Incremental SVM and Visualization Tools for Biomedical Data Mining

Thanh-Nghi Do, François Poulet
ESIEA Recherche
38, rue des Docteurs Calmette et Guérin
Parc Universitaire de Laval-Changé
53000 Laval - France
{dothanh, poulet}@esiea-ouest.fr

Abstract. Most of the bio-data analysis problems process datasets with a very large number of attributes and few training data. This situation is usually suited for support vector machine (SVM) approaches. We have implemented a new column-incremental linear proximal SVM to deal with this problem. Without any feature selection step, the algorithm can deal with very large datasets (at least $10^9$ attributes) on standard personal computers. We have evaluated its performance on bio-medical datasets and compared the results with other SVM algorithms. Furthermore, we propose a new visualization tool to try to explain the results of automatic SVM algorithms by displaying the data distribution according to the distance to the separating plane.

1 Introduction

In recent years, data stored worldwide has doubled every 20 months, so that the need to extract knowledge from very large databases is increasing. Knowledge Discovery in Databases (KDD) has been defined as the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data. Data mining can be defined as one particular step of the KDD process: the identification of interesting structures in data. It uses different algorithms for classification, regression, clustering or association rules. These new progresses in data mining have already been applied to bio-data analysis (Zaki et al., 2002). SVM algorithms (Vapnik, 1995) are one of the most well-known of a class of performing methods for bio-data analysis. Recently, a number of powerful SVM learning algorithms have been proposed (Bennett et al., 2000). SVM uses the idea of kernel substitution for classifying the data in a high dimensional feature space. They have been successfully applied to various applications, for example in face identification, text categorization, bioinformatics (Guyon, 1999). Especially, SVM performs particularly well in bio-data analysis problems (Mukherjee et al., 1999, Brown et al., 1999, Furey et al., 2000). However, SVM solution is obtained from quadratic programming problem possessing a global solution, so that, the computational cost of a SVM approach depends on the optimization algorithm used. The very best algorithms today are typically quadratic and require multiple
The new proximal SVM algorithm proposed by (Fung et al., 2001) changes the inequality constraints to equalities in the optimization problem, thus the training task requires the solution of a system of linear equations, so that PSVM is very fast to train. Furthermore, PSVM can construct incrementally the model without loading the whole dataset in main memory. The Sherman-Morrison-Woodbury formula (Golub et al., 1996) is used to adapt PSVM to a row-incremental or column-incremental version (but not both yet). Thus the algorithm can deal with large datasets (at least $10^9$ in one dimension: row or column) on standard personal computers (with standard RAM and disk capacities). The algorithm has been used on bio-medical datasets, the results are compared with some others in terms of learning time and classification accuracy. We have also developed a visualization tools to try to explain the SVM results. The display of the datapoint distribution according to the distance to the separating plane helps us to verify the robustness of obtained models.

We summarize the content of the paper now. In section 2, we introduce very general SVM classifier. In section 3, we describe PSVM and its incremental versions. We present the graphical method for visualizing the SVM results in section 4. Some numerical test results on bio-medical datasets can be found in section 5 before the conclusion in section 6.

Some notations will be used in this paper: all vectors are column vectors, the 2-norm of the vector $x$ is denoted by $\|x\|$, the matrix $A[m\times n]$ is the $m$ training points in the $n$-dimensional input space $\mathbb{R}^n$. The diagonal matrix $D[n\times n]$ of $\pm 1$ contains the classes $+1$ and $-1$ of $m$ training points. $e$ is the column vector of $1$. $w, b$ are the coefficients and the scalar of the hyperplane. $\xi$ is the slack variable and $\nu$ is a positive constant. The identity matrix is denoted $I$.

## 2 Support Vector Machines

![Fig. 1. Linear separation of the datapoints into two classes](image)

Let us consider a linear separation classification task, as depicted in figure 1, with $m$ data points in the $n$-dimensional input space $\mathbb{R}^n$, represented by the $m\times n$ matrix $A$, having corresponding labels $\pm 1$, denoted by the $m\times m$ diagonal matrix $D$ of $\pm 1$. 

Let $x^T w - b = 0$ be the separating line. The distance from the separating line to a point $x$ is $d(x) = \frac{|x^T w - b|}{||w||}$. The maximum margin is $\frac{2}{||w||}$.
For this problem, the SVM 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 support planes for each class \((x^Tw - b = +1\) for class +1, \(x^Tw - b = -1\) for class -1). The margin between these supporting planes is \(2/\|w\|\). Any point falling on the wrong side of its supporting plane is considered to be an error. Therefore, the SVM has to simultaneously maximize the margin and minimize the error. The standard SVM solution with linear kernel is given by the following quadratic program (1):

\[
\min f(z,w,b) = \nu e^Tz + (1/2)\|w\|^2
\]

s.t. \(D(Aw - eb) + z \geq e\)

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

The plane \((w,b)\) is obtained by the solution of the quadratic program (1). And then, the classification function of a new data point \(x\) based on the plane is:

\[
f(x) = \text{sign}(w.x - b)
\]

(2)

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 the objective function instead of the original one. The details can be found in (Bennett et al. 2000) and (Cristianini et al., 2000).

### 3 Linear Proximal Support Vector Machines

The proximal SVM classifier proposed by Fung and Mangasarian changes the inequality constraints to equalities in the optimization problem (1) and adding a least squares 2-norm error into the objective function \(f\), it changes the formulation of the margin maximization to the minimization of \((1/2)\|w,b\|^2\). Thus substituting for \(z\) from the constraint in terms \((w,b)\) into the objective function \(f\) we get an unconstraint problem (3):

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

(3)

The Karush-Kuhn-Tucker optimality condition of (3) will give the linear equation system (4) of \((n+1)\) variables \((w,b)\):

\[\begin{bmatrix}
w_1 \\
w_2 \\
.. \\
w_n \\
b
\end{bmatrix} = \begin{bmatrix}(I/\nu + E^TE)^{-1}E^T \\
A \\
-\text{e}
\end{bmatrix}
\]

(4)

where \(E = [A \\
-\text{e}]\)

Therefore, the linear PSVM is very fast to train because it expresses the training in terms of solving a set of linear equations of \((w,b)\) instead of quadratic programming.
The accuracy can be compared with standard SVM. Note that all we need to store in memory is the \((m)\times(n+1)\) training data matrix \(E\), the \((n+1)\times(n+1)\) matrix \(E^T E\) and the \((n+1)\times1\) vector \(d = E^T D e\). If the dimensional input space is small enough (less than \(10^3\)), even if there are millions of datapoints, PSVM is able to classify them on a standard personal computer. For example, the linear PSVM can easily handle large datasets as shown by the classification of 2 million 10-dimensional points in 15 seconds on a Pentium-4 (2.4GHz, 256 Mb RAM, Linux).

The algorithm is limited by the storage capacity of the \((m)\times(n+1)\) training data matrix \(E\). In order to deal with very large (at least one billion points) datasets, the incremental version is extended from the computation of \(E^T E\) and \(d = E^T D e\).

### 3.1 Row-incremental PSVM (Fung et al., 2002)

We can split the training dataset \(E\) into blocks of lines \(E_i\), \(D_i\) and compute \(E^T E\) and \(d = E^T D e\) from these blocks:

\[
E^T E = \sum E_i^T E_i, \quad d = \sum E_i^T D_i e
\]

For each step, we only need to load the (blocksize)\times(n+1) matrix \(E_i\) and the (block-size)\times1 vector \(D_i e\) for computing \(E^T E\) and \(d = E^T D e\). Between two incremental steps, we need to store in memory \((n+1)\times(n+1)\) and \((n+1)\times1\) matrices although the order of the dataset is one billion data points. The authors have performed the linear classification of one billion data points in 10-dimensional input space into two classes in less than 2 hours and 26 minutes on a Pentium II (400 MHz, 2 GB RAM, 30% of the time being spent to read data from disk).

### 3.2 Column-incremental PSVM

The algorithm described in the previous section can handle datasets with a very large number of datapoints and small number of attributes. But some applications (like bioinformatics or text mining) require datasets with a very large number of attributes and few training data. Thus, the \((n+1)\times(n+1)\) matrix \(E^T E\) is too large and the solution of the linear equation system of \((n+1)\) variables \((w, b)\) has a high computational cost. To adapt the algorithm to this problem, we have applied the Sherman-Morrison-Woodbury formula to the linear equation system (4) to obtain:

\[
[w_1, w_2, \ldots, w_n, b]^T = (I/v + E^T E)^{-1} E^T D e = v E^T[D e - (I/v + EE^T)^{-1} EE^T D e] \quad (5)
\]

where \(E = [A -e]\)

The solution of (5) depends on the inversion of the \((m)\times(m)\) matrix \((I/v + EE^T)\) instead of the \((n+1)\times(n+1)\) matrix \((I/v + E^T E)\) in (4). The cost of storage and computation depends on the number of training data. This formulation can handle datasets with very large number of attributes and few training data.
We have imitated the row-incremental algorithm for constructing the column-incremental algorithm able to deal with very large number of dimensions. The data are split in blocks of columns $E_i$ and then we perform the incremental computation of $EE^T = \sum E_iE_i^T$. For each step, we only need to load the $(m)\times(block\,size)$ matrix $E_i$ for computing $EE^T$. Between two incremental steps, we need to store in memory the $(m)\times(m)$ matrix $EE^T$ although the order of the dimensional input space is very high.

With these two formulations of the linear incremental PSVM, we are able to deal with very large datasets (large either in training data or number of attributes, but not yet both simultaneously). We have used them to classify bio-medical datasets with interesting results in terms of learning time and classification accuracy.

4 Visualization of Linear SVM Results

Although the SVM algorithms have been successfully applied to a number of applications, they provide limited information. Most of the time, the user only knows the classification accuracy and the hyperplane equation. It is difficult to really explain or verify the constructed model or to understand why the SVM algorithms are more efficient than the other ones. Visualization techniques can be used to improve the results comprehensibility. We have developed a new method that can help the user to understand the model constructed by the SVM algorithm.

While the classification task is processed (based on the hyperplane obtained), we also compute the datapoint distribution according to the distance to the hyperplane as shown in figure 2.

![Fig. 2. Datapoints distribution based on parallel planes](image_url)

For each class, the positive distribution is the set of correctly classified data points, and the negative distribution is the set of misclassified data points. An example of such a distribution is shown in the right part of the figure 2. The method is applied to Lung Cancer dataset (2 classes, 32 training, 149 testing, 12533 numerical attributes) as shown in figure 3.
The visualization of the datapoint distribution according to the distance to the separating plane helps us to evaluate the robustness of obtained models by SVM algorithm.

5 Numerical Test Results

The software program is written in C/C++ on SGI-O2 workstation (IRIX) and PC (Linux). To validate the performances of the incremental algorithms, we have classified bio-medical datasets (Jinyan et al., 2002) and the Thrombin drug design dataset from the KDD cup’01 (Hatzis et al., 2001). These datasets are described in table 1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>classes</th>
<th>datapoints</th>
<th>attributes</th>
<th>evaluation method</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL-AML Leukemia</td>
<td>2</td>
<td>72</td>
<td>7129</td>
<td>38 Tr – 34 Tst</td>
</tr>
<tr>
<td>Breast Cancer</td>
<td>2</td>
<td>97</td>
<td>24481</td>
<td>78 Tr – 19 Tst</td>
</tr>
<tr>
<td>Central Nervous System</td>
<td>2</td>
<td>60</td>
<td>7129</td>
<td>leave-one-out</td>
</tr>
<tr>
<td>Colon Tumor</td>
<td>2</td>
<td>62</td>
<td>2000</td>
<td>leave-one-out</td>
</tr>
<tr>
<td>MLL Leukemia</td>
<td>3</td>
<td>72</td>
<td>12582</td>
<td>57 Tr – 15 Tst</td>
</tr>
<tr>
<td>Ovarian Cancer</td>
<td>2</td>
<td>253</td>
<td>15154</td>
<td>leave-one-out</td>
</tr>
<tr>
<td>Prostate Cancer</td>
<td>2</td>
<td>136</td>
<td>12600</td>
<td>102 Tr – 34 Tst</td>
</tr>
<tr>
<td>Subtypes of Acute Lymphoblastic Leukemia</td>
<td>7</td>
<td>327</td>
<td>12558</td>
<td>215 Tr – 112 Tst</td>
</tr>
<tr>
<td>Lung Cancer</td>
<td>2</td>
<td>181</td>
<td>12533</td>
<td>32 Tr – 149 Tst</td>
</tr>
<tr>
<td>Translation Initiation Sites</td>
<td>2</td>
<td>13375</td>
<td>927</td>
<td>10-fold</td>
</tr>
<tr>
<td>Thrombin Drug Design</td>
<td>2</td>
<td>2543</td>
<td>139351</td>
<td>1909 Tr – 634 Tst</td>
</tr>
</tbody>
</table>

Thus, we have obtained the results (concerning the training time and accuracy) shown in table 2 on a personal computer Pentium-4 (2.4GHz, 256 Mb RAM, Linux Redhat 7.2). Without any feature selection, almost all datasets are classified by the column-incremental linear PSVM with one exception Translation Initiation Sites is
classified by the row-incremental and the column-incremental (*) versions of the linear PSVM. The one-against-all approach has been used to classify multi-class datasets (more than 2 classes), thus we have taken the average accuracy. The results are compared with the linear kernel of SVMLight (Joachims, 2002).

Table 2. Numerical test results.

<table>
<thead>
<tr>
<th></th>
<th>Learning time (secs)</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IPSVM</td>
<td>SVMLight</td>
</tr>
<tr>
<td>ALL-AML Leukemia</td>
<td>0.64</td>
<td>9.14</td>
</tr>
<tr>
<td>Breast Cancer</td>
<td>4.43</td>
<td>269.66</td>
</tr>
<tr>
<td>Central Nervous System</td>
<td>0.68</td>
<td>15.29</td>
</tr>
<tr>
<td>Colon Tumor</td>
<td>0.41</td>
<td>1.80</td>
</tr>
<tr>
<td>MLL Leukemia</td>
<td>3.42</td>
<td>133.68</td>
</tr>
<tr>
<td>Ovarian Cancer</td>
<td>12.51</td>
<td>403.60</td>
</tr>
<tr>
<td>Prostate Cancer</td>
<td>2.73</td>
<td>61.97</td>
</tr>
<tr>
<td>SAL Leukemia</td>
<td>8.26</td>
<td>42.10</td>
</tr>
<tr>
<td>Lung Cancer</td>
<td>1.25</td>
<td>20.80</td>
</tr>
<tr>
<td>Translation Initiation Sites</td>
<td>63 (4238*)</td>
<td>314</td>
</tr>
</tbody>
</table>

As we can see in table 2, the incremental version of linear PSVM is generally outperforming SVMLight on learning time and classification correctness. The results shown in table 2 indicate that the column-incremental PSVM can be compared with standard SVM in terms of classification accuracy. But its training time is always better (from 4 to 60 times) than SVMLight one. The only one exception is the Thrombin Drug Design: this dataset has very few null values (0.6823 %) and then SVMLight is faster than our column-incremental PSVM because we did not use a compressed data representation like SVMLight. The other algorithms can not run even with 1 GB RAM. They usually have to use a pre-processing task. Our column-incremental PSVM has given 79.02 % accuracy without any feature selection. The results concerning the Translation Initiation Sites show the interest of being able to choose between the column or the row incremental versions of the PSVM. The time needed for the classification task is divided by 70 (with exactly the same accuracy) when using the most appropriate version of the incremental algorithms.

6 Conclusion and Future Work

We have presented a new implementation of incremental linear PSVM which can deal with large datasets (at least $10^9$ in one dimension: row or column if the other one stays under $10^4$) on standard personal computers without any pre-processing task. The Sherman-Morrison-Woodbury formula is used to adapt PSVM to row-incremental or column-incremental. The algorithm has been estimated on bio-medical datasets. The column-incremental is a very convenient way to handle bio-medical datasets because
it avoids loading the whole dataset in main memory and is very fast to train datasets with a very high number of attributes and few training data. Its performance concerning training time and classification accuracy can be compared with standard SVM.

We have also proposed a way to visualize SVM results. The datapoints distribution according to the distance to the separating plane helps us to estimate the robustness of models obtained by SVM algorithms. A forthcoming improvement will be to extend these algorithms to the non-linear kernel cases. Another one will be to combine our method with other graphical techniques (Fayyad et al., 2001) to construct another kind of cooperation between SVM and visualization tools for applying it to bio-medical data analysis.

References

6. Fung, G., Mangasarian O.: Proximal Support Vector Machine Classifiers. in proc. of the 7th ACM SIGKDD, Int. Conf. on KDD’01, San Francisco, USA (2001) 77-86
7. Fung, G., Mangasarian O.: Incremental Support Vector Machine Classification. in proc. of the 2nd SIAM Int. Conf. on Data Mining SDM’02 Arlington, Virginia, USA (2002) 247-260