A SIMPLE, FAST SUPPORT VECTOR MACHINE ALGORITHM FOR DATA MINING

Hiep-Thuan Do, Nguyen-Khang Pham, Thanh-Nghi Do
College of Information Technology, Cantho University
1 Ly Tu Trong Street, Ninh Kieu District
Cantho City, Vietnam
{dhthuan, pnkhang, dtnghi}@cit.ctu.edu.vn

Abstract: Support Vector Machines (SVM) and kernel related methods have shown to build accurate models but the learning task usually needs a quadratic programming, so that the learning task for large datasets requires big memory capacity and a long time. A new incremental, parallel and distributed SVM algorithm using linear or non linear kernels proposed in this paper aims at classifying very large datasets on standard personal computers. We extend the recent finite Newton classifier for building an incremental, parallel and distributed SVM algorithm. The new algorithm is very fast and can handle very large datasets in linear or non linear classification tasks. An example of the effectiveness is given with the linear classification into two classes of two million datapoints in 20-dimensional input space in some seconds on ten personal computers (3 GHz Pentium IV, 512 MB RAM, Linux).

Keywords: Support Vector Machine, Data Mining, Newton Method, Incremental Learning, Parallel and Distributed Algorithm, Large Datasets.

1. INTRODUCTION

In recent years, real-world databases increase rapidly (double every 9 months [11], [16]). So the need to extract knowledge from very large databases is increasing. Knowledge Discovery in Databases (KDD [10]) can be defined as the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data. Data mining is the particular pattern recognition task in the KDD process. It uses different algorithms for classification, regression, clustering and association. We are interested in SVM learning algorithms proposed by Vapnik [25] because they have shown practical relevance for classification, regression and novelty detection. Successful applications of SVMs have been reported for various fields, for example in face identification, text categorization and bioinformatics [13]. The approach is systematic and properly motivated by statistical learning theory. SVMs are the most well known algorithms of a class using the idea of kernel substitution [5]. SVM and kernel-based methods have become increasingly popular data mining tools. In spite of the prominent properties of SVM, they are not favorable to deal with the challenge of large datasets. SVM solutions are obtained from quadratic programs (QP), so that the computational cost of an SVM approach is at least square of the number of training datapoints and the memory requirement making SVM impractical. There is a need to scale up learning algorithms to handle massive datasets on personal computers (PCs). The effective heuristics to improve SVM learning task are to divide the original QP into series of small problems [2], [4], [20], [21], incremental learning [3], [12], [23] updating solutions in growing training set, parallel and distributed learning [22] on PC network or choosing interested datapoints subset (active set) for learning [8], [24], boosting of SVM [7] based on sampling techniques for scaling up learning.
We have created a new algorithm that is very fast for building incremental, parallel and distributed SVM classifiers. It is derived from the finite Newton method for classification proposed by Mangasarian [17]. The new SVM algorithm can linearly classify two million datapoints in 20-dimensional input space into two classes in some seconds on ten PCs (3 GHz Pentium IV, 512 MB RAM, Linux).

We briefly summarize the content of the paper now. In section 2, we introduce the finite Newton method for classification problems. In section 3, we describe how to build the incremental learning algorithm with the finite Newton method. In section 4, we describe our parallel and distributed versions of the incremental algorithm. We present numerical test results in section 5 before the conclusion in section 6.

Some notations are used in this paper. All vectors will be column vectors unless transposed to row vector by a T superscript. The inner dot product of two vectors, x, y is denoted by x.y. The 2-norm of the vector x will be denoted by ||x||. The matrix A[mxnx] will be m datapoints in the n-dimensional real space R^n. The classes +1, -1 of m datapoints are denoted by the diagonal matrix D[mxm] of -1, +1. e will be the column vector of 1. w, b will be the coefficients and the scalar of the hyper-plane. z will be the slack variable and C is a positive constant. I denotes the identity matrix.

2. FINITE STEPLESS NEWTON SUPPORT VECTOR MACHINE

Let us consider a linear binary classification task, as depicted in figure 1, with m datapoints in the n-dimensional input space R^n, represented by the mxn matrix A, having corresponding labels ±1, denoted by the mxm diagonal matrix D of ±1.

For this problem, the SVM algorithms try to find the best separating plane, i.e. furthest from both class +1 and class -1. It can simply maximize the distance or margin between the supporting planes for each class (x.T.w – b = +1 for class +1, x.T.w – b = -1 for class -1). The margin between these supporting planes is 2/||w|| (where ||w|| is the 2-norm of the vector w). Any point x_i falling on the wrong side of its supporting plane is considered as an error (having corresponding slack value z_i > 0). Therefore, a SVM algorithm has to simultaneously maximize
the margin and minimize the error. The standard SVM formulation with a linear kernel is given by the following QP (1):

\[
\min f(w, b, z) = C e^T z + (1/2)||w||^2 \\
\text{s.t. } D(Aw - eb) + z \geq e
\]

where slack variable \( z \geq 0 \), constant \( C > 0 \) is used to tune errors and margin size.

The plane \((w, b)\) is obtained by the solution of the QP (1). Then, the classification function of a new datapoint \( x \) based on the plane is: \( \text{predict}(x) = \text{sign}(w.x - b) \)

SVM can use some other classification functions, for example a polynomial function of degree \( d \), a RBF (Radial Basis Function) or a sigmoid function. To change from a linear to non-linear classifier, one must only substitute a kernel evaluation in (1) instead of the original dot product. More details about SVM and others kernel-based learning methods can be found in [1] and [5].

Recent developments for massive linear SVM algorithms proposed by Mangasarian [17], [18] reformulate the classification as an unconstrained optimization. By changing the margin maximization to the minimization of \((1/2)||<w, b>||^2\) and adding with a least squares 2-norm error, the SVM algorithm formulation with linear kernel is given by the QP (2).

\[
\min f(w, b, z) = (C/2)||z||^2 + (1/2)||<w, b>||^2 \\
\text{s.t. } D(Aw - eb) + z \geq e
\]

where slack variable \( z \geq 0 \), constant \( C > 0 \) is used to tune errors and margin size.

The formulation (2) can be rewritten by substituting for \( z = (e - D(Aw - eb))_+ \) (where \((x)_+\) replaces negative components of a vector \( x \) by zeros) into the objective function \( f \). We get an unconstrained problem (3):

\[
\min f(w, b) = (C/2)||e - D(Aw - eb)||_+^2 + (1/2)||<w, b>||^2 \\
\]

By setting \([w_1 \ w_2 \ ... \ w_n \ b]^T\) to \( u \) and \([A \ -e]^T\) to \( H \), then the SVM formulation (3) is rewritten by (4):

\[
\min f(u) = (C/2)||e - DHu||_+^2 + (1/2)u^Tu \\
\]

Mangasarian [17] has shown that the finite stepless Newton method can be used to solve the strongly convex unconstrained minimization problem (4). The algorithm can be described as the algorithm 1.
Algorithm 1. The finite stepless Newton SVM algorithm

- Input: training dataset represented by \( A \) and \( D \) matrices
- Starting with \( u_0 \in \mathbb{R}^{n+1} \) and \( i = 0 \)
- Repeat
  1) \( u_{i+1} = u_i - \partial^2 f(u_i)^{-1} \nabla f(u_i) \)
  2) \( i = i + 1 \)
Until \( \nabla f(u_i) = 0 \)
- Return \( u_i \)

Where the gradient of \( f \) at \( u_i \),

\[
\nabla f(u_i) = C(-DH)^T(e - DHu_i)_+ + u_i
\]  
(5)

and the generalized Hessian of \( f \) at \( u_i \),

\[
\partial^2 f(u_i) = C(-DH)^T \text{diag}((e - DHu_i)_+)(-DH) + I
\]  
(6)

with \( \text{diag}((e - DHu_i)_+) \) denotes the \((n+1)x(n+1)\) diagonal matrix whose \( j^{th} \) diagonal entry is sub-gradient of the step function \( (e - DHu_i)_+ \).

Mangasarian [17] has proved that the sequence \( \{u_i\} \) of the algorithm 1 terminates at the global minimum solution. In most of the tested cases, the stepless Newton algorithm has given the good solution with a number of iterations varying between 5 and 8.

The SVM formulation (4) requires thus only solutions of linear equations of \((w, b)\) instead of QP. If the dimensional input space is small enough (less than \( 10^4 \)), even if there are millions datapoints, the finite stepless Newton SVM algorithm is able to classify them in minutes on a PC.

The algorithms can deal with non-linear classification tasks; however at the input of the algorithm 1, the training dataset represented by \( A[mxn] \) is replaced by the matrix the kernel matrix \( K[mxm] \), where \( K \) is a non linear kernel created by whole dataset \( A \) and the support vectors being \( A \) too, e.g.:

- A degree \( d \) polynomial kernel of two datapoints \( x_i, x_j \) : \( K[i,j] = (x_i, x_j + 1)^d \)
- A radial basis kernel of two datapoints \( x_i, x_j \) : \( K[i,j] = \exp(-\gamma || x_i - x_j ||^2) \)

The finite stepless Newton SVM algorithm using the kernel matrix \( K[mxm] \) requires very large memory size and execution time. Reduced support vector machine (RSVM) [15] creates rectangular \( mxs \) kernel matrix of size \( s<<m \) by sampling, the small random datapoints \( S \) being a representative sample of the entire dataset (and RSVM uses it as a set of support vectors). RSVM reduces the problem size. RSVM has a good classification accuracy compared to standard SVM algorithms.

3. INCREMENTAL ALGORITHM OF THE FINITE STEPLESS NEWTON SVM

Although the finite stepless Newton SVM algorithm is fast and efficient to classify large datasets, it needs load whole dataset in the memory. Our investigation aims at scaling up this
algorithm to mine very large datasets on the PC (i.e. Pentium IV, 512 MB RAM). The main idea is to incrementally compute the gradient of \( f \) and the generalized Hessian of \( f \) at \( u \) for each iteration in the finite Newton algorithm described in the algorithm 1.

Suppose we have a very large dataset decomposed into small blocks by rows \( A_i, D_i \). The incremental algorithm of the finite stepless Newton SVM can simply incrementally compute the gradient and the generalized Hessian of \( f \) by the formulation (5) and (6).

Consequently, the incremental finite stepless Newton SVM algorithm can handle massive datasets on a PC. If the dimension of the input space is small enough (less than \( 10^3 \)), even if there are billions datapoints, the incremental finite stepless Newton SVM algorithm is able to classify them on a standard personal computer (Pentium IV, 512 MB RAM). The algorithm only needs to store a small \((n+1)\times(n+1)\) matrix and two \((n+1)\times 1\) vectors in memory between two successives steps.

The accuracy of the incremental algorithm is exactly the same as the original one.

Algorithm 2. The incremental algorithm of the finite stepless Newton SVM

4. PARALLEL AND DISTRIBUTED INCREMENTAL ALGORITHM OF THE FINITE STEPLESS NEWTON SVM

The incremental SVM algorithm described above are very fast to train in most of the cases and can deal with very large datasets on PCs. However it runs only on one single machine. With using the remote procedure calls (RPC) mechanism and the thread concept, we have extended it to build a parallel and distributed version on a computer network. The parallel and distributed algorithm profits from the PCs’ performance of a computer network. It speeds up data loading task and computational cost.

First, we distribute the datasets \( \{A_k, D_k\} \) on remote servers. For the \( i^{th} \) step, the remote servers compute independently, incrementally the sums of \( p_i = (-D_i H_j)^T(e - D_i H_j u_i) \) and \( q_i = (-D_i H_j)^T \text{diag}(e - D_i H_j u_i)(-D_i H_j) \). Then a client machine will use these sums to update \( u \) at the \( i^{th} \) iteration. The RPC protocol does not support asynchronous communication. A synchronous request-reply mechanism in RPC requires that the client and server are always available and
functioning (i.e. the client or server is not blocked). The client can issue a request and must wait for the server's response before continuing its own processing. Therefore, we have parallelized waitings on the client side with the set of threads.

The accuracy of the parallel and distributed incremental finite stepless Newton SVM algorithm is exactly the same as the original one.

The accuracy of the parallel and distributed incremental finite stepless Newton SVM algorithm is exactly the same as the original one.

5. NUMERICAL TEST RESULTS

To evaluate the performance of our parallel and distributed incremental finite stepless Newton SVM algorithm on very large datasets, we have implemented its on the PC running Linux Fedora Core 3. The program toolkit is written in C/C++. We have also used the high performance linear algebra library, Lapack++ [9] to benefit by the high speed of computational matrix. Thus, the software program is able to deal with large datasets in linear and non linear classification tasks. We focus on numerical tests with large datasets (cf. table 1) generated by the RingNorm program [6]. It is a 20 dimensional, 2 class classification example. Each class is drawn from a multivariate normal distribution. Class 1 has mean zero and covariance 4 times the identity. Class 2 (considered as -1) has mean (mean, mean, \ldots, mean) and unit covariance with mean = 2/sqrt(20). We use them to evaluate the execution time on PCs (3GHz Pentium IV, 512 MB RAM running Linux Fedora Core 3).
First, we have split the datasets into small blocks by rows (5000 datapoints). With a non linear classification task of the RingNorm dataset, [8] has found that SVM algorithms need about 250 support vectors to non linearly classify this dataset. Then, we have also tried to tune the number of support vectors, we could obtain the good results by only using 200 random datapoints being a representative sample of the entire dataset (as support vectors). The RBF kernel functions are constructed with the whole dataset and 200 random datapoints. The finite stepless Newton SVM algorithm is able to linearly and non-linearly classified datasets in the parallel and distributed incremental way on PCs.

<table>
<thead>
<tr>
<th></th>
<th>Data1</th>
<th>Data2</th>
<th>Data3</th>
<th>Data4</th>
<th>Data5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Training set</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$10^4$</td>
<td>$5.10^3$</td>
<td>$10^3$</td>
<td>$5.10^3$</td>
<td>$10^3$</td>
</tr>
<tr>
<td><strong>Testing set</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$10^4$</td>
<td>$5.10^3$</td>
<td>$10^3$</td>
<td>$5.10^3$</td>
<td>$10^3$</td>
</tr>
<tr>
<td><strong>Linear</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Training time (s)</td>
<td>0.025</td>
<td>0.13</td>
<td>0.255</td>
<td>1.293</td>
<td>2.585</td>
</tr>
<tr>
<td>Accuracy (%)</td>
<td>77.00%</td>
<td>76.42%</td>
<td>76.22%</td>
<td>76.26%</td>
<td>76.36%</td>
</tr>
<tr>
<td><strong>Non-linear</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Training time (s)</td>
<td>0.913</td>
<td>4.418</td>
<td>8.903</td>
<td>44.611</td>
<td>99.321</td>
</tr>
<tr>
<td>Accuracy (%)</td>
<td>98.90%</td>
<td>98.58%</td>
<td>98.67%</td>
<td>98.58%</td>
<td>98.57%</td>
</tr>
</tbody>
</table>

Table 1. The classification results with execution time reported on ten PCs

By varying the number of PCs, the size of datasets and the number of dimensions, we have measured computational time. Thus, the parallel and distributed incremental algorithm has linear dependences on the number of machines, size of datasets and a second order of the number of dimensions. Concerning the communication cost, they take about one second when the dataset dimension is less than 100. The algorithm has linearly classified one million datapoints with 20-dimensional input space on ten machines in 2.585 seconds as shown in table 1. Thus, we can estimate that one billion datapoints in 20-dimensional can be classified into two classes in 2585 seconds (43 minutes) on 10 PCs. The results obtained have presented the effectiveness of these new algorithms to deal with very large datasets on PCs with linear and non linear classification tasks.

6. CONCLUSION AND FUTURE WORK

We have presented a new SVM algorithm being able to deal with very large datasets in linear and non linear classification tasks on PCs. We have extended the recent finite stepless Newton SVM algorithm to build incremental, parallel and distributed SVM. The accuracy of the new algorithm is exactly the same as the original one but its complexity is linearly dependence on the number of machines, size of datasets and a second order of the number of dimensions. The algorithm also requires to store a $(n+1)\times(n+1)$ matrix and two $(n+1)\times 1$ vectors in memory (where $n$ is the number of dimensions).

We focus on numerical tests with large datasets generated by the RingNorm program. Our new algorithm can perform the classification of one million datapoints with 20-dimensional input space into two classes in 2.585 seconds on ten machines (3 GHz Pentium IV, 512 MB RAM, Linux Fedora Core 3). A new version of the algorithm is also implemented by XML-RPC [14] which can do the parallel and distributed operations over any XML-capable transport protocol, typically over HTTP. The software program could then be distributed on different kind of machines, for example on a set of various remote PCs, Unix stations or any other computer reachable via the web.
In general, a complex non-linear classification task needs a large number of support vectors. This requires a large number of random datapoints from the entire dataset for creating the rectangular kernel matrix at the input. The algorithm must work on a large number of dimensions. And thus it is impractical. A forthcoming improvement will be to combine our method with other machine learning algorithms to construct another approach that can deal with a complex non-linear classification task.

REFERENCES


[24] Tong, S., and Koller, D.: Support Vector Machine Active Learning with Applications to Text Classification. in proc. of ICML’00, the 17th Int. Conf. on Machine Learning, Stanford, USA, 2000, pp. 999-1006.