Data Condensation in Large Databases by Incremental Learning with Support Vector Machines

Pabitra Mitra, C. A. Murthy and Sankar K. Pal
Machine Intelligence Unit, Indian Statistical Institute
203 B. T. Road, Calcutta 700 035, India *

Abstract

An algorithm for data condensation using support vector machines (SVM's) is presented. The algorithm extracts data points lying close to the class boundaries, which form a much reduced but critical set for classification. The problem of large memory requirements for training SVM's in batch mode is circumvented by adopting an active incremental learning algorithm. The learning strategy is motivated from the condensed nearest neighbor classification technique. Experimental results presented show that such active incremental learning enjoy superiority in terms of computation time and condensation ratio, over related methods.

Keywords: Data Condensation, Support Vector Machines, Incremental Learning, Active Learning, Data Mining

1 Introduction

The current popularity of data mining, data warehousing has led to proliferation of terabyte data warehouses [4]. Mining a database of even a few gigabytes is an arduous task for machine learning algorithms, and requires advanced parallel hardware and algorithms. A popular approach for dealing with the intractability problem of learning from huge databases is to select a small subset for training. Databases often contain redundant data. It would be convenient if large databases could be replaced by only a small subset of informative patterns. The accuracy of a classifier trained on such reduced data set should be comparable to results obtained from training with the entire data set.

Various statistical sampling methods such as random sampling, stratified sampling [3] have been developed which often help in speed-up of classifier design. However, naive sampling methods are not suitable for real world problems with noisy data, where classification results can degrade unpredictably and significantly. Most of these sampling methods select data in a model independent fashion, in a sense that no implicit classifier is used. Another class of selection techniques use a specific classifier model for data selection. One of the earliest such algorithms were the data condensation algorithms proposed by Hart [5] for designing k-nn classifiers. Several variants of the algorithm have been developed and used for effective data condensation. There exists another class of *boundary hunting* data condensation methods, which seek out training examples near class boundaries because these examples tend to be useful for locating class boundaries precisely. Assuming that only a small fraction of training examples in a database lie along class boundaries, these boundary hunting methods can be desirable for selecting small but highly informative training data subsets.

Support Vector Machines (SVMs) are a recently developed class of classifier architectures, derived from the *structural risk minimization* principle suggested by Vapnik [2]. A promising aspect of the SVM classifiers is that, SVM training involves selecting a small subset of critical data points (known as *support vectors*) from the original training data base. This critical data subset fully and succinctly defines the classification task at hand.

SVMs however, suffer from the problem of large memory requirement and CPU time when trained in batch mode. The training process involves the solution of a non-sparse quadratic programming (QP) problem. Several strategies have been suggested to circumvent this problem. An important such technique is the "decomposition method" suggested by Osuna *et al* [6]. In decomposition method the Hessian matrix involved in the QP problem is iteratively split into an active and a redundant set; thereby reducing the computational complexity.

A simple strategy of incremental learning may also help to overcome the huge memory requirements of batch mode SVM design. The trick is to partition a huge database into small subsets and incrementally train SVM classifier with the subsets. The training would preserve only the support vectors at each incremental step, and add them to the training set for the next step. It has been experimentally observed that the model obtained by this method is similar to

*EMail: {pabitra_r,murthy,sankar}@isical.ac.in

what would have been obtained by using all the data together to train [7].

We suggest a generalization of the above strategy, for incrementally extracting the support vectors. The SVM is incremented using small samples drawn from the entire database. However, the resampling is done in an active manner, in the sense that the classifier obtained at the current iteration is implicitly used. We present experimental results to show that such resampling strategies are computationally superior to random partitioning of the data base, while achieving comparable accuracy and smaller condensed sets.

2 Support Vector Machines

Support Vector Machines are a general class of learning architecture inspired from Statistical Learning Theory that perform *structural risk minimisation* on a nested set structure of separating hyperplanes. Given a training data, the SVM training algorithm obtains the optimal separating hyperplane in terms of generalisation error.

The Support Vector Algorithm. Suppose we are given a set of examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l) \in R^N . y_i \in \{-1, +1\}$. We consider functions of the form $sgn((\mathbf{w} \cdot \mathbf{x}) + b)$, in addition we impose the condition

$$\inf_{i=1,\ldots,l} |(\mathbf{w} \cdot \mathbf{x}_i) + b| = 1. \tag{1}$$

We would like to find a decision function $f_{\mathbf{w},b}$ with the properties $f_{\mathbf{w},b}(x_i) = y_i \ ; \ i = 1, \ldots, l$. If this function exists, condition (1) implies

$$y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) \geq 1, \quad i = 1, \ldots, l. \tag{2}$$

In many practical situations, a separating hyperplane does not exist. To allow for possibilities violating Equation 2, slack variables are introduced

$$\xi_i \geq 0, \text{ to get } y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) \geq 1 - \xi_i, \ i = 1, ..., l. \tag{3}$$

The support vector approach to minimise the generalisation error consists of the following:

$$\textbf{Minimize}: \qquad \Phi(\mathbf{w}, \xi) = (\mathbf{w} \cdot \mathbf{w}) + \gamma \sum_{i=1}^{l} \xi_i \tag{4}$$

subject to the constraints (3).

It can be shown that minimising the first term in Equation 4, amounts to minimising the VC-dimension, and minimising the second term corresponds to minimising the missclassfication error [2]. The minimisation problem can be posed as a constrained quadratic programming (QP) problem. The solution gives rise to a decision function of the form :

$$f(\mathbf{x}) = sgn \left[\sum_{i=1}^{l} y_i \alpha_i (\mathbf{x} \cdot \mathbf{x_i}) + b \right].$$

Only a small fraction of the α_i coefficients are non-zero. The corresponding pairs of $\mathbf{x_i}$ entries are known as *support vectors* and fully define the decision function. The support vectors are geometrically the points lying near the class boundaries. We use linear kernels for SVM, nonlinear kernels may also be used. However, it has been found that support vectors are almost independent of the kernel used [8], hence choice of linear kernel may suffice for most datasets.

3 Data Condensation Algorithm

Our Algorithm is motivated from the condensation technique proposed by Hart [5] for reducing the computational complexity and storage requirements of k-nn classifiers.

3.1 Condensed Nearest Neighbor Technique (CNN)

The objective of the condensed nearest neighborhood rule is to select one minimal subset such that the k-NNR with the selected subset would correctly classify the remaining points in the sample set. The following algorithm produces a subset with the above properties, except for the fact that it is not guaranteed to be minimal.

Algorithm

Set up bins called STORE and GRABBAG. k randomly selected samples are placed in STORE, all other samples are placed in GRABBAG. Let n_g denote the current number of samples in GRABBAG.

Step 1: Use k-NNR with the current contents of STORE to classify the ith sample of GRABBAG. If correctly classified the sample is returned to GRABBAG, otherwise it is placed in STORE. Repeat this operation for $i = 1, \ldots, n_g$.

Step 2: If one complete pass is made through Step 1 with no transfer from GRABBAG to STORE, or GRABBAG is exhausted, then terminate. Else go to Step 1.

The final contents of STORE constitute the condensed subset to be used with k-NNR. The contents of GRABBAG are discarded.

3.2 Proposed Algorithm

Our method is similar to the CNN technique. However, instead of the k-NNR we use SVM classifiers. Also instead of classifying all the points in GRABBAG, we incrementally use only small number of points from GRABBAG to update STORE.

Algorithm

We place a small number (k) of samples in STORE and the remaining samples of the training set in GRABBAG.

Step 1: Design a SVM using the samples in STORE. Retain the *support vectors* in STORE, discard other points.

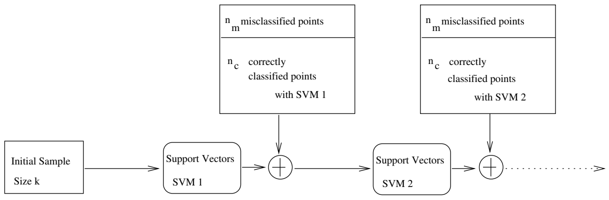

Figure 1. Schematic of the proposed algorithm

Step 2: Resample GRABBAG. Select n_c and n_m points, correctly classified and misclassified respectively using the SVM obtained in Step 1, from GRABBAG. Append the (n_c+n_m) resampled points to STORE obtained after Step 1. Repeat Step 1, till the required accuracy is achieved on a test set, or GRABBAG is exhausted.

The algorithm is illustrated in Figure 1.

4 Results

The database is sampled to obtain a moderately sized training set, such that the training set is a fair sample of the original database, yet feasible in terms of memory requirement. A smaller test set is also obtained by sampling the database. Since the database is likely to be large accuracy on the test set is likely to be a fair measure of the generalisation performance of the classifier. As a measure of efficiency of the condensation algorithms we study three quantities : the accuracy of classifier on the test set (in terms of error percentage), size of the condensed subset (as fraction of the training set), and CPU time required (on a Pentium 200 MHz processor) by the condensation algorithm.

We use two relatively large datasets and three smaller datasets for our experiments. The datasets are described below:

twonorm: This is a 20 dimension, 2 class data. Each class is drawn from a unit multivariate normal distribution with unit covariance matrix. Class 1 has mean $(a, a, ...a)$ and Class 2 has mean $(-a, -a, ..., -a)$. $a = 2/(20)^{1/2}$.

ringnorm: This is a 20 dimension, 2 class data. Class 1 is multivariate normal with mean zero and covariance matrix 4 times identity. Class 2 has unit covariance matrix and mean $(a, a, ..., a)$. $a = 1/(20)^{1/2}$.

Both the above datasets are widely used by Breiman [1]. Breiman reports a test set error of 7.4% and 11.0% respectively for the datasets. We also use three other smaller real life datasets for our experiments, namely - Wisconsin breast cancer data, Monks-3 dataset and Hearts dataset. All the datasets are available at UCI Machine learning repository.

We present results of three related condensation techniques for comparison in Table 5.

1. The classical condensed nearest neigbor (CNN) algorithm of Hart [5].

2. Incremental SVM with passive resampling, i.e. the training set is randomly resampled at each iteration [7].

3. Incremental SVM with active resampling. The training set is sampled to obtain n_c and n_m correctly classified and misclassified points respectively using the classifier obtained at current iteration.

Following conclusions can be drawn from Table 5:

Condensed nearest neighbor technique is computationally less demanding but achieves poorer condensation.

SVM when designed in *batch mode* gives best accuracy and condensation, but it has too high computational requirements.

Incremental SVM design with passive resampling performs better than condensed k-NN (CNN), in terms of condensation but requires greater CPU time.

Incremental SVM design with active resampling performs better than CNN in terms of condensation ratio and accuracy and requires comparable CPU time. Compared with batch mode SVM and incremental SVM with passive resampling, it has comparable condensation ratio but requires less CPU time than both batch mode design and passive resampling. The gain in time is even more in case of datasets having overlaps (*i.e* larger bayes error) as can be seen from the comparisons in case of the *ringnorm* data.

An important consideration with regards to the active resampling technique is the choice of number of correctly classified (n_c) and misclassified points (n_m) to be sampled. It was experimentally found that our algorithm works for almost any value of n_c and n_m if the amount of overlap is small. However, for datasets having substantial overlap, the choice of n_c and n_m influences the convergence. It has been observed that choosing the ratio $n_c/n_m \leq 1$ leads to oscillation of the error convergence curve. On the other hand choosing a large value of n_c/n_m leads to better convergence but more CPU time. Whether there exists any value of n_c/n_m corresponding to convergence in minimal time is not clear. In principle, for larger overlap, higher n_c/n_m ratio gives better results. If nothing is known about the overlap, we choose $n_c/n_m = 1$, which performs atleast better than the passive resampling. This is illustrated for the data *twonorm* in Figure 2, for different ratios of n_c/n_m. In Figure 2, it is seen that 7:1 ratio is better than 1:1 for the *twonorm* data. In general it need not be true for other datasets.

Another important observation concerns the asymptotic nature of the condensed subset. We have found out that the condensed set of points obtained by our learning strategy converges to an almost fixed set. To illustrate this point we plot the Hausdorff distance between the condensed sets of two successive iterations in Figure 3 (for the *twonorm* data). It can be seen that the metric converges to a small value with

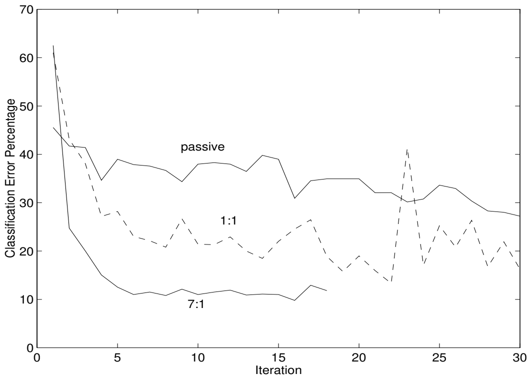

Figure 2. Convergence of the error rate for different values of $n_c : n_m$ for *twonorm* data.

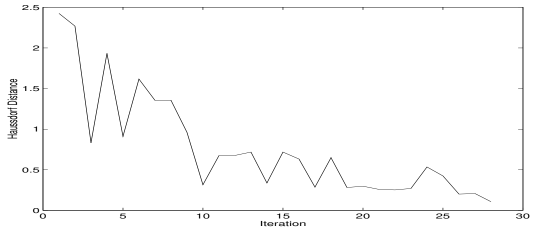

Figure 3. Plot of the Hausdorff distance between condensed subsets of two succesive iterations against iterations, for the *twonorm* data.

iterations.

5 Conclusions and Discussion

We have proposed an algorithm for extracting a reduced but critical subset of points from a large database using Support Vector Machine. To circumvent the computational complexity and large memory requirements of training SVM in batch mode, we proposed an incremental learning algorithm inspired from the condensed nearest neighborhood technique.

Results are presented to show that actively selecting the examples to update the classifier at each iteration leads to faster convergence, with almost comparable level of accuracy and condensation ratio. However, whether any optimal active resampling strategy is present or not is not very clear, and seems to be somewhat dependent on the amount of overlap present.

It may be noted that our method is similar in spirit to Adaptive Resampling and Combining (ARCing) classifiers [1] developed in the framework of PAC learning. The difference being, in all the aforesaid methods the classifiers obtained at each iterations are combined to obtain the final classifier, our algorithm does not have the combining step. Also the data samples used for designing the classifiers in the above methods, do not converge to some fixed set. Such convergence is however necessary for our purpose. The sampling strategy used by us is shown to converge to an almost fixed subset of critical points, which ensures robustness of the incremental learning strategy.

References

[1] L. Breiman. Arcing classifiers. *Annals of Statistics*, 26:801–849, 1998.

[2] C. J. C. Burges. A tutorial on support vector machines. *Data Mining and Knowledge Discovery*, 2, 1998.

[3] J. Catlett. *Megainduction: machine learning on very large databases*. PhD thesis, Department of Computer Science, University of Sydney, Australia, 1991.

[4] U. Fayyad and R. Uthurusamy. Data mining and knowledge discovery in databases. *Communications of the ACM*, 39(11):24–27, Nov. 1996.

[5] P. E. Hart. The condensed nearest neigbor rule. *IEEE Trans. Information Theory*, 14:515–516, 1968.

[6] E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In *IEEE NNSP'97*. IEEE, 1997.

[7] N. A. Sayeed, H. Liu, and K. K. Sung. Handling concept drifts in incremental learning with support vector machines. In *Proceedings of 1st Intl. Conf. on Knowledge Discovery and Data Mining*, pages 317–321. AAAI, 1999.

[8] B. Scholkopf, C. Burges, and V. Vapnik. Extracting support data for a given task. In *Proceedings of 1st Intl. Conf. on Knowledge Discovery and Data Mining*, pages 252–257. AAAI, 1995.

Table 1. Comparison of the performances of the condensing techniques

Dataset	Condensed k-NN			SVM (Batch Mode)		
	Error on Test Set (%)	Size of condensed set as (%)	CPU time (sec)	Error on Test Set (%)	Size of condensed set as (%)	CPU time (sec)
twonorm	9.78	16.46	16.15	10.02	2.31	204.51
ringnorm	17.59	21.95	20.40	23.95	2.34	216.50
cancer	3.19	11.88	0.82	2.59	1.97	117.02
heart	5.3	22.52	8.92	23.02	2.87	91.00
monks-3	1.2	14.21	5.11	1.09	1.45	179.21

Dataset	Incremental SVM with Passive Resampling			Incremental SVM with Active Resampling		
	Error on Test Set (%)	Size of condensed set as (%)	CPU time (sec)	Error on Test Set (%)	Size of condensed set as (%)	CPU time (sec)
twonorm	10.87	3.03	36.41	10.87	2.52	15.66
ringnorm	24.02	2.87	110.97	24.02	2.52	21.01
cancer	2.59	2.17	32.41	2.59	2.32	11.09
heart	23.95	3.0	24.02	24.17	2.87	14.19
monks-3	1.40	1.87	41.09	1.40	1.79	8.92