie, YAO Li-Xiu, YE Chen-Zhou

(Institute of Image Processing and Pattern Recognition, Shanghai Jiaotong University, Shanghai 200030, China)

+ Corresponding author: Phn: +86-10-32261230, E-mail: quanysjtu@sjtu.edu.cn

Received 2002-12-08; Accepted 2003-04-15

Quan Y, Yang J, Yao LX, Ye CZ. Successive overrelaxation for support vector regression. *Journal of Software*, 2004,15(2):200~206.

http://www.jos.org.cn/1000-9825/15/200.htm

Abstract: Training a SVR (support vector regression) requires the solution of a very large QP (quadratic programming) optimization problem. Despite the fact that this type of problem is well understood, the existing training algorithms are very complex and slow. In order to solve these problems, this paper firstly introduces a new way to make SVR have the similar mathematic form as that of a support vector machine. Then a versatile iterative method, successive overrelaxation, is proposed. Experimental results show that this new method converges considerably faster than other methods that require the presence of a substantial amount of data in memory. The results give guidelines for the application of this method to large domains.

Key words: support vector regression; support vector machine; successive overrelaxation; quadratic programming; chunking algorithm

摘 要: 支持向量回归(support vector regression,简称 SVR)训练算法需要解决在大规模样本条件下的凸二次规划(quadratic programming,简称 QP)问题.尽管此种优化算法的机理已经有了较为明确的认识,但已有的支持向量回归训练算法仍较为复杂且收敛速度较慢.为解决这些问题.首先采用扩展方法使 SVR 与支撑向量机分类(SVC)具有相似的数学形式,并在此基础上针对大规模样本回归问题提出一种用于 SVR 的简化 SOR(successive overrelaxation)算法.实验表明,这种新的回归训练方法在数据量较大时,相对其他训练方法有较快的收敛速度,特别适于在大规模样

* Supported by the National Natural Science Foundation of China under Grant No.50174038 (国家自然科学基金); partly supported by the National Natural Science Foundation of China under Grant No.30170274 (国家自然科学基金)

QUAN Yong was born in 1976. He is a Ph.D. candidate at the Institute of Image Processing and Pattern Recognition, Shanghai Jiaotong University. His current research areas include machine learning and data mining. **YANG Jie** was born in 1964. He is a professor and doctoral supervisor at the Institute of Image Processing and Pattern Recognition, Shanghai Jiaotong University. His research areas are image processing, pattern recognition, data mining and application. **YAO Li-Xiu** was born in 1973. She is an associate professor at the Institute of Image Processing and Pattern Recognition, Shanghai Jiaotong University. Her current research areas include data mining techniques and their applications. **YE Chen-Zhou** was born in 1974. He is a Ph.D. candidate at the Institute of Image Processing and Pattern Recognition, Shanghai Jiaotong University. His current research areas include artificial intelligence and data mining.

本条件下的回归训练算法设计.

关键词: 支持向量回归;支持向量机;SOR 算法;凸二次规划;chunking 算法
中图法分类号: TP18 文献标识码: A

In the last few years, there has been a surge of interest in SVMs (support vector machines)[1]. SVMs have empirically been shown to give a good generalization performance on a wide variety of problems. However, the use of SVMs is still limited to a small group of researchers. One possible reason is that training algorithms for SVMs are slow, especially for large problems. Another explanation is that SVM training algorithms are complex, subtle, and sometimes difficult to implement.

In 1997, a theorem[2] that introduces a whole new family of SVM training procedures was proved. In a nutshell, Osuna's theorem shows that the global SVM training problem can be broken down into a sequence of smaller sub-problems and that optimizing each sub-problem minimizes the original QP problem. Even more recently, the sequential minimal optimization algorithm (SMO) was introduced[3] as an extreme example of Osuna's theorem in practice. Because SMO uses a sub-problem of size two, each sub-problem has an analytical solution. Thus, for the first time, SVMs could be optimized without a QP solver.

In addition to SMO, other new methods[4,5] have been proposed for optimizing SVMs online without a QP solver. While these other online methods hold a great promise, SMO is the only online SVM optimizer that explicitly exploits the quadratic form of the objective function and simultaneously uses the analytical solution of the size two cases.

While SMO has been shown to be effective on sparse data sets and especially fast for linear SVMs, the algorithm can be extremely slow on non-sparse data sets and on problems that have many support vectors. Regression problems are especially prone to these issues because the inputs are usually non-sparse real numbers (as opposed to binary inputs) with solutions that have many support vectors. Because of these constraints, Shevade[6] and Chih-Jen[7] generalized SMO so that it can handle regression problems. However, one problem with SMO is that its rate of convergence slows down dramatically when data are non-sparse and when there are many support vectors in the solution — as is often the case in regression — because kernel function evaluations tend to dominate the runtime in this case. Moreover, caching kernel function outputs can easily degrade SMO's performance even more because SMO tends to access kernel function outputs in an unstructured manner.

Mangasarian[8] proposed a new training algorithm based on SOR (successive overrelaxation). Because SOR handles one point at a time, similar to Platt's SMO algorithm which handles two constraints at a time and Joachims' SVM which handles a small number of points at a time, SOR can process very large datasets that need not reside in memory. The algorithm converges linearly to a solution.

In this work, we propose a new way to make SVR (support vector regression) have the similar mathematic form as that of a support vector classification, and derive a generalization of SOR to handle regression problems. Simulation results indicate that the modification to SOR for regression problems yields dramatic runtime improvements.

1 Successive Overrelaxation for Support Vector Machine

Given a training set of instance-label pairs (x_i,y_i), $i=1,\ldots,l$, where $x_i\in R^n$ and $y\in\{1,-1\}^l$, Mangasarian outlines the key modifications from standard SVM to RSVM[8]. He starts from adding an additional term $b^2/2$ to the objective function:

$$\begin{cases}\min\limits_{w,b,\xi} \dfrac{1}{2}\left(w^T w+b^2\right)+C\sum\limits_{i=1}^{l}\xi_i \\ \text{s.t.}\quad y_i\left(w^T\varphi(x_i)+b\right)\geq 1-\xi_i \\ \quad \xi_i\geq 0, i=1,...,l\end{cases} \tag{1}$$

The training is expressed as a minimization of a dual quadratic form:

$$\begin{cases}\min \dfrac{1}{2}a^T Ha+\dfrac{1}{2}a^T Ea-e^T a \\ \text{s.t.}\quad 0\leq a_i\leq C, i=1,...,l\end{cases} \tag{2}$$

where $H=ZZ^T$, $Z=\begin{pmatrix}y_1\varphi(x_1)\\ \vdots \\ y_l\varphi(x_l)\end{pmatrix}_{l\times 1}$, $E=YY^T$, $Y=\begin{pmatrix}y_1\\ \vdots \\ y_l\end{pmatrix}_{l\times 1}$, $e=\begin{pmatrix}1\\ \vdots \\ 1\end{pmatrix}_{l\times 1}$, $a=\begin{pmatrix}a_1\\ \vdots \\ a_l\end{pmatrix}_{l\times 1}$.

Define $A=H+E$, $L+D+L^T=A$. The nonzero elements of $L\in R^{l\times l}$ constitute the strictly lower triangular part of the symmetric matrix A, and the nonzero elements of $D\in R^{l\times l}$ constitute the diagonal of A. The SOR method, which is a matrix splitting method that converges linearly to a point a satisfying (2), leads to the following algorithm:

Choose $\omega\in(0,2)$. Start with any $a^0\in R^l$. Have a^i to compute a^{i+1} as follows:

$$a^{i+1}=\left(a^i-\omega D^{-1}\left(Aa^i-e+L\left(a^{i+1}-a^i\right)\right)\right)_* \tag{3}$$

where $(\cdot)_*$ denotes the 2-norm projection on the feasible region of (2), that is

$$(a_i)_*=\begin{cases}0 & a_i\leq 0\\ a_i & 0<a_i<C\\ C & a_i\geq C\end{cases},\quad i=1,...,l.$$

2 Simplified SVR and Its SOR Algorithm

Most of those already existing training methods are originally designed to only be applicable to SVM. Compared with SVM, SVR has a more complicated form. For SVR, there are two sets of slack variables $(\xi_1,...,\xi_l)$ and $(\xi_1^*,...,\xi_l^*)$ and their corresponding dual variables $(a_1,...,a_l)$ and $(a_1^*,...,a_l^*)$. The analytical solution to the size two QP problems must be generalized in order to work on regression problems. Even though Smola has generalized the SMO to handle regression problems, one has to distinguish four different cases: (a_i,a_j), (a_i,a_j^*), (a_i^*,a_j), and (a_i^*,a_j^*). This makes the training algorithm more complicated and difficult to be implemented. In this paper, we propose a new way to make SVR have the similar mathematic form as that of a support vector classification, and derive a generalization of SOR to handle regression problems.

2.1 Simplified SVR formulation

Similar to (1), we also introduce an additional term $b^2/2$ to SVR. Hence we arrive at the formulation stated as follows:

$$\begin{cases}\min\ \dfrac{1}{2}\left(w^T w+b^2\right)+C\sum\limits_{i=1}^{l}\left(\xi_i+\xi_i^*\right) \\ \text{s.t.}\quad y_i-w\varphi(x_i)-b\leq\varepsilon+\xi_i \\ \quad w\varphi(x_i)+b-y_i\leq\varepsilon+\xi_i^* \\ \quad \xi_i,\xi_i^*\geq 0\qquad i=1,...,l\end{cases} \tag{4}$$

By introducing two dual sets of variables, we construct a Lagrange function from both the objective function and the corresponding constraints. It can be shown that this function has a saddle point with respect to the primal

and dual variables at the optimal solution:

$$\begin{aligned} L &= \frac{1}{2}w^T w+\frac{1}{2}b^2+C\sum_{i=1}^{l}\left(\xi_i+\xi_i^*\right)-\sum_{i=1}^{l}a_i\left(\varepsilon+\xi_i-y_i+w\varphi(x_i)+b\right)- \\ &\sum_{i=1}^{l}a_i^*\left(\varepsilon+\xi_i^*+y_i-w\varphi(x_i)-b\right)-\sum_{i=1}^{l}\left(\eta_i\xi_i+\eta_i^*\xi_i^*\right) \end{aligned} \tag{5}$$

It is understood that the dual variables in (5) have to satisfy positive constraints, i.e. $a_i,a_i^*,\eta_i,\eta_i^*\geq 0$. It follows from the saddle point condition that the partial derivatives of L with respect to the primal variables $\left(w,b,\xi_i,\xi_i^*\right)$ have to vanish for optimality.

$$\begin{cases} \dfrac{\partial L}{\partial b}=b+\sum_{i=1}^{l}\left(a_i^*-a_i\right)=0 & \Rightarrow\quad b=\sum_{i=1}^{l}\left(a_i-a_i^*\right) \\ \dfrac{\partial L}{\partial w}=w-\sum_{i=1}^{l}(a_i-a_i^*)\varphi(x_i)=0 & \Rightarrow\quad w=\sum_{i=1}^{l}\left(a_i-a_i^*\right)\varphi(x_i) \\ \dfrac{\partial L}{\partial \xi_i^{(*)}}=C-a_i^{(*)}-\eta_i^{(*)}=0 & \end{cases} \tag{6}$$

Substituting (6) into (5) yields the dual optimization problem:

$$\begin{cases} \min\quad \dfrac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\left(a_i-a_i^*\right)\left(a_j-a_j^*\right)\varphi(x_i)\varphi(x_j)+\dfrac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\left(a_i-a_i^*\right)\left(a_j-a_j^*\right)+ \\ \qquad \varepsilon\sum_{i=1}^{l}\left(a_i+a_i^*\right)-\sum_{i=1}^{l}\left(a_i-a_i^*\right)y_i \\ \text{s.t.}\quad a_i,a_i^*\in[0,C] \end{cases} \tag{7}$$

The main reason for introducing our variant (4) of the SVR is that its dual (7) does not contain an equality constraint, as does the dual optimization problem of the original SVR. This enables us to apply in a straightforward manner the effective matrix splitting methods such as those of Ref.[8] that process one constraint of (4) at a time through its dual variable, without the complication of having to enforce an equality constraint at each step on the dual variable *a*. This permits us to process massive data without bringing them all into a fast memory.

Define

$$H=ZZ^T,\ Z=\begin{pmatrix} d_1\varphi(x_1) \\ \vdots \\ d_l\varphi(x_l) \\ d_{l+1}\varphi(x_{l+1}) \\ \vdots \\ d_{2l}\varphi(x_{2l}) \end{pmatrix}_{2l\times 1},\ a=\begin{pmatrix} a_1 \\ \vdots \\ a_l \\ a_1^* \\ \vdots \\ a_l^* \end{pmatrix}_{2l\times 1},\ E=dd^T,\ d=\begin{pmatrix} 1 \\ \vdots \\ 1 \\ -1 \\ \vdots \\ -1 \end{pmatrix}_{2l\times 1},\quad c=\begin{pmatrix} y_1-\varepsilon \\ \vdots \\ y_l-\varepsilon \\ -y_1-\varepsilon \\ \vdots \\ -y_l-\varepsilon \end{pmatrix}_{2l\times 1}$$

Thus (7) can be expressed in a simpler way.

$$\begin{cases} \min\quad \dfrac{1}{2}a^T Ha+\dfrac{1}{2}a^T Ea-c^T a \\ \text{s.t.}\quad a_i\in[0,C],\quad i=1,\ldots,l \end{cases} \tag{8}$$

If we ignore the difference of matrix dimension, (8) and (2) have the same mathematic form. Compared with (7), (8) has a simpler mathematic form. However, the only difference between formulations (7) and (8) is the mathematic representation. If we expand (7) and (8) to polynomials with respect to *a*, their mathematical formulations will be identical. So their optimization problems are essentially the same. Through this important transformation, SVR and SVM can be written in the same mathematic form. According to the statistical learning theory, we can solve the optimization problem for SVR in the same way as SVM. Thus many training algorithms that are used in SVM can be used in SVR directly.

2.2 SOR algorithm for SVR

Here we let $A=H+E$, $L+D+L^T=A$. The nonzero elements of $L\in R^{2l\times 2l}$ constitute the strictly lower triangular part of the symmetric matrix A, and the nonzero elements of $D\in R^{2l\times 2l}$ constitute the diagonal of A. The SOR method, which is a matrix splitting method that converges linearly to a point a satisfying (8), leads to the following algorithm:

SOR Algorithm. Choose $\omega\in(0,2)$. Start with any $a^0\in R^{2l}$. Have a^i to compute a^{i+1} as follows:

$$a^{i+1}=\left(a^i-\omega D^{-1}\left(Aa^i-e+L\left(a^{i+1}-a^i\right)\right)\right)_* \tag{9}$$

until $\left\|a^{i+1}-a^i\right\|$ is less than some prescribed tolerance.

The components of a^{i+1} are computed in order of increasing component index. Thus the SOR iteration (9) consists of computing a_j^{i+1} using $\left(a_1^{i+1},...,a_{j-1}^{i+1},a_j^i,...,a_{2l}^i\right)$. That is, the latest computed components of *a* are used in the computation of a_j^{i+1}. The strictly lower triangular matrix *L* in (3) can be thought of as a substitution operator, substituting $\left(a_1^{i+1},...,a_{j-1}^{i+1}\right)$ for $\left(a_1^i,...,a_{j-1}^i\right)$. Thus, SOR can be interpreted as using each new component value of u immediately after it is computed, thereby achieving improvement over other iterative methods such as gradient methods where all component of a are updated at once.

Even though our SOR iteration (9) is written in terms of the full $2l\times 2l$ matrix A, it can easily be implemented one row at a time without bringing all of the data into memory as follows for $j=1,...,2l$:

$$a_j^{i+1}=\left(a_j^i-\varpi A_{jj}^{-1}\left(\sum_{k=j}^{2l}A_{jk}a_k^i-c_j+\sum_{k=1}^{j-1}A_{jk}a_k^{i+1}\right)\right)_* \tag{10}$$

A simple interpretation of this step is that one component of the multiplier a_j is updated at a time by bringing one constraint of (8) at a time.

3 Experimental Results

The SOR algorithm is tested against the standard chunking algorithm and against the SMO method on a series of benchmarks. The SOR, SMO and chunking are all written in C++, using Microsoft's Visual C++ 6.0 complier. Joachims' package SVM^{light} (version 2.01) with a default working set size of 10 is used to test the decomposition method. The CPU time of all algorithms are measured on an unloaded 633 MHz Celeron II processor running Windows 2000 professional.

The chunking algorithm uses the projected conjugate gradient algorithm as its QP solver, as suggested by Burges[1]. All algorithms use sparse dot product code and kernel caching. Both SMO and chunking share the folded linear SVM code.

Experiment 1. In the first experiment, we consider the approximation of the sinc function $f(x)=(\sin\pi x)/\pi x$. Here we use the kernel $K(x_1,x_2)=\exp\left(-\|x_1-x_2\|^2/\delta\right)$, $C=100$, $\delta=0.1$ and $\varepsilon=0.1$. Figure 1 shows the approximated results of SMO method and SOR method respectively.

In Fig.1(b) we can also observe the action of Lagrange multipliers, acting as forces $\left(a_i,a_i^*\right)$ pulling and pushing the regression inside the ε-tube. These forces, however, can only be applied at the samples where the regression touches or even exceeds the predetermined tube. This directly accords with the illustration of the KKT-conditions: either the regression lies inside the tube (hence the conditions are satisfied with a margin), and

Lagrange multipliers are 0, or the condition is exactly met and forces have to apply $a_i \neq 0$ or $a_i^* \neq 0$ to keep the constraints satisfied. This observation indicates that SOR method can handle regression problems successfully.

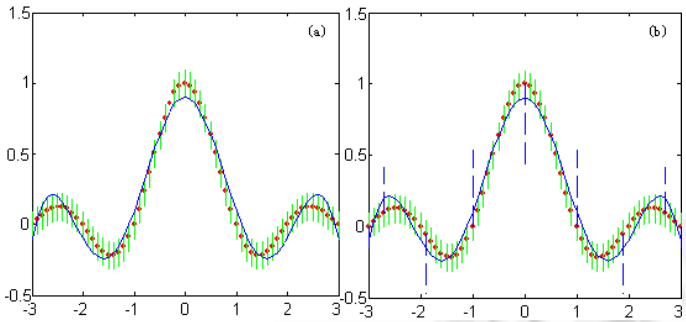

Fig.1 Approximation results of SMO method (a) and SOR method (b)

In Table 1 we can see that the SVMs trained with various methods have nearly the same approximation accuracy. When handling a small size of data sets, there is no great difference in time consumption.

Table 1 Approximation effect of SVMs using various methods

Experiment	Time (s)	Number of SVs	Expectation of error	Variance of error
SOR	0.432 ± 0.023	9 ± 1.26	0.0014 ± 0.0009	0.0053 ± 0.0012
SMO	0.467 ± 0.049	9 ± 0.4	0.0016 ± 0.0007	0.0048 ± 0.0021
Chunking	0.521 ± 0.031	9 ± 0	0.0027 ± 0.0013	0.0052 ± 0.0019
SVM^{light}	0.497 ± 0.056	9 ± 0	0.0021 ± 0.0011	0.0067 ± 0.0023

Experiment 2. In order to compare the time consumption of the different training methods on massive data sets, we test these algorithms on three real-world data sets.

In this experiment, we adopt the same data sets used in Ref.[9]. That is, we choose the Boston Housing and the Abalone dataset from the UCI Repository and the USPS database of handwritten digits. The first data is of size 506 (350 training, 156 testing), the Abalone dataset of size 4177 (3000 training and 1177 testing). In the first two cases the data were rescaled to zero mean and unit variance, coordinate-wise, while the USPS dataset remained

Table 2 Comparison on various data sets

Data set	Time (s)	Trainingset size	Number of SVs	Training error	Testing error
	SOR SMO Chunking SVM^{light}	SOR SMO Chunking SVM^{light}	SOR SMO Chunking SVM^{light}	SOR SMO Chunking SVM^{light}	SOR SMO Chunking SVM^{light}
Boston housing	0.15 ± 0.07 0.23 ± 0.11 2.74 ± 1.06 3.42 ± 0.08	350	168 ± 13 163 ± 15 167 ± 6 168 ± 10	25.0	23.4 ± 1.34 25.7 ± 0.94 24.3 ± 1.08 23.6 ± 1.12
Abalone	8.72 ± 2.31 12.63 ± 3.98 87.44 ± 12.01 88.37 ± 15.24	3000	1308 ± 24 1316 ± 7 1317 ± 15 1317 ± 11	2.5	2.46 ± 0.65 2.23 ± 0.81 2.37 ± 0.96 2.65 ± 0.72
USPS	97.41 ± 28.07 183.2 ± 16.85 1563.1 ± 54.6 1866.5 ± 46.7	29463	11534 ± 6 11532 ± 8 11533 ± 5 11534 ± 5	30.0	28.6 ± 1.43 33.4 ± 1.25 31.8 ± 1.16 29.4 ± 1.53

unchanged. Its data size is 40337 (29463 training and 10874 testing). Finally, the gender encoding in Abalone (male/female/infant) was mapped into $\{(1,0,0),(0,1,0),(0,0,1)\}$.

We adopt the same kernel function as Experiment 1. For the first dataset, Boston Housing dataset, the parameters are $C=1000$, $\delta=1.5$ and $\varepsilon=30$. For the second dataset, Abalone dataset, the parameters are $C=1000$, $\delta=5$ and $\varepsilon=3.5$. While for USPS dataset, these parameters are $C=1000$, $\delta=18$ and $\varepsilon=30$.

Table 2 illustrates the time consumption, training set size and number of support vectors for different training algorithms. Here we can see that as the size of data set increases, the difference of training time among these training algorithms also increases greatly. When the size of data set reaches 29463, the training time needed by Chunking and SVM^{light} is more than 16 times than that of SOR.

Experiment 3. In this experiment, we will use the SVM trained by SOR to predict the time series data set. Here we adopt Greenwich's sunspot data. We can gain these data from Greenwich's homepage http://science.msfc.nasa.gov/ssl/pad/solar/greenwch.htm. We use historic sunspot data to predict future sunspot data. Figure 2 shows the comparison between the real sunspot data and the predicted sunspot data. This illustrates that the SVM gives a good prediction to sunspot. This experiment shows that the SVM trained by SOR algorithm can be used in practical problems successfully.

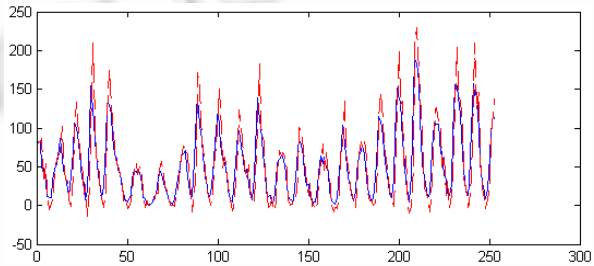

Fig.2 Comparison between real sunspot data (real line) and predicted sunspot data (dashed line)

Acknowledgement We gratefully acknowledge the constructive comments from the reviewers for improving this paper.

References:

[1] Burges CJC. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 1998,2(2): 121~167.

[2] Osuna E, Freund R, Girosi F. An improved training algorithm for support vector machines. In: Principle J, Giles L, Morgan N, eds. Proc. of the 1997 IEEE Workshop on Neural Networks and Signal Processing. Amelia Island: IEEE Press, 1997. 276~285.

[3] Platt J. Fast training of support vector machines using sequential minimal optimization. In: Schölkopf B, Burges C, Smola A, eds. Advances in Kernel Methods – Support Vector Learning. Cambridge, Massachusetts: MIT Press, 1998. 185~208.

[4] Mukherjee S, Osuna E, Girosi F. Nonlinear prediction of chaotic time series using support vector machines. In: Principle J, Giles L, Morgan N, eds. Proc. of the 1997 IEEE Workshop on Neural Networks and Signal Processing. Amelia Island: IEEE Press, 1997. 511~520.

[5] Keerthi SS, Shevade SK, Bhattacharyya C, Murthy KRK. A fast iterative nearest point algorithm for support vector machine classifier design. IEEE Trans. on Neural Networks, 2000,11(1):124~136.

[6] Shevade SK, Keerthi SS, Bhattacharyya C, Murthy KRK. Improvements to the SMO algorithm for SVM regression. IEEE Trans. on Neural Networks, 2000,11(5):1188~1193.

[7] Lin Chih-Jen. Asymptotic convergence of an SMO algorithm without any assumptions. IEEE Trans. on Neural Networks, 2002,13(1):248~250.

[8] Mangasarian OL, Musicant DR. Successive overrelaxation for support vector machines. IEEE Trans. on Neural Networks, 1999, 10(5):1032~1037.

[9] Smola AJ, Schölkopf B. Sparse greedy matrix approximation for machine learning. In: Langley P, ed. Proc. of the 17th Int'l Conf. on Machine Learning. San Francisco, 2000. 911~918. http://mlg.anu.edu.au/~smola/publications.html