From Incremental Learning to Model Independent Instance Selection - A Support Vector Machine Approach

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

Abstract

With large amounts of data being available to machine learning community, the need to design techniques that scale well is more critical than ever before. As some data may be collected over long periods, there is also a continuous need to incorporate the new data into the previously learned concept. Incremental learning techniques can satisfy the need for both the scalability and incremental update. In this paper, we categorize the incremental techniques into two broad categories: block by block vs instance by instance. We suggest *three* criteria to evaluate the robustness and reliability of incremental learning methods. We then propose an incremental learning method for Support Vector Machines, and use the suggested criteria to evaluate the effectiveness of the suggested training method. Motivated by positive results on these experiments, we research the possibility of using SVMs for another approach to handling very large datasets. We have carried out a study to evaluate whether the Support Vector Machine (SVM) training can be used to select a small subset of examples from the training set in a model independent way. We compare the results of SVM selection, with IB2 selection method and random sampling. We analyze the experiment results, and discuss their implications. All the results have been illustrated using standard machine learning benchmark datasets.

Keywords : *Incremental Learning, Support Vector Machine, Concept Drift, Instance Selection*

1 Introduction

Example-based learning is a widely used framework for extracting knowledge from empirical data, with the goal of generalizing well on new input patterns. Many real-world problems can be formulated as a 2-way classification task that may be solved using example-based learning methods. Given:

- a set of input-output training examples sampled according to an *unknown* probability distribution $P(\mathbf{x}, y)$ that describes the process:
$(\mathbf{x_1}, y_1), ..., (\mathbf{x_l}, y_l), \mathbf{x_i} \in \Re^N, y_i \in \{-1, +1\}$, and

- a set of admissible functions, parameterized by $\alpha \in \Lambda$, for modeling the process: $\{f_\alpha : \alpha \in \Lambda\}\ f_\alpha : \Re^N \to \{-1, +1\}$

find a classification function f_{α^*} (where $\alpha^* \in \Lambda$) which gives the smallest possible misclassification rate (*risk*) on new input patterns:

$$R(\alpha) = \int \frac{1}{2} |f_\alpha(\mathbf{x}) - y| dP(\mathbf{x}, \mathbf{y}).$$

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 [12]. Researchers in machine learning and data mining [7, 8, 22] have therefore been trying to scale up classical inductive learning algorithms to handle extremely large data sets. Scalability is the next big challenge being tackled by the Knowledge Discovery and Data Mining (KDD) community [11, 13, 21]

Ideally, it is desirable to consider all the examples together for the best learning performance. However, when the training set is very 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. Another possible way of handling this problem is to select a representative subset from the original database for training.

Support Vector Machines [33] have been claimed to have certain desired properties, that make them an attractive tool for both the above approaches to scalable data mining. They are able to summarize the data space concisely and accurately in the form of support vectors. Since they preserve the boundary information for the data space on which they are trained, they should easily fit into certain *standard* incremental-learning frameworks for classification problems. On the other hand, since they can summarize the data succinctly, they should be helpful in carrying out sample selection from very large databases. In this paper our aim is to study the extent to which SVMs can be utilized to address these two different approaches to handling large amounts of data. More specifically, in this paper we will:

- define *two* broad approaches to incremental learning,

- propose *three criteria* for evaluating the robustness of incremental learning algorithms,
- define an incremental learning procedure based on SVMs,
- using empirical data from standard machine learning data sets, evaluate the robustness of incremental training method for SVMs based on our three criteria.
- propose a *model independent* instance selection procedure based on incremental SVMs,
- using standard machine learning data sets, compare SVM instance selection with IB2 [2] selection method and with random sampling, and
- analyze the results of empirical studies, and study their implications.

In the next section we define the term *incremental learning* and discuss related issues. Section 3 provides a brief introduction to Support Vector Machines, and Section 4 provides our incremental training method for SVMs, and evaluates its effectiveness Related methoda are also discussed. In Section 5, we propose model independent instance selection as a possible extension to incremental SVM, and carry out an empirical comparison with two other methods. Finally we conclude by summing up the findings. Parts of this work have appeared in [29, 30, 31].

2 Incremental Learning and Related Issues

In this paper we define *incremental learning* to have the following characteristics:

1. It takes place in multiple steps.
2. Data is divided in disjoint subsets.
3. At each incremental step, the algorithm is provided only with one subset of all the training data.
4. The concept learned in the previous steps is available to the algorithm together with the current subset.
5. The training algorithm is allowed to see any given training subset only once.
6. In one step of incremental training, the algorithm can iterate through the given data subset any number of times.

We propose this framework for incremental learning, keeping in mind the needs of data mining. With very large databases, even one complete scan through the database can be considered expensive [36, 5]. In this scenario, it may not be possible to go back to the previously seen data again. For carrying out learning on very large databases, there are certain desirable properties [5] including:

1. The training stage should require at the most one scan through the whole database.

2. One should be able to incrementally update the learned model when new data becomes available, without having to go back to the previously seen data.

3. The learning method should be able to work with a limited RAM.

4. The learning process should be able to work with just the forward scan of the database, since additional access to the database may be expensive.

Any method that involves iterating through the incremental steps (e.g. the standard decomposition methods for training SVMs [20, 15, 6]) will not be able to satisfy most of these requirements. As such the need is for an incremental training method that will not require iterating through the steps. On the other hand iterating through the data loaded into the RAM within any one incremental step can be carried out a lot more cheaply.

There could be other scenarios where a real-time process may be generating data rapidly. It may not be possible to store all this data into secondary storage. So, the training method should not go back to the previous data as it may not be available anymore. In such a problem domain, the above incremental learning framework will be very useful.

Having discussed our incremental learning framework, we will now broadly categorize the incremental training methods into two types:

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*).

IIL may be considered as a degenerate form of *BBL*, whereby the block size is one, but in practice since we can always process a lot more than one instance at a time, the two types of approaches yield different results. In the case of *BBL*, all the examples collectively influence learning, whereas this collective information is missing in the *IIL* case.

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.

When learning from an online data stream, where samples are not available beforehand, there is no choice but to use the first incremental approach A1, i.e., *IIL* to train the classifier. On the other hand, when some amount of data is available beforehand, the second approach A2, i.e., *BBL* appears more appealing. In this situation, effort can be made to get as close to the ideal case as possible, by considering as many examples together as possible. In the case of large datasets where all examples cannot be accommodated in the main memory, the best alternative is to consider as many examples at a time as the buffer can hold. As such, the second form of incremental learning, *BBL*, can be employed to learn from large databases. In this paper we focus on this approach to incremental learning for handling large datasets.

2.1 Evaluation

Various methods have been reported in the literature on incremental learning, mostly in the context of first two scenarios (S1 and S2) [2, 10, 26, 27, 32]. 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 [27] also propose three criteria:

- the *number of observations (instances)* needed to obtain a 'stable' concept description,
- the *cost of updating memory*, and
- 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 data.

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.

2.2 Concept Drift

With the growing size of databases, the incremental techniques are going to be more popular to tame these monsters. This will necessitate the design of evaluation methods which can demonstrate the reliability of these methods with a high level of confidence, when used in various applications. One notable aspect of current goodness measures is that they are 'one-time' measures, which do not capture the overall picture as the learning evolves through multiple incremental steps.

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* [34]. 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.

It is important at this stage to both emphasize and understand the difference between the *virtual* and the *real* concept drift. The *real concept drift* takes place when the actual underlying structure of the data space or the underlying concept itself changes, as opposed to the virtual concept drift where the actual concept stays the same, but it is perceived to be changing by the learning algorithm. The main idea behind handling real concept drift is that of forgetting or unlearning, i.e. how to unlearn the outdated concept and remember only the new and the relevant one. On the other hand when tackling *virtual concept drift*, what we have is a partially correct concept, or a concept which needs to partially change and evolve towards the *static* actual concept as more examples become available. In this case the focus is more how to remember the relevant information from the past, and use it to evolve a better concept from the data.

2.3 Our Evaluation Criteria

We propose that for effective incremental learning, 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.

To demonstrate the value of these criteria and their usage, we will now devise an incremental training method for SVMs, and show that these criteria are able to capture the effectiveness of our method. A brief summary of the SVM algorithm is provided in the next section, before the incremental training procedure is discussed.

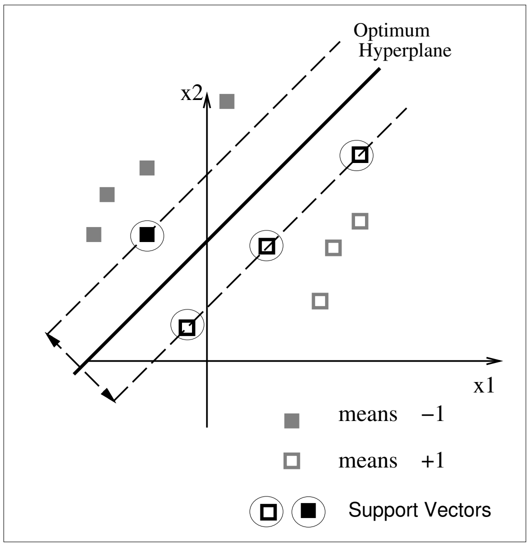


Figure 1: The Separating Hyperplane and the Support Vectors

3 Support Vector Machines

Recently, support vector machines [33] have been rapidly gaining in popularity as a learning paradigm. In classification problems SVMs work by mapping the input vectors into a high dimensional feature space constructing the *Optimal Separating Hyper-plane* through *Structural Risk Minimization* [33]. Only the examples which are relevant in defining the Optimal Separating Hyper-plane (so called support vectors) are stored for doing future classification.

We briefly review the key ideas behind Support Vector Machines which make them so appealing to be adapted to an incremental form of learning.

3.1 Overview

Support Vector Machines work on the principle of Structural Risk Minimization (SRM). This principle tries to maximize the generalization capability on the unseen data by simultaneously reducing the empirical risk on the seen data, and minimizing model complexity. The solution takes the form of an optimal separating hyper-surface that partitions the input space into positive and negative examples. The boundary examples which define the separating hyper-surface are preserved for implementing the decision function during classification. These are the support vectors (Figure 1). The rest of the examples are considered redundant and are discarded. The algorithm is defined in more detail below.

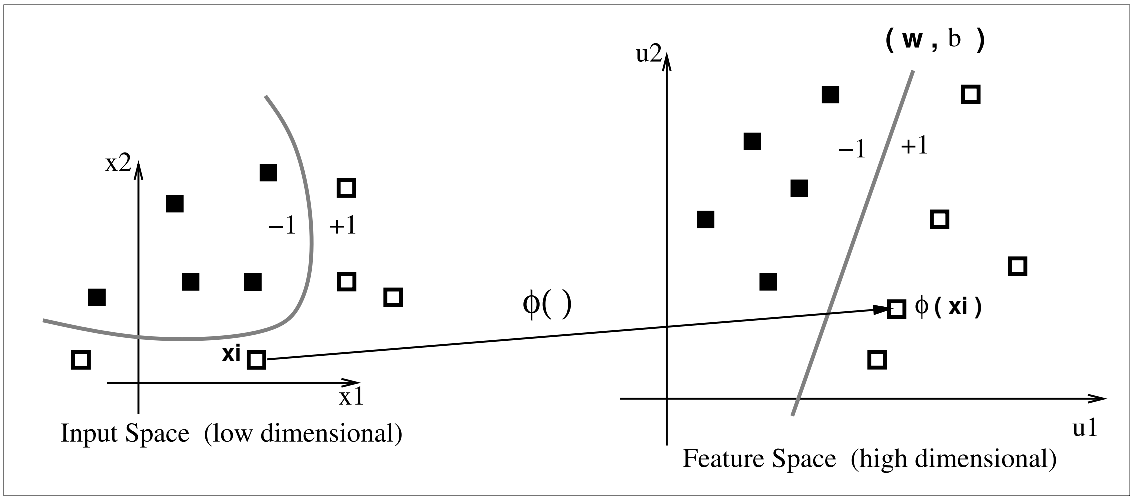


Figure 2: Mapping into higher dimensional space using a kernel function

3.2 The Support Vector Algorithm

Given l data points $\{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x}_l, y_l)\}$, SVM training involves solving a quadratic programming problem that efficiently minimizes an *actual risk* measure. One can show that the optimal solution gives rise to a decision function of the following form; where $\mathbf{x_i}$ are training input vectors:

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

Often, only a small fraction of the α_i coefficients are non-zero. The corresponding pairs of $\mathbf{x_i}$ entries (known as *support vectors*) and y_i output labels fully define the decision function. Together with the α_i coefficients, they are preserved for use in the classification procedure. All remaining training examples that do not contribute to the decision function may be regarded as redundant data.

To allow for decision surfaces other than hyper-planes, one can first non-linearly transform the set of input training data $\mathbf{x_1}, \ldots, \mathbf{x}_l$ into a higher-dimensional feature space (Figure 2) using a map $\mathbf{\Phi}(\mathbf{x_i}) \mapsto \mathbf{z_i}$, and then do a linear separation there. This leads to decision functions of the form:

$$\begin{aligned} f(\mathbf{x}) &= sgn\left[\sum_{i=1}^{l} y_i \alpha_i (\mathbf{\Phi}(\mathbf{x}) \cdot \mathbf{\Phi}(\mathbf{x_i})) + b\right] \\ &= sgn\left[\sum_{i=1}^{m} y_i \alpha_i K(\mathbf{x}, \mathbf{x_i}) + b\right] \end{aligned}$$

where $K(\mathbf{x}, \mathbf{x_i}) = (\mathbf{\Phi}(\mathbf{x}) \cdot \mathbf{\Phi}(\mathbf{x_i}))$ is known as a *kernel*. Often, as in the case of separating hyper-planes, only a small fraction of the α_i coefficients are non-zero, so one can similarly discard a large number of redundant training examples that do not contribute to the decision function.

Notice that the training and classification procedures now involve computing high dimensional dot products of the form $(\mathbf{\Phi}(\mathbf{x}) \cdot \mathbf{\Phi}(\mathbf{x_i}))$. To avoid these computationally expensive operations, we consider only those classes of $\mathbf{\Phi}$ that reduce to simple kernel functions K. Some commonly used kernel functions and the families of decision surfaces/architectures they emulate are:

$K(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y}$	(Separating hyper-planes)
$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^P$	(Polynomial classifiers of degree P)
$K(\mathbf{x}, \mathbf{y}) = e^{-\lvert\lvert \mathbf{x} - \mathbf{y} \rvert\rvert^2 / 2\sigma^2}$	(Gaussian Radial Basis Function classifiers)
$K(\mathbf{x}, \mathbf{y}) = tanh(\kappa \mathbf{x}.\mathbf{y} - \delta)$	(Sigmoidal Perceptron classifiers)

3.3 Desirable Properties of SVM

Previous studies [33, 28] have shown that SVMs possess some desirable properties that might make them easily adaptable to incremental learning framework:

Succinct Representation: The number of training examples that end up as support vectors form a very small percentage of the training data. As a result SVMs are able to capture and retain the information about the classification boundary in a very compact form. This is a very useful feature for incremental training procedure. A concise and accurate summary of class boundary would mean that this information can always be kept in the RAM, and as a result the previously derived classifier will be readily available to the incremental algorithm and can be continuously refined.

Good Generalization: Since the support vector algorithm derives the class boundary through structural risk minimization, it tends to exhibit good generalization to unseen examples. This property is very important for incremental learning to ensure that the partially correct concept at any incremental step is not too different from the actual desired concept.

Kernel Independent: The set of support vectors chosen by SVM stays more or less the same irrespective of the kernel function used. This property is again useful for incremental learning because if the SV set is too dependent on the kernel, it will be very difficult to determine which kernel function (K) to use in the first few incremental steps, when the incremental algorithm would have seen a very small portion of the actual data.

Given these reported results about SVMs, we expect that they should be suitable tools for block-by-block (BBL) incremental learning. In order to evaluate SVMs suitability, we proceed to evaluate the sufficiency and necessity of the selected support vector sets for the purpose of incremental learning.

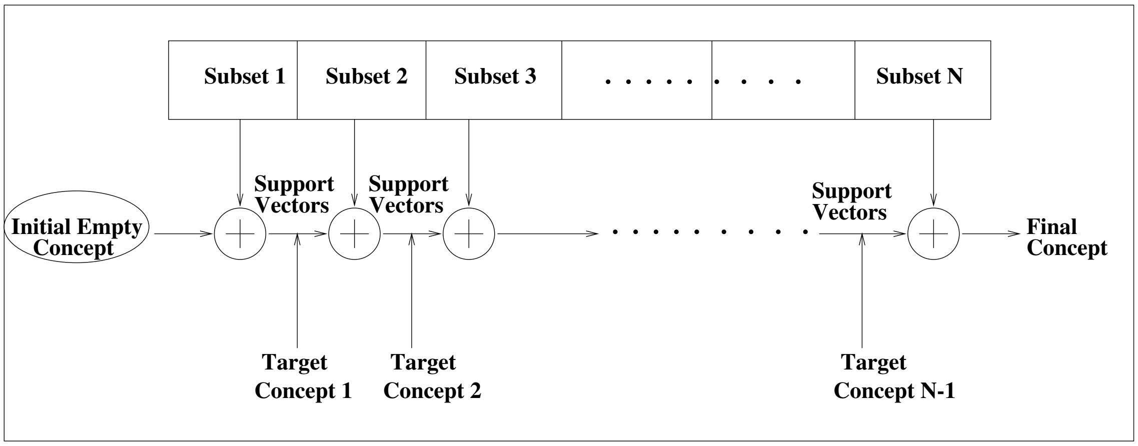


Figure 3: The Incremental Training Procedure

4 Incremental Training of Support Vector Machine

4.1 Incremental Training Method and its Evaluation

The SVM algorithm is able to summarize the class information adequately using a very small set of support vectors. This suggests that we could adopt the following procedure for incrementally training the SVM (Figure 3):

1. *partition the large training dataset TR into k subsets DS_1, $DS_2 ... DS_k$ such that each partition fits into the main memory.*
2. *set training subset $TS = DS_1$*
3. *train SVM with TS and preserve the set of selected support vectors SV_1*
4. *repeat for $1 \leq i < k$*
   (a) *set $TS = SV_i \cup DS_{i+1}$*
   (b) *train SVM with TS and preserve the set of selected support vectors SV_{i+1}*
5. *save SV_k to reconstruct the class boundary for classification.*

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 class boundary will be preserved, and refined with more examples at each step.

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?

Throughout this paper we will use predictive accuracy of the trained classifier on unseen data as our primary criterion of comparison. After evaluating the possibility of using other measures for demonstrating the effectiveness of the methods discussed in this paper, we have decided to use the predictive accuracy, because this is really what learning-based classification is all about. Ultimately, what really matters is the ability of the classifier to make accurate predictions.

4.2 Experimental Procedure

In order to analyze these two cases (Case 1 and Case 2), the experiment was designed as follows:

- Given a full dataset, split it into disjoint training set TR and test set TE.
- The training set is further split into 10 parts DS_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 are carried out:

  E1. 1. *set training subset* $TS = DS_1$
  2. *train SVM with* TS *and preserve the set of selected support vectors* SV_1
  3. *use the SVM trained in the last step to classify* TE *and record the prediction accuracy.*
  4. *repeat for* $1 \leq i < k$
     (a) *set* $TS = SV_i \cup DS_{i+1}$
     (b) *train SVM with* TS *and preserve the set of selected support vectors* SV_{i+1}
     (c) *use the SVM trained in the last step to classify* TE *and record the prediction accuracy.*

Dataset	*Number of Examples*	*No. of Attr.*	*No. of Norm. Attr.*
Australian Credit	670	14(N=6, S=8)	43
Diabetes	768	8(N=8, S=0)	8
German Credit	1000	24(N=24, S=0)	24
Heart Disease	270	13(N=7, S=6)	23
Ionosphere	351	34(N=34, S=0)	34
Liver-Disorders	345	6(N=6, S=0)	6
Monks-1	432	6(N=0, S=6)	17
Monks-2	432	6(N=0, S=6)	17
Monks-3	432	6(N=0, S=6)	17
Mushroom	8124	22(N=0, S=22)	126
Promoter-Gene	106	57(N=0, S=22)	228
Sonar	208	60(N=60, S=0)	60

Table 1: The datasets used in experiments.

E2. *1. set training subset* $TS = DS_1$

2. train SVM with TS

3. use the SVM trained in the last step to classify TE *and record the prediction accuracy.*

4. repeat for $1 \leq i < k$

(a) set $TS = DS_i \cup DS_{i+1}$

(b) train SVM with TS

(c) use the SVM trained in the last step to classify TE *and record the prediction accuracy.*

- **E1** and **E2** are are repeated 10 times. Each time mutually exclusive 10% of the full data set are used as a test set *TE*, and the rest are used for the training set *TR* (on the lines of 10-fold cross validation).

We use 10% of the data for each partition DS_i. There is no special reason for choosing this value as such. While incrementally learning from large datasets, 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 fits into the buffer. In our experiments 10 partitions of the training set *TR* appeared to be a convenient number to illustrate our results. The problems being handled in all these experiments are of two-class classification.

The datasets used for carrying out the empirical study are summarized in Table 1. The third column shows the number of attributes, with the values in the brackets showing the number of numeric (N) and symbolic (S) attributes. The last column shows the number of attributes obtained after normalizing the numeric and binarising the symbolic attributes [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 [16]. We used SVM^{light} [14] for our experiments.

[1] The normalization preprocessing is recommended in [14].

For the choice of kernels the policy of simplest first was used. As such the linear kernel was always the first choice. When the classifier trained with the linear kernel gave prediction accuracy which was far below the previously reported results, the gaussian kernel was used.

E1 gives us a model obtained through incremental training. With **E2** we get a model trained with all the examples available so far, i.e., $DS_1 \bigcup DS_2 \bigcup ... \bigcup DS_i$ 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 4 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.

4.3 Concept Drift

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 (observed in all graphs except 5 and 8) (*recoverability*).

Overall, we can clearly see that SVMs can handle the concept drift well.

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 define the classification boundary, 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 [33], 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.

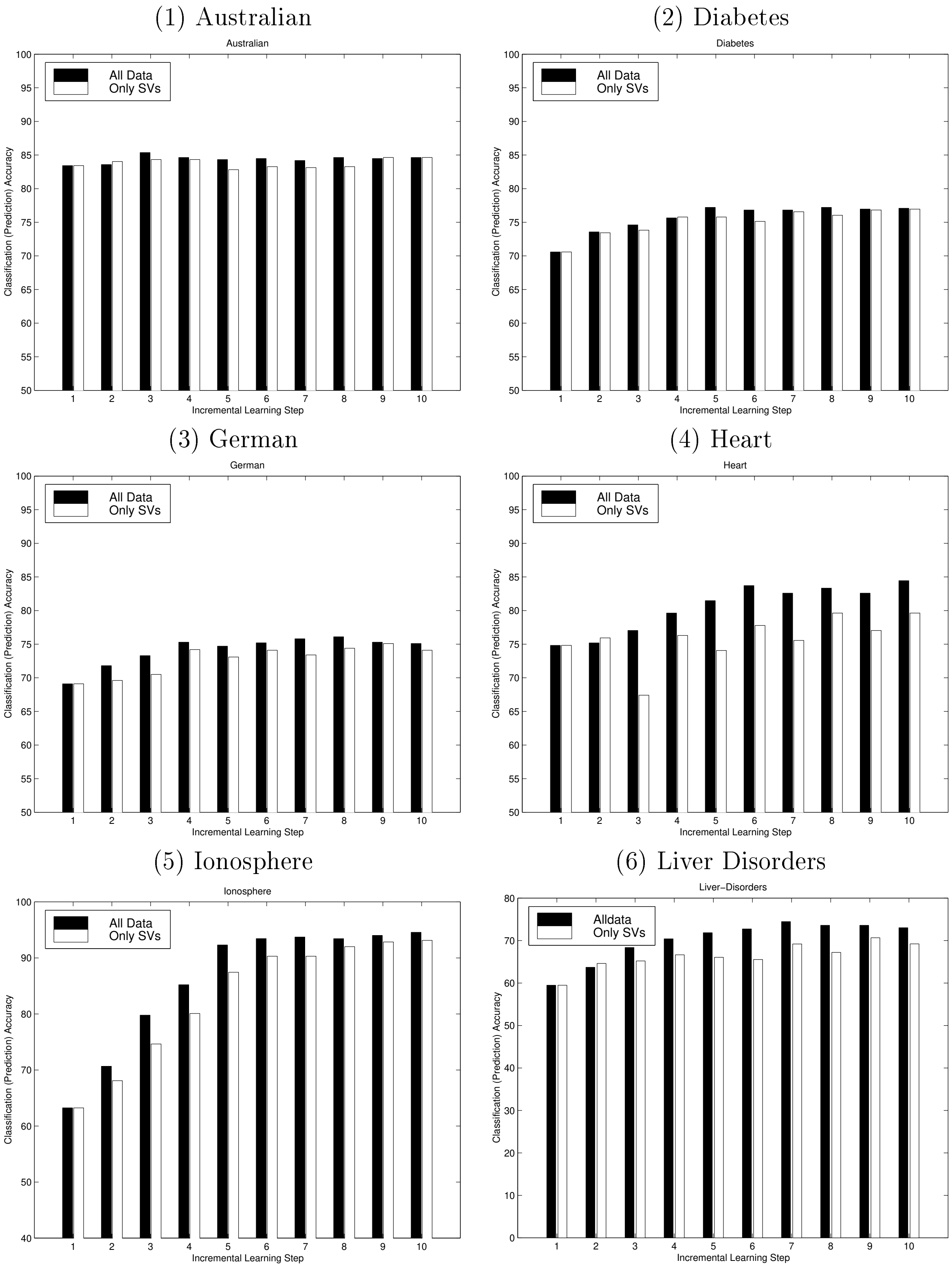
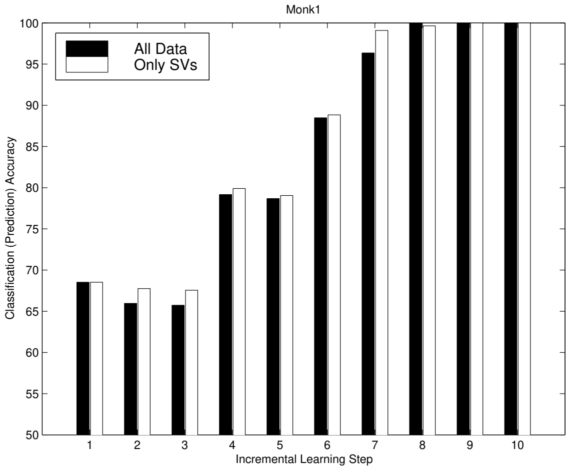
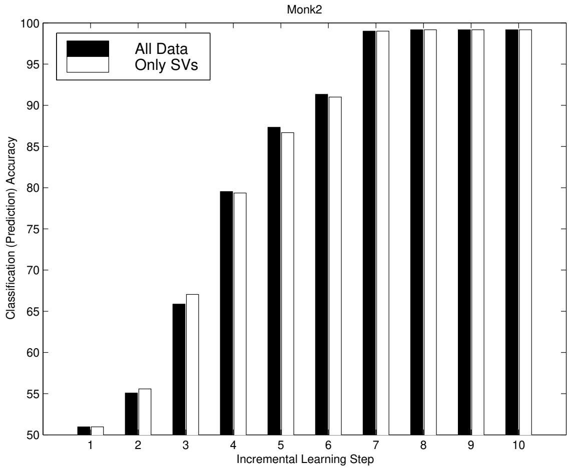


Figure 4: Concept Drift Results on benchmark datasets - Part 1

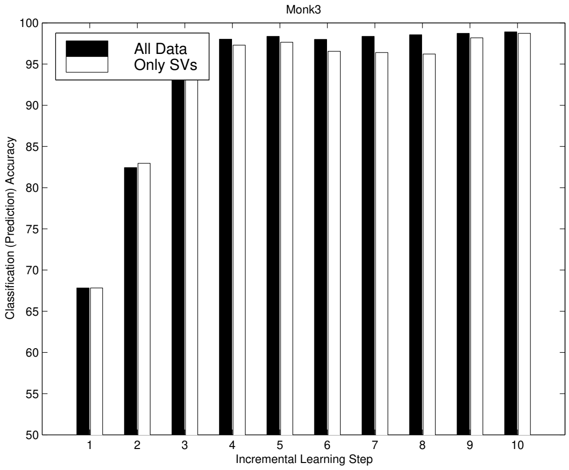
(7) Monk-1



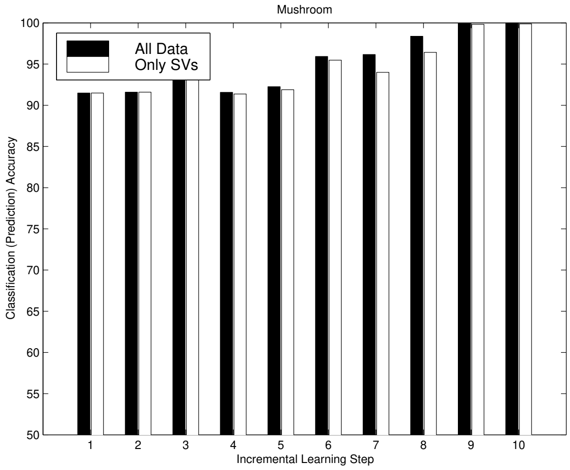
(8) Monk-2



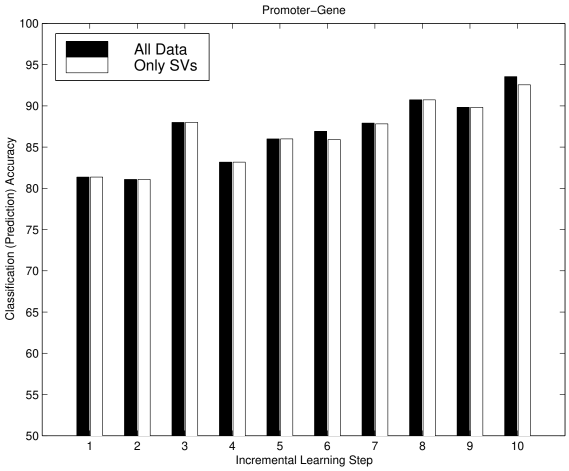
(9) Monk-3



(10) Mushroom



(11) Promoter-gene



(12) Sonar

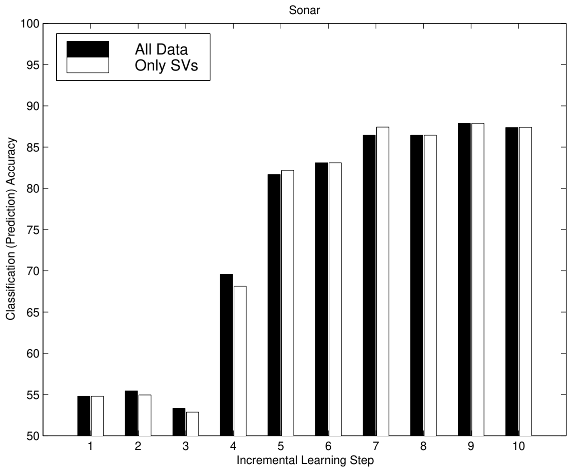


Figure 5: Concept Drift Results on benchmark datasets - Part 2

Dataset	*% Accuracy (Batch method)*	*% Accuracy (Incremental 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	73.03	69.23
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	88.99	87.95

Table 2: Final prediction accuracy of batch and incremental methods

4.4 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. The aim of this experiment 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 set SV was obtained.

Step 2: SVM was trained on the obtained 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.

The effect of Step 2 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

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	73.03	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	88.99	67.99

Table 3: Results illustrating the succinctness of support vector representation

removing any examples from SV would result in the loss of vital information about the class distribution.

4.5 Summary and Related Work

These results clearly demonstrate that set of chosen SVs forms a compact set that retains the structure of the original classification problem. 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 and practical incremental learning method.

Our incremental training method for SVMs can be considered similar in spirit to decomposition or "chunking" techniques employed to train SVMs, as in [20, 15, 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, the decomposition algorithms are iterative, and cycle through the training set multiple 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.

5 Beyond SVM Incremental learning

Our experiments in the last section show that the support vectors are able to preserve the boundary information in a very concise form. This suggests that the SVM may be utilized for another approach to handling large datasets, namely sampling a representative subset for training from a huge dataset. We can simply store the obtained support vectors, and use them to train other classifiers. We investigate the feasibility of this approach in this section.

5.1 Model Independent Instance Selection

One popular approach for dealing with the intractability problem of learning from huge databases is to select small data subsets for training. Training with a small data sample from a large database requires less computation and memory resource. On the other hand, a small training set can omit important information captured in the original large data set, and hence lead to significantly poorer prediction results. Various data sampling methods have been studied [4] such as windowing [23], random sampling, stratified sampling, peepholing [7], and information-theoretic peepholing [18]. Catlett's study [8] showed that certain sampling strategies helped in speed-ups, and improving the accuracy of classifiers in noise free domains. However, he also acknowledged that naive sampling methods are often not suitable for real world domains with noisy training data, where prediction results can degrade unpredictably and significantly.

Boundary hunting methods like IB2 [2] seek out training examples near class boundaries because these examples tend to be useful for locating class boundaries precisely. Assuming only a small fraction of training examples in a large database lie along class boundaries, these boundary hunting methods can be desirable for selecting small but highly informative training data subsets. However, examples selected by a particular boundary hunting method may be specifically tuned for use with an implicit classifier architecture, and hence may not work well for training other classifier types.

An interesting theoretical issue to address is the extent to which one can select small but highly informative training data subsets from large databases in a *model independent* fashion. *Model independence* means selecting data that trains well with a wide range of different classifier types, such that prediction accuracy for each type is comparable to results obtained from training with the entire data set. From a practical standpoint, model independent instance selection can be extremely useful for efficient data storage, because one can treat omitted data points as being redundant in a very broad sense, and these points may be discarded from the database.

Over the past few years, researchers have reported experimental results indicating that the SVM training procedure exhibits certain promising model independent instance selection characteristics for pattern classification. Specifically:

1. SVM training involves selecting a small subset of critical data points (known as *support vectors*) from the original training database. This critical data subset fully and succinctly defines the classification task at hand.

2. SVMs can be implemented with different *kernel functions* to simulate a variety of standard classifier architectures (e.g., polynomial, Gaussian RBF and Neural Network classifiers). SVMs trained with different kernel functions on the same database have been observed to select almost identical data subsets as support vectors [28].

3. SVMs trained with different kernel functions on the same database have given rise to similar high prediction rates on novel data [9].

It is thus tempting to treat SVM training as a form of *model independent* instance selection procedure. Here we attempt to verify the *model independent* characteristics of support vectors, obtained via the *batch mode* SVM training over the entire training database.

5.2 Related Instance Selection methods

In the past various methods have been tried to obtain a subset of the original dataset either to improve accuracy or to speed up the training process. One of earliest strategies which used only a subset of the original dataset for training was windowing [23]. The idea was to reduce the learning time by building an initial tree from a small sample (10-20%) of the training instances called the *window*. Wirth and Catlett [35] found that with windowing, on noisy data the learning time was increased, sometimes by a large factor. O'keefe [19] provided an alternative mechanism to windowing that reduces memory requirements, but it it is even slower than windowing.

Another approach commonly used to reduce the large amounts of data is random sampling (without replacement, with replacement and stratified). Though the random sampling method is model independent the main problem is that many non-boundary samples could get selected whereas the boundary samples could be left out. As such, information about class distribution could get lost and this in turn could cause a drastic fall in the classifier's performance. Statistical learning theory can explain the degradation effect of random sampling.

The IB2 algorithm [2] is an instance based method which stores only those examples which are misclassified during training. The idea here is that only those examples which lie on the class boundary or help to define it will get misclassified. By using only these examples for prediction, the storage needs can be reduced drastically and the significant speed-up can be obtained in the classification process. The training stage of IB2 can be used as an instance selection strategy, and then the subset it selects can be stored to train other classifiers efficiently. One possible problem with this approach could be that even though it selects boundary examples, the selection strategy could be biased towards the *instance-based learning* model, and might not work with other kind of classifiers.

The methods described above are representative of instance selection methods. In our experiments we compare the SVM selection strategy with random sampling (without replacement) since it is model independent, and with the IB2 strategy because it selects the boundary examples. At this stage, we can expect that the examples selected by the SVM be model independent and "better" than those selected by IB2 and random sampling.

5.3 Instance selection - support vectors

Using the same machine learning databases as in the previous section as benchmarks, we conducted the following two sets of experiments. For a few representative classifier types, we compared the differences in prediction results of:

1. training each classifier type with a full database versus training the same classifier with SVM selected examples. This study reveals how well support vectors preserve relevant class information in a model independent fashion.

2. training each classifier type with SVM selected *support vectors* versus training the same classifier with data selected by two other selection methods: *random sampling* and *IB2*. This study compares the information-retaining quality of SVM selected data against data selected by other methods.

We now discuss the SVM instance selection procedure. Given a large input database, run the SV learning algorithm using an appropriate kernel function K, and keep the resulting *support vectors* as selected examples. It is tempting to expect support vectors to exhibit the following characteristics desirable of sampled data:

High information content. Because the optimal decision function from SVM training depends only on examples that are support vectors, one may view the support vector set as a comprehensive data sample that fully defines the classification task at hand.

Representative of available and unseen data. This follows from the argument that SVM training is based on the *structural risk minimization* principle which attempts to minimize an *actual risk* error measure.

Model independence. Fairly model independent, since different kernels (K) have been reported to give similar support vectors. Examples selected should be good training data for different classifier architectures. This also implies that the choice of K may not be critical.

The experiments in the next subsection attempt to quantify the extent to which all this is true.

5.4 Empirical comparison of SVM selection with IB2 selection and RS

We have carried out empirical studies to understand the scope and applicability of these claims. To verify the model independence of the selected examples, we have trained various classifiers with the selected subset, and studied the effect on their performance. To provide us with a benchmark, we have also compared these results with those obtained by training the classifiers with examples selected using the IB2 selection method and random sampling.

We are interested in studying the effect of using a subset of examples to train a classifier on its predictive accuracy. If the change in performance of various classifiers, when trained with a selected subset of examples is comparable, i.e. the increase or drop in performance of various classifiers is similar, then we will have empirical evidence to confirm that the selected subset is indeed model independent. This will in turn mean

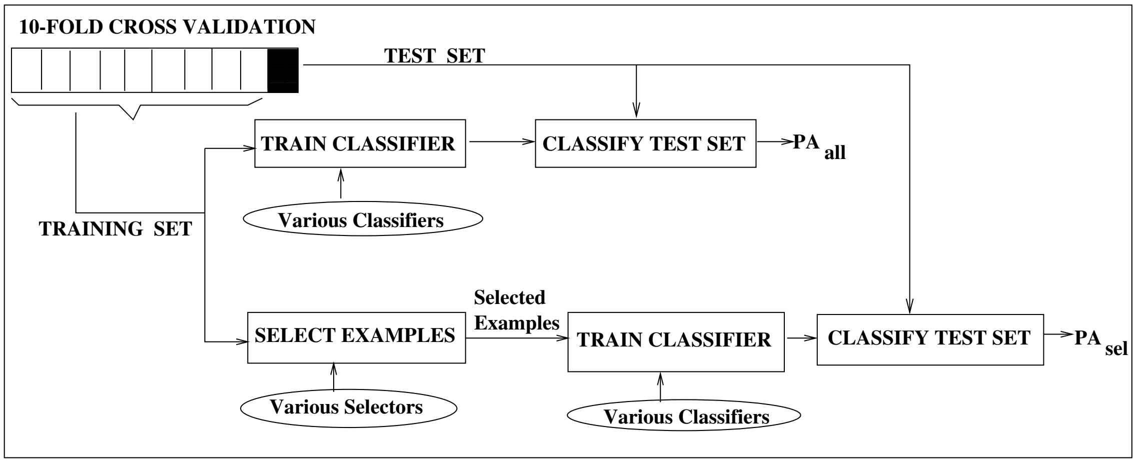


Figure 6: Diagrammatic representation of the experimental procedure

that the selection strategy used to obtain the subset is model independent. It might be that the selection strategy is model independent, but on using the selected subset, the predictive accuracy of various classifiers drops drastically, equally among all the classifiers. In such a situation, the selection strategy is of not much use either. So to provide evidence for model independence of any instance selection strategy, we consider the following two criteria:

1. For a given selection strategy S, the changes in performance of classifiers when trained with the the selected examples should be comparable with when the classifiers are trained with all the examples.

2. If the predictive accuracy of any classifier drops upon using only the selected examples for training, such a drop in accuracy should not be significant.

If we can obtain such a method, then it will be very valuable for scaling up classification algorithms: any classifier will work just as well with the selected examples, and the subset can be used to train various inductive learning algorithms efficiently and quickly.

To verify the model independence of the selected examples, and to study the effect on the accuracy of the classifiers trained with these examples, the following procedure was adopted (Figure 6 shows the steps):

1. *A representative set of classifiers is selected, representing various learning architectures.*

2. *These classifiers are trained with all the data using 10-fold cross validation, and the average classification accuracy on the test set,* PA_{all} *is recorded.*

	Kernel Function	SVM	1-NN	IB2	C4.5	MLP
Austr.	g-0.0005	84.63 (203.9)	81.50 (603.0)	73.58 (151.5)	85.1	97.76
Diab.	g-0.0005	77.08 (401.7)	69.52 (691.2)	64.95 (253.5)	74.4	80.21
German	g-0.005	75.10 (487.0)	66.90 (900.0)	61.20 (338.7)	73.7	96.60
Heart	g-0.0005	84.45 (84.8)	75.58 (243.0)	66.33 (76.1)	77.8	97.78
Ionos.	g-1.6	94.57 (167.1)	84.57 (315.9)	86.86 (54.7)	90	99.43
Liver.	g-0.1	73.03 (209.7)	62.35 (310.5)	57.99 (121.3)	68.4	74.75
Monk-1	g-0.1	100.0 (413.2)	84.91 (500.4)	81.64 (70.3)	100	100
Monk-2	g-0.1	99.17 (425.8)	76.07 (540.9)	68.39 (191.4)	83.7	100
Monk-3	g-0.01	98.91 (116.6)	95.85 (498.6)	88.25 (71.7)	98.9	98.73
Mush.	L	100.0 (437.3)	100.0 (7311)	99.59 (25.6)	100	91.7
Prom.	L	93.55 (65.7)	81.90 (95.4)	84.90 (28.4)	85.8	100
Sonar	g-0.4	87.38 (125.8)	85.47 (187.2)	80.20 (46.9)	72.6	99.5
Avg.		88.99 (261.64)	80.36 (1016.35)	76.16 (119.18)	84.2	94.71

Table 4: Predictive Accuracy (PA_{all}) of various algorithms trained using all the examples.

3. *Next, within the 10-fold cross validation method instance selection is carried out on the the training set, using SVM, IB2, and random sampling respectively, and the selected subsets are stored.*

4. *Finally, the classifiers were again trained with the selected examples, and the average predictive accuracy on the test set* PA_{sel} *is recorded.*

We chose to use the following classifiers as a representative set:

(a) Multi-Layer Perceptron: (MLP) [25, 3] - connectionist learning architecture. Simple backpropagation algorithm was used to train the Neural Network.

(b) Nearest Neighbor (1-NN): [10] - instance-based learning architecture.

(c) C4.5: [24] - tree based classifier.

(d) SVM: [33] - optimal margin classifier.

By looking at our choice of classifiers, it is clear that they cover a large breadth of learning architectures. The criterion for choosing classifiers was that they should have different learning bias, and should cover a wide base of learning architectures. As a result, any selected subset of examples that performs equally well with all the classifiers in the list can be reasonably assumed to be model independent and useful. For $IB2$, we used the one in IBL [1] software in our experiments.

Table 4 provides predictive accuracy of various classifiers (including SVM) when trained with all the training data. The column labeled "kernel function" shows the kernel used for the SVM training: "g" for *gaussian*, and L for *linear*. So, for instance, g-0.4 would mean a gaussian kernel (RBF Kernel) with $\sigma = 0.4$. The linear kernel was chosen by the policy of simplest first. When the classifier trained with the linear kernel gave

	SVM Selection			IB2 Selection				Random Sampling			
	MLP	1-NN	C4.5	MLP	1-NN	SVM	C4.5	MLP	1-NN	SVM	C4.5
Aus.	98.81	54.78	69.7	98.36	72.83	85.37	84.2	98.81	81.05	84.18	84.8
Dia.	67.71	52.84	67.8	69.00	65.87	74.48	70.0	82.56	69.37	73.57	77.6
Ger.	97.19	51.59	65.0	98.1	61.5	74.8	70.6	98.40	66.0	74.4	76.6
Hrt.	100	49.26	66.7	97.04	65.96	81.11	79.6	99.26	70.77	78.88	83.0
Ins.	99.14	85.43	88.6	100	87.15	58.00	69.2	99.43	86.01	66.57	94.0
Lvr.	63.82	57.70	36.6	76.81	58.04	68.41	67.5	88.69	56.23	67.23	69.5
Mk1	100	86.16	100	98	78.99	96.58	98	99.82	78.98	100	100
Mk2	100	71.08	84.8	100	65.06	98.5	63.2	100	67.57	98.00	79.0
Mk3	94.05	73.15	93.8	94.96	87.16	93.49	98.9	99.46	78.19	93.17	95.5
Msh.	99.47	96.28	95.3	61.32	99.71	91.32	86	26.55	98.00	98.14	98.7
Prm.	100	80.99	94.4	100	79.26	79.37	81.0	100	78.99	87.00	88.5
Snr	99.5	85.41	87	100	83.06	68.24	69.7	99.52	79.28	76.33	85.1
Avg.	93.31	70.39	79.25	91.14	75.36	81.05	78.16	91.0	75.85	83.37	86.03

Table 5: Predictive Accuracy (PA_{sel}) of various algorithms trained with selected examples.

prediction accuracy which was far below the previously reported results, the gaussian kernel was used.

The numbers in the brackets show the number of examples saved by the classifier for classification. For SVMs, it means the number of support vectors, for 1-NN and IB2 the number of saved prototypes. Since 1-NN stores all the examples from the training set, the number in the brackets under this column shows the average number of examples in the training dataset. In the case of C4.5 and MLP, they do not store the training data, and so there are no such brackets under their column.

The support vectors selected by SVMs after training were stored as the selected subset. Similarly the prototypes by IB2 training were saved after training. The instance selection process was also carried out using random sampling. For selecting examples by random sampling, the number of examples sampled was equal to the maximum of the numbers of examples selected by SVM and IB2. After doing instance selection using the three strategies, the classifiers were again trained using only the selected examples, with 10-fold cross validation, and the average predictive accuracy on the test set PA_{sel} was recorded. This predictive accuracy is shown in Table 5 for the classifiers. Since we are interested in the change in performance, it will be more instructive to look at difference in predictive accuracy ($PA_{sel} - PA_{all}$). Table 6 shows these values: the negative numbers show a drop in performance on using just the selected examples for training, and the positive numbers mean an increase in predictive accuracy.

5.5 Findings and Discussion

The following can be observed from Table 6.

1. Which selection strategy is the best?

	SVM Selection			IB2 Selection				Random Sampling			
	MLP	1-NN	C4.5	MLP	1-NN	SVM	C4.5	MLP	1-NN	SVM	C4.5
Aus.	1.05	**-26.72**	**-15.4**	0.6	-8.67	0.74	-0.9	1.05	-0.45	-0.45	-0.3
Dia.	**-12.5**	**-16.68**	-6.6	**-11.21**	-3.65	-2.6	-4.4	2.35	-0.15	-3.51	3.2
Ger.	0.59	**-15.31**	-8.7	1.5	-5.4	-0.3	-3.1	1.8	-0.9	-0.7	2.9
Hrt.	2.22	**-26.32**	**-11.1**	-0.74	-9.62	-3.34	1.8	1.48	-4.81	-5.57	5.2
Ins.	-0.29	0.86	-1.4	0.57	2.58	**-36.57**	**-20.8**	0.0	1.44	**-28.0**	4.0
Lvr.	**-10.93**	-4.65	**-31.8**	2.06	-4.31	-4.62	-0.9	13.94	-6.12	-5.80	1.1
Mk1	0.0	1.25	0.0	-2.0	-5.92	-3.42	-2.0	-0.18	-5.93	0.0	0.0
Mk2	0.0	-4.99	1.1	0.0	**11.01**	-0.67	**-20.5**	0.0	1.44	**-28.0**	4.0
Mk3	-4.68	**-22.7**	-5.1	-3.77	-8.69	-5.42	0.0	0.73	**-17.66**	-5.74	-3.4
Msh.	7.77	-3.72	4.7	**-30.38**	-0.29	-8.68	**-14.0**	**-65.15**	-2.0	-1.86	-1.3
Prm.	0.0	-0.91	8.6	0.0	-2.64	**-14.18**	-4.8	0.0	-2.91	-6.55	2.7
Snr.	0.0	-0.06	**14.4**	0.5	-2.41	**-19.14**	-2.9	0.0	-6.19	**-11.05**	**12.5**
Avg.	**-1.40**	-10.0	-4.275	**-3.57**	-5.00	-8.85	-6.04	**-3.71**	-4.51	-8.11	1.83

Table 6: Changes ($PA_{sel} - PA_{all}$) in predictive accuracy of various classifiers. The bold cases are those where accuracy changes by 10% or more. Under each selection strategy, the average change in bold means the most stable.

(a) The random sampling selection strategy is the best in terms of model independence. The reason behind this could be that in general the sampling will be based on distribution of data points.

(b) In spite of better performance in general, because of its random nature, random sampling can sometimes cause catastrophic decrease in accuracy, as can be seen the performance of MLP on the mushroom data when it is trained with randomly sampled subset.

(c) Overall, among the three example selection strategies, SVM selected examples perform the poorest in terms of model independence. The reason behind maybe that the support vectors are specifically tuned to do well with methods based on *structural risk minimization*, whereas most algorithms carry out only *empirical risk minimization*

(d) Random sampling still holds the edge over other techniques.

2. Which learning algorithm is the most robust in using the selected examples?

(a) MLP in general is the most robust of all the classifiers, irrespective of the selection strategy used to select its training subset.

(b) C4.5 actually performs very well with training set selected using random sampling. In many cases, its predictive accuracy improves when trained with the randomly sampled training subset. However, its performance drops in large number of cases on using SVM, and IB2 selected examples for training.

(c) For the majority of cases 1-NN performs reasonably well when trained with IB2 selected, and randomly sampled examples, but its predictive accuracy drops drastically, when it is trained using SVM selected examples.

(d) Conversely, SVM also performs very badly when trained with IB2 selected examples. Further it performs badly even when trained with randomly sampled examples.

In short, the SVM selection strategy performs poorly as a model independent example selector, whereas the random sampling strategy appears to provide best results - the average decrease in predictive accuracy is the least. Valuable lessons are learned from these observations:

- More often than not the random sampling strategy is a good choice for model independent instance selection. It is the most robust among the three strategies, yet simplest and fastest.
- Use of SVMs for model independent instance selection is not a feasible idea in general.
- Although the instance based learning algorithms and SVMs hunt boundary data points, they vary vastly in their learning bias. As such using one as an instance selection method to get a training set for training the other will often not work.
- The observation (1.(b)) suggests a need for caution when using random sampling as a data reduction method. Sometimes, it could cause an adverse effect as was also found in [8].

The above evidence suggests that the model independence of the support vectors is limited to the SVM formulation. In fact, the bias of the SVM formulation seems to be very different from the bias of the algorithms corresponding to the SVM kernels. For random sampling, its very randomness lands it with advantage in gaining model independence. Its lack of bias means that in many cases, randomly sampled subsets can still give us a reasonable performance. On the other hand, this randomness also lends itself to being unreliable because there could be cases when the randomly selected examples cause catastrophic loss in predictive accuracy.

5.6 Summary and Future Work

In conclusion, when no prior knowledge about which learning algorithm will be used for classification, the best strategy for example selection is random sampling. When we know which type of learning algorithm to be used, however, we can have more choices. The salient finding of this study is to realize the limits of SVMs and therefore we are one step closer towards our goal of model independent example selection, and able to suggest a general guideline for example selection facing a suite of selection strategies.

Our positive results with SVM incremental learning and relatively better performance of the examples selected by random sampling in terms of model independence, point to another possible approach to model independent example selection. In future, studies can be carried out to study the feasibility of using SVM density estimation method [17] for selecting representative data based on sample distribution instead of boundary information.

6 Conclusion

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. Because of these positive results for SVMs, we set out to study whether the support vector sets can be used to efficiently train other kinds of learning algorithms. In a nut shell, our endeavor lead us to many valuable findings but we failed to find a model independent selection strategy that is more robust than random sampling.

References

[1] D. W. Aha. IBL Softare. Available from `http://www.aic.nrl.navy.mil/~aha/software/list.html`.

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

[3] I. Aleksander and H. Morton. *An Introduction to Neural Computing*. Chapman and Hall, 1990.

[4] A.L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97:245–271, 1997.

[5] P. Bradley, U. M. Fayyad, and C. Reina. Scaling clustering algorithms to large databases. In *Proceedings of The Fourth International Conference on Knowledge Discovery and Data Mining, New York City, August 27-31*, 1998.

[6] P. S. Bradley and O. L. Mangasarian. Massive data discrimination via linear support vector machines. Technical Report 98-05, CS Dept., University of Wisconsin, Madison, May 1998.

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

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

[9] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[10] 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.

[11] U. M. Fayyad, D. Haussler, and P. Stolorz. KDD for science data analysis: Issues and examples. In *Proceddings of the Second International Conference on Data Mining and Knowldege Discovery*, pages 50–56, Menlo Park, CA., 1996. AAAI Press.

[12] 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.

[13] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. Knowldege discovery and data mining: Towards a unifying framework. In *Proceddings of the Second International Conference on Data Mining and Knowldege Discovery*, pages 82–88, Menlo Park, CA., 1996. AAAI Press.

[14] T. Joachims. SVM^{light} : Support vector machine. SVM software available free for scientific purposes at `http://www-ai.informatik.uni-dortmund.de/FORSCHUNG/VERFAHREN/SVM_LIGHT/svm_light.eng.html`.

[15] T. Joachims. Making large-scale svm learning practical. In B. Schlkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999.

[16] C. J. Merz and P. M. Murphy. UCI repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html`. Irvine, CA: University of California, Department of Information and Computer Science, 1996.

[17] S. Mukherjee and V. Vapnik. Multivariate density estimation: A support vector machine approach. Technical Report A.I. Memo No. 1653, MIT AI Lab, 1999. available by anonymous ftp from publications.ai.mit.edu under the pathname ai-publications/1500-1999/AIM-1653.ps.

[18] R. Musick, J. Catlett, and S. Russel. Decision theoretic subsampling for induction on large databases. In *Proceedings of The Tenth International Conference on Machine Learning*, pages 212–219, Mateo, CA., 1993. Morgan Kaufmann.

[19] R. A. O'Keefe. Concept formation from very large training sets. In *Proceedings of the Twelfth International Joint Conference On Artificial Intelligence*, Milano, Italy, 1983. Morgan Kaufmann.

[20] 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.

[21] G. Piatetsky-Shapiro, R. Brachman, T. Khabaza, W. Kloesgen, and E. Simoudis. An overview of issues in developing industrial data mining and knowledge discovery applications. In *Proceddings of the Second International Conference on Data Mining and Knowldege Discovery*, pages 89–95, Menlo Park, CA., 1996. AAAI Press.

[22] 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.

[23] J. R. Quinlan. Learning efficient classification procedures and their appplication to chess endgames. In R. Michalski, J. Carbonell, and T. Mitchell, editors, *Machine Learning : An AI Approach*. Morgan Kaufmann, Los Altos, CA., 1983.

[24] J. R. Quinlan. *C4.5: Programs For Machine Learning*. Morgan Kaufmann, San Mateo, CA., 1993.

[25] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart, D. E. McCleland, and "the PDP Research Gruop, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1, Foundations.*, pages 318–362. MIT Press, Cambridge MA, 1986.

[26] 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.

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

[28] B. Scholkopf, D. Burges, and V. Vapnik. Extracting support data for a given task. In *Proceedings of Fist International Conference on Knowledge Discovery & Data Mining*, pages 252 – 257. AAAI, 1995.

[29] N. A. Syed, H. Liu, and K. K. Sung. Handling concept drifts in incremental learning with support vector machines. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-99)*, San Diego, CA, USA, 1999.

[30] N. A. Syed, H. Liu, and K. K. Sung. Incremental learning with support vector machines. In *Proceedings of the Workshop on Support Vector Machines at the International Joint Conference on Artificial Intelligence (IJCAI-99)*, Stockholm, Sweden, 1999.

[31] N. A. Syed, H. Liu, and K. K. Sung. A study of support vectors on model independent example selection. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-99)*, San Diego, CA, USA, 1999.

[32] P. 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.

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

[34] 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.

[35] J. Wirth and J. Catlett. Costs and benefits of windowing in ID3. In J. Laird, editor, *Proceedings of the Fifth International Conference on Machine Learning*, Ann Arbor, Michigan, 1988. Morgan Kaufmann.

[36] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In H. V. Jagadish and I. S. Mumick, editors, *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*, pages 103–114. ACM Press, 1996.