Handling Concept Drifts in Incremental Learning with Support Vector Machines

Nadeem Ahmed Syed **Huan Liu** **Kah Kay Sung**
Program for Research in Intelligent Systems (PRIS)
School of Computing
National University of Singapore
Singapore 119260
{nadeem, liuh, sungkk}@comp.nus.edu.sg

Abstract

With the increase in the size of real-world databases, there is an ever-increasing need to scale up inductive learning algorithms. Incremental learning techniques are one possible solution to the scalability problem. In this paper, we propose *three criteria* to evaluate the robustness and reliability of incremental learning methods, and use them to study the robustness of an incremental training method for Support Vector Machines. We provide empirical results using benchmark machine learning datasets to show that support vectors form a *succinct* and *sufficient* set for block-by-block incremental learning.

1 Introduction

When developing classifiers using learning methods, more training data can reduce *risk*, but the learning process can get computationally intractable. This issue is becoming more evident with large amounts of data available [5]. Researchers in machine learning and data mining [2, 3, 7] have therefore been trying to scale up classical inductive learning algorithms to handle extremely large data sets. Ideally, it is desirable to consider all the examples together for the best learning performance. However, when the training set is large, not all data can be loaded into the memory. One approach to overcoming this constraint is to train using some incremental learning technique, whereby only a subset of the data is used at any one time.

The aim of this paper is to : **(1)** define *two* broad approaches to incremental learning, **(2)** propose *three criteria* for evaluating the robustness of incremental learning algorithms, **(3)** based on previous observations and claims, suggest that support vector machines (SVM) should handle incremental form of learning, and **(4)** empirically verify the validity of our suggestion, and evaluate the robustness of incremental training method for SVM, based on our three criteria.

In the next section we define two incremental learning approaches, and criteria for evaluating the incremental algorithms. Section 3 provides empirical evidence that support vectors (SVs) form a sufficient set for incremental learning. Section 4 is about the succinctness of the SV representation. Finally we conclude by summing up our findings and their implications.

2 Goodness Measures

In general, incremental learning approaches are needed in the following scenarios : [**S1**] the example generation itself is time-dependent, e.g. time series data; [**S2**] new data is obtained in chunks at intervals, e.g. scientific research data; or [**S3**] the training data is too large to be loaded into the main memory. Given the various types of learning problems incremental methods can help to solve, there are broadly two approaches to incremental learning, viz, [**A1**] use one example at a time, lets call it instance by instance learning (*IIL*). [**A2**] use a suitable-size subset of examples at a time. We can call this block by block learning (*BBL*). Various methods have been reported in the literature on incremental learning, mostly in the context of first two scenarios [1, 4, 8, 9, 10]. A general criterion to evaluate the quality of incremental learning methods is to consider the prediction accuracy on some benchmark data sets. Schlimmer and Granger [9] also propose three criteria: **1.** the *number of observations (instances)* needed to obtain a 'stable' concept description, **2.** the *cost of updating memory*, and **3.** the *quality of learned concept descriptions* to measure the usefulness and effectiveness of an incremental learning method. These measures are useful in evaluating the quality of the online algorithms which are trained using time-based training vectors.

When incrementally training a classifier from a large database, some new questions have to be answered, such as: [**Q1**] How much better is a learned model at step $n + i$ than another model obtained after step n? **[Q2]** Can an incremental algorithm recover in the next incremental step or steps, if it goes drastically off the 'actual' concept at any stage? These questions are important in quantifying the robustness and reliability of incremental algorithms. When the data is not available beforehand, we do not have any choice but to do incremental training. On the other hand when we have the data available, we have a choice between a batch method (even if its expensive, the choice is there) and the incremental method, and so answers to the above questions can help us make those decisions.

When an inductive learning algorithm has to be trained incrementally, with each new step, the sample distribution of a training subset may vary. This in turn will create *perceived drift* in the underlying structure of the data space, or *virtual concept drift*[1] [12]. The challenge then is to ensure that the concept learned in the current step is incorporated into the concept learned in the previous step, so that actual structure of the data space is realized, preserved, and represented in the trained model. This may also be necessary to get a good generalization on the unseen data. So it appears that a very important measure of the effectiveness of any incremental learning method is its ability to capture the *concept drift* as the training progresses.

For effective incremental learning, we propose that any method that attempts to handle concept drift, should satisfy the following criteria : **[C1]** *Stability* - the prediction accuracy on the test set should not vary wildly at every incremental learning step; **[C2]** *Improvement* - there should be improvement in this prediction accuracy as the training progresses and the learning algorithm sees more training examples; and **[C3]** *Recoverability* - the learning method should be able to recover from its errors, i.e., even if the performance drops at a certain learning step, the algorithm should be capable of recovering and improving back to or better than the previous best performance.

3 Incremental Learning using SVMs

Recently, support vector machines (SVMs) [11] have been rapidly gaining in popularity as a learning paradigm. SVMs work by mapping the input vectors into a high dimensional feature space, and constructing the *Optimal Separating Hyper-plane* through *Structural Risk Minimization* [11]. Only the examples which are sufficient to define the Optimal Separating Hyper-plane (so called support vectors) are stored for doing future classification. Given that only a small fraction of training examples end up as SVs [11], the SVM algorithm can summarize the data space concisely.

[1]The *real concept drift* takes place when the underlying structure of the data space or the underlying concept changes in reality. For details see [12].

This suggests that we could partition a huge database such that each partition fits into the main memory, and then incrementally train an SVM with the partitions. The training would preserve only the SVs at each step, and add them to the training set for the next step. The model obtained by this method should be the same or similar to what would have been obtained using all the data together to train. The reason for this is that the SVM algorithm will preserve the essential class boundary information seen so far in a very compact form as SVs, which would contribute appropriately to deriving the new concept in the next iteration. Further, it is tempting to believe that such a method should also satisfy the three criteria of stability, improvement, and recoverability, since the overall structure of the data space will be preserved, and refined with more examples at each step.

4 Concept Drift

We train an SVM incrementally, using one partition of data at each step, and then preserve just the support vectors. We study whether incremental SVM training can be robust and reliable based on the three criteria of *stability,improvement*, and *recoverability*. Further, to get a more complete picture, it is also desirable that at each incremental learning step, we compare the performance of the incrementally trained model against the model trained with all the data so far in one batch. Specifically, we need to compare two cases:

Case 1: With only the learned model from the seen data, if new information comes in, how does an SVM do? Does the performance deteriorate as we keep discarding examples at each successive step?

Case 2: With all the seen data, if we see new information, how does an SVM fare on unknown data?

To analyze these two situations the experiment was designed as follows. Let us call the training set TR, and the test set TE. In each iteration the training set was further split into 10 parts TR_i, where $i = 1, 2, ...10$, with each part containing mutually exclusive 10% of the data from TR. The following two types of incremental learning were carried out:

E1. We trained an SVM on TR_1, then took the SVs from TR_1 as SV_1 and added them to TR_2. Again we trained an SVM on $SV_1 + TR_2$ and used the trained machine to classify TE. Now the SVs chosen from the training of $SV_1 + TR_2$ were taken as SV_2, and TR_3 was added to them. Again the training and testing were done. This step was repeated until all the TR_i were used.

E2. An SVM was trained using TR_1. Then the trained machine was used to classify the examples in TE, and prediction accuracy was recorded. Then TR_2 was added to TR_1, and $TR_1 + TR_2$ was used for training. Again the trained machine was used to classify TE, and results

Dataset	*# of Examples*	*No. of Attr.*
Australian	670	14
Diabetes	768	8
German	1000	24
Heart	270	13
Ionosphere	351	34
Liver-Disorders	345	6
Monks-1	432	6
Monks-2	432	6
Monks-3	432	6
Mushroom	8124	22
Promoter-Gene	106	57
Sonar	208	60

Table 1: The datasets used in experiments.

were recorded. Then TR_3 was added to $TR_1 + TR_2$, and so on, until all the TR_i were used.

We use 10% of the data for each partition TR_i. While incrementally learning from large dataset, we can only use as much data at any step as the buffer allows. So this partition could be a certain percentage of the data that's allowed. The two steps above are repeated 10 times. Each time mutually exclusive 10% of the examples were used as a test set TE, and the rest were used for the training set TR (on the lines of 10-fold cross validation.) The datasets used for carrying out the empirical study are summarized in Table 1. The datasets used are those that are standard among the machine learning community, for benchmarking purposes. All the datasets have been obtained from the UCI-machine learning repository[2]. We used SVM^{light}[3] for our experiment. E1 gives us a model obtained through incremental training. With E2 we get a model trained with all the examples so far, i.e., $TR_1 \bigcup TR_2 \bigcup ... \bigcup TR_i$ in one batch. At each step, comparing these two cases gives us an insight as to how the incremental method fares when compared with the batch method, and how well it handles the concept drift. Figure 1 shows the plots of the two cases for the datasets as the learning progresses. The horizontal axis shows incremental steps, and the vertical axis (the height of the bar) denotes the prediction accuracy of the models obtained at the end of the corresponding step.

As can be seen from the figures, the incremental training satisfies the three criteria of stability, improvement and recoverability : **(a)** The change in prediction accuracy at each step does not vary wildly from the performance in the previous step (*stability*). **(b)** In general the accuracy improves as the learning progresses through multiple steps (*improvement*). **(c)** In cases where the prediction accuracy drops at any incremental step, it recovers in a later step or steps (graph 4, 6, 11) (*recoverability*). Overall, we can clearly see that SVMs can handle the concept drift well.

[2] http://www.ics.uci.edu/~mlearn/MLRepository.html
[3] http://www-ai.informatik.uni-dortmund.de/FORSCHUNG/VERFAHREN/SVM_LIGHT/svm_light.eng.html

Dataset	*Batch method*	*Incr. method*
Australian	84.63	84.63
Diabetes	77.08	76.95
German	75.10	74.10
Heart	84.45	79.63
Ionosphere	94.57	93.14
Liver-Disorders	68.68	61.03
Monk-1	100	100
Monk-2	99.17	99.17
Monk-3	98.91	98.73
Mushroom	100	99.88
Promoter-Gene	93.55	92.55
Sonar	87.38	87.41
Average	88.87	87.27

Table 2: Final prediction accuracy of batch and incremental methods

Another interesting and important observation is that the prediction accuracy of the incrementally trained SVM method closely follows the performance of the SVM trained in one batch with all the data. This observation is important for the reason that it provides empirical evidence that the SVs chosen by the SVM algorithm are sufficient to represent the data space, even after successive stages of discarding other examples. Table 2 shows the final prediction accuracy of the incremental and the batch training methods after training is completed through 10 steps. The table clearly shows that there is no drop in prediction accuracy of SVMs when they are incrementally trained.

The very idea of SVs is that they are sufficient to represent the data space, and they have previously been demonstrated to clearly do so [11], but all these results have been shown with batch training and testing. The novelty of our result is that it is the first empirical evidence that even after successive steps of discarding the redundant examples (as in the context of incremental learning), the successively collected set of SVs still remains a representative set. These results demonstrate that support vectors form a *sufficient set* of examples for incremental learning.

5 Succinctness of Support Vectors

Since our aim is to incrementally train SVM from a large dataset, it is desirable that the selected SV set is as small as possible. To investigate whether the selected set of SVs are really a necessary set, we conducted another experiment. In this experiment our aim is to observe the effect of removing some examples from the SV set on the representativeness of the remaining SVs. We conduct an experiment in two steps: **Step 1** The SVM algorithm was run on the whole dataset D, and the SV set SV were obtained. **Step 2** SVM was trained on the obtained SV set SV using 10-fold cross validation i.e. for 10-fold cross validation the training set TR and the test set TE were both obtained from SV. The average prediction accuracy of the trained machine on the test set was recorded.

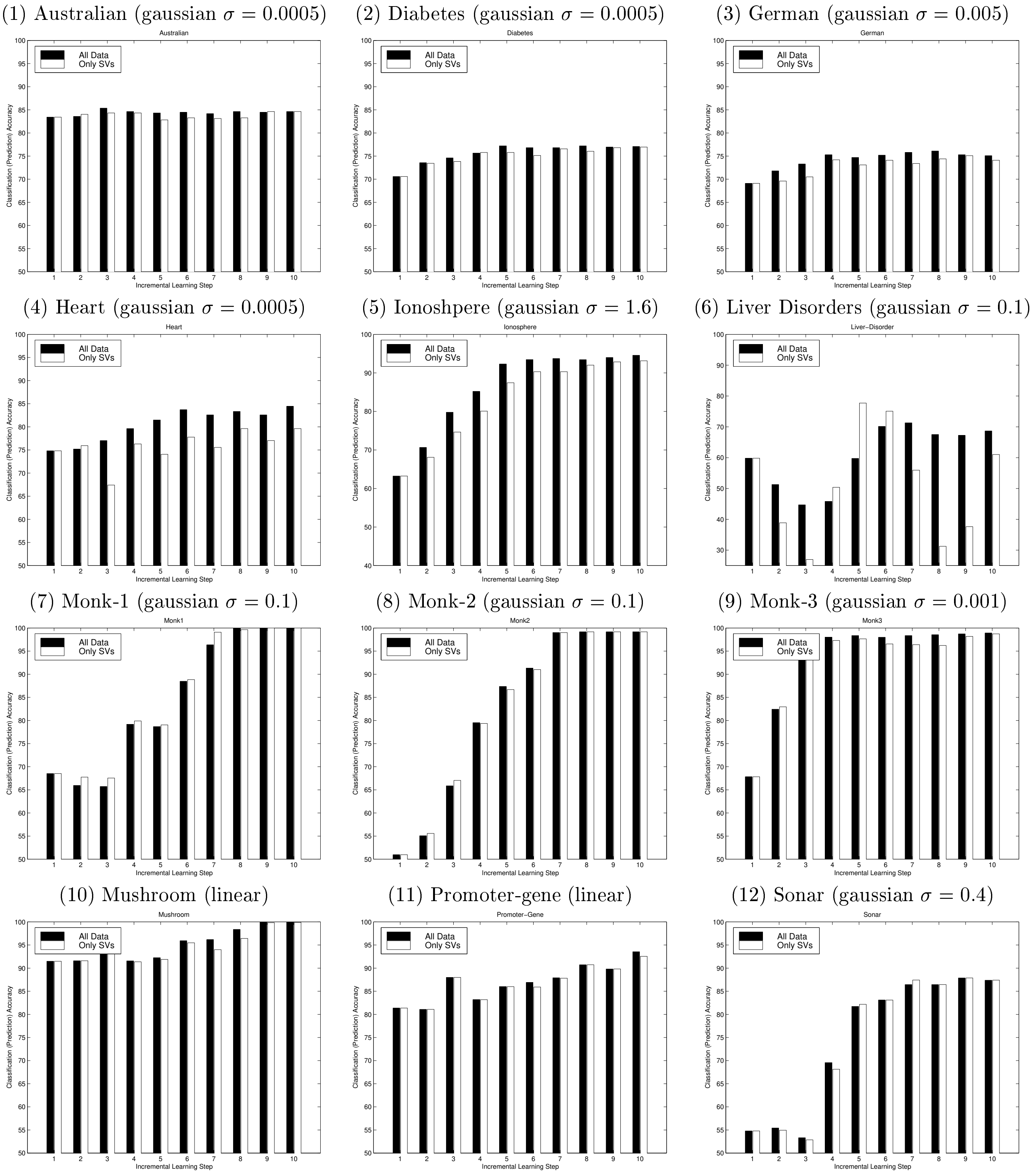

Figure 1: Concept Drift Results on benchmark datasets

Dataset	*% of Exs.* Selected	*%Accuracy* (All data)	*%Accuracy* (Selected Exs.)
Australian	33.76	84.63	48.40
Diabetes	58.03	77.08	51.68
German	54.16	75.10	45.35
Heart	35.16	84.45	33.31
Ion.	53.05	94.57	84.77
Liver.	67.64	68.68	21.25
Monk-1	82.61	100.0	100
Monk-2	78.76	99.17	98.75
Monk-3	23.54	98.91	94.04
Mush.	05.99	100.0	100
Gene	69.19	93.55	79.46
Sonar	67.38	87.38	58.81
Avg.	52.44	88.87	67.99

Table 3: Results illustrating the succinctness of support vector representation

The effect of second step was that 10% of the examples from SV were used to form TE and the rest used to form TR. Further over the 10 folds of cross validation, different $1/10th$ of the SVs were missing from the training set. If all the SVs are necessary then each one of them will be non-redundant. As such, it is reasonable to expect that training an SVM with one part TR of the SV and testing on another part TE should yield poor prediction accuracy. Table 3 provides the prediction accuracy on various datasets for this experiment. The first column shows the percentage of examples chosen as SVs in step 1 of the experiment, the second column shows the average prediction accuracy of the classifier when the 10 fold cross validation was carried out on the whole dataset D. The last column shows the prediction accuracy obtained in step 2 above. It can be seen that the accuracy drops drastically when a training set is created from a subset (90%) of the selected SVs. This means that removing even a small portion of the SVs (10%) from the set SV affects the representation capability of the remaining set. This in turn implies that the SV set chosen by the SVM algorithm is a minimal set and removing any examples from SV would result in the loss of vital information about the class distribution. These results clearly demonstrate that set of chosen SVs forms a compact set that retains the structure of the original data space. These results combined with the results of the previous section provide clear evidence that the SVs form a succinct and sufficient set to ensure that incremental training of SVMs is able to capture concept drift, and hence is a robust incremental learning method.

6 Related Work and Conclusion

Our method can be considered similar in spirit to decomposition or "chunking" techniques employed to train SVMs, as in [6]. Our method differs in that unlike other methods our technique looks at the examples only once to determine if they will become SVs. Once discarded, the vectors are not considered again. On the other hand, for example, Osuna's decomposition algorithm is an iterative method, which cycles through the training set a few times to select the SVs. Our method can be considered as a kind of lossy approximation of the chunking methods. Consequently, our positive results show that such an approximation can be performed, without significantly deteriorating the performance of the algorithm.

We started with the two broad categories of incremental learning techniques. Then we defined three new criteria for the robustness of incremental learning methods. These criteria relate to the ability of any incremental learning method to capture concept drift. Further, we suggested that SVMs should be adaptable to incremental training, and provide the reasons behind our suggestions. To investigate and demonstrate the validity of our suggestion, we empirically showed that support machines are able to capture the concept drift very well. We also provided empirical evidence that the chosen SV sets form a *succinct* and *sufficient* set that enables the incremental SVM to handle the concept drift and satisfy the three criteria for robustness.

References

[1] D. W. Aha, D. Kilber, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[2] J. Catlett. *Megainduction : A Machine Learning on Very Large Databases.* PhD thesis, Department of Computer Science, University of Sydney, Australia, 1991.

[3] J. Catlett. Megainduction : A test flight. In *Proceedings of Eighth International Workshop on Machine Learning*, pages 596–599. Morgan Kaufmann, 1991.

[4] T. M. Cover and P. E. Hart. Nearest neighbour pattern classification. *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, 13:21–27, 1967.

[5] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery:an overview. In *Advances in Knowledge Discovery and Data Mining.* MIT Press, 1996.

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

[7] F. J. Provost and V. Kolluri. A survey of methods for scaling up inductive learning algorithms. Technical Report ISL-97-3, Intelligent Systems Lab., Department of Computer Science, University of Pittsburgh, 1997.

[8] J. C. Schlimmer and D. H. Fisher. A case study of incremental concept induction. In T. Kehler and S. Rosenschein, editors, *Proceedings of the Fifth National Conference on Artificial Intelligence, Philadelphia*, volume 1, pages 496–501, San Mateo, CA, 1986. Morgan Kaufmann.

[9] J. C. Schlimmer and R. H. Granger (Jr.). Incremental learning from noisy data. *Machine Learning*, 1:317–354, 1986.

[10] Paul E. Utgoff. An improved algorithm for incremental induction decision trees. Technical Report 94-07, Department of Computer Science, University of Massachusetts, Amherst,MA 01003, 1994.

[11] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer Verlag, New York, 1995.

[12] G. Widmer and M. Kubat. Effective learning in dynamic environments by explicit context tracking. Technical Report 92-35, Austrian Institute for Artificial Intelligence, Vienna, Australia, 1992.