Article

Using Domain Adaptation for Incremental SVM Classification of Drift Data

Junya Tang, Kuo-Yi Lin and Li Li *

School of Electronics and Information Engineering, Tongji University, Shanghai 201804, China
* Correspondence: lili@tongji.edu.cn; Tel.: +86-021-69589241

Abstract: A common assumption in machine learning is that training data is complete, and the data distribution is fixed. However, in many practical applications, this assumption does not hold. Incremental learning was proposed to compensate for this problem. Common approaches include retraining models and incremental learning to compensate for the shortage of training data. Retraining models is time-consuming and computationally expensive, while incremental learning can save time and computational costs. However, the concept drift may affect the performance. Two crucial issues should be considered to address concept drift in incremental learning: gaining new knowledge without forgetting previously acquired knowledge and forgetting obsolete information without corrupting valid information. This paper proposes an incremental support vector machine learning approach with domain adaptation, considering both crucial issues. Firstly, a small amount of new data is used to fine-tune the previous model to generate a model that is sensitive to the new data but retains the previous data information by transferring parameters. Secondly, an ensemble and model selection mechanism based on Bayesian theory is proposed to keep the valid information. The computational experiments indicate that the performance of the proposed model improved as new data was acquired. In addition, the influence of the degree of data drift on the algorithm is also explored. A gain in performance on four out of five industrial datasets and four synthetic datasets has been demonstrated over the support vector machine and incremental support vector machine algorithms.

Keywords: incremental learning; domain adaptation; SVM classification; ensemble learning

MSC: 68T09

1. Introduction

The success of machine learning has been demonstrated in data analysis methods such as classification, clustering, regression, and image and speech recognition. In batch machine learning projects, a common assumption is made that the training dataset is obtained at once and follows a specific invariant distribution. However, the assumptions used in the traditional data analysis do not fully meet the requirements of the practical applications due to the continuous generation of new data. Some machine learning tasks require datasets collected continuously over an extended period to be processed. With the increasing amount of data collected, historical data may become less and less relevant for the purposes of the current project. For example, the importance of browsing data used by a recommendation increases over time. Moreover, getting enough training data in the initial stages of machine learning does not ensure that its models perform well enough on new data due to the changing distribution of intrinsic data over time. For example, Amazon collects historical shopping information about customers to obtain their shopping preferences. However, customers’ shopping biases tend to change with age and region, and at this time, the corresponding recommendation model should dynamically adapt to these changes.
Natural learning systems have an inherent incremental nature, in which new knowledge is continuously acquired over time while existing knowledge is maintained. In the real world, incremental learning capabilities are required for many applications. In the case of a face recognition system, for example, it should be possible to add new persons while keeping the previously recognised faces. Unfortunately, most machine learning algorithms suffer from catastrophic forgetting, which results in significant performance degradation when past data are unavailable. Incremental learning is a great approach to meet the need to learn new data and update models continually.

Incremental learning involves the continuous use of data to enrich a model developed on previously available data. Polikar R. provided the most widely accepted definition of an incremental learning algorithm that meets the following conditions [1]: it should be able to learn from new data, and at the same time, it should be able to retain previously acquired knowledge without accessing previous data.

Current incremental learning always follows the definition and assumption proposed by Polikar R. [1]. Two challenges are presented by the absence of data for old classes: maintaining the classification performance on old classes and balancing the classification performance between old and new classes. The former challenge has been effectively addressed through distillation. Recent studies have also demonstrated that selecting a few exemplars from the old classes can help correct the imbalance problem. These incremental learning methods usually assume that the training and the test data share the same distribution. However, it is essential to note that the concept drift problem makes training a stable model challenging, because predictions become increasingly inaccurate as the data sample’s distribution changes over time. This variation in a regular pattern of evolution over time can result in the input data shifting from time to time following a period of minimal persistence. As the number of samples increases, most batch machine learning models can gradually approximate the stationary distribution of the data [2]. However, in a non-stationary environment with concept drift, this machine learning faces more difficulties. For example, in recommender systems, user preferences change.

Models of incremental learning must consider two factors when considering concept drift. The first is the stability–plasticity dilemma: a model gains new knowledge without corrupting or forgetting previously acquired knowledge. The second is how the model selectively forgets worthless previous information and retains still valuable, already learned information [3]. In this paper, an ISVM (incremental support vector machine) using domain adaptation is proposed to decrease the impact of concept drift. First, for the first factor, a small amount of new data is used to fine-tune the previous model to obtain a model that adapts the new data and retains the previous data information by parameter transfer. The amount of past information retained is determined by the difference in the distribution of old and new data. Second, for the second factor, an ensemble mechanism and model selection mechanism based on Bayesian theory are proposed. Only a limited number of models are stored as base models for the ensemble. The effectiveness of the proposed methodology is demonstrated with the help of a case study.

The remainder of this paper is structured as follows. In Section 2, a review of relevant studies of incremental learning, concept drift in incremental learning, and domain adaptation is provided. Section 3 outlines the proposed methodology. Section 4 reports an experimental study using several datasets to demonstrate the effectiveness of the method. Section 5 concludes the paper.

2. Related Work
2.1. Incremental Learning Method

A common assumption in machine learning is that training data is complete, and the data distribution is fixed. Since this assumption does not hold in many practical applications, machine learning sometimes does not fully meet the requirement. Therefore, incremental learning, which does not need this assumption, is an urgent need. There are
three main approaches: ensemble learning-based approaches, batch machine learning-based approaches, and deep learning approaches.

Ensemble learning is a common algorithm for an incremental learning task, mainly focusing on combining multiple neural networks. Polikar R. proposed the first ensemble algorithm named Learn++, which utilises Adaboost to integrate several neural networks and update the weight of each one when a chunk of new data arrives [1]. This algorithm performed well on several standard datasets and a real classification dataset. However, it also exposed shortcomings such as out-voting, non-stationary environment, unbalanced data, and new classes. To address these problems, some improvement has been made. Muhlbaier proposed Learn++.MT and Learn++.MT2 [4–7]. It solved out-voting by improving the performance of weak classifiers and reducing the generation of weak classifiers. Moreover, this algorithm is also friendly to data with new classes. Learn++.NSE was proposed for learning in non-stationary environments [8–10]. It gave a solution to the periodic drift of data distribution. Suppose the periodic drift of data distribution re-relates the previous classifiers to the new data. In that case, the algorithm will recognise this change in the data and assign a higher weight to the previous classifiers. Ditzler proposed an improved algorithm Learn++.NIE for unbalanced data analysis [11]. Learn++.NIE is based on Learn++.NSE, which introduced a dynamic weight mechanism. In the present, the Learn++ algorithm has been greatly developed, and it can gain good performance in most situations. Here, we mention ensemble learning to provide a kind of incremental learning approach rather than discuss ensemble learning in-depth. Extensive overviews of ensemble learning can be found in [12,13].

Some traditional machine learning algorithms can also be developed as incremental algorithms. An ISVM is a common one. A support vector is a small part of samples that can fully describe exclusive features. This characteristic allows an SVM (support vector machine) to be easily developed as an ISVM. Wang utilises support vectors and proposes an incremental learning algorithm based on forgetting factors [14]. It selects parts of samples through the knowledge of the spatial distribution of samples accumulated gradually and forgets them. This algorithm greatly reduces the storage space occupied and improves the efficiency of the algorithm. Liang proposed an incremental SVM implemented in the original model and showed that data arriving in the sequence was learned very well [15]. Zheng proposed an incremental SVM consisting mainly of two parts: a learning prototype and a learning support vector [16]. The learning prototype learns prototypes and constantly adjusts the prototypes to fit the conceptual transfer of data. The learning support vector will obtain a new SVM by combining the learned primitive type with the trained support vector set to realise incremental learning. This algorithm can promote the efficiency of big data analysis. Based on ISVM, many researchers proposed improved algorithms and applications. Wang J et al. introduced a kernel regression to ISVM to deal with semi-supervised incremental learning by estimating the unlabelled samples [17]. Gu B. et al. extended the traditional cost-sensitive learning to an online scenario by a chunk of incremental SVM to avoid time waste [18]. Li J et al. combined the incremental learning paradigm with incremental SVM, which has a remarkable improvement for solving the travelling salesman problem [19]. Aldana Y.R. et al. proposed an SVM-based incremental learning method to address the problem that the batch (nonconvulsive epileptic seizures) detection method failed in the changing data [20].

Furthermore, some methods extend incremental learning to semi-supervised and unsupervised learning. For example, Li Y. et al. used a generative network to learn the intrinsic features of the data, some of which are unlabelled [21]. Hu J et al. used soft clustering to determine region boundaries, helping deal with the overlapping distribution in incremental classification [22]. Pari R. et al. proposed an MTSE (Multitier stacked ensemble) algorithm, which introduced meta-learning [23]. Magdiel J.G. et al. applied a class incremental learning approach to EEG-based emotion recognition [24].
Both batch machine learning-based and ensemble learning-based incremental learning algorithms are widely studied. However, most algorithms face a difficult problem in incremental learning: concept drift.

2.2. Concept Drift in Incremental Learning

Concept drift refers to the change in data distribution. Based on Bayes theory, it has two categories: the change of $P(X)$, called virtual concept drift, and the change of $P(Y|X)$, called real concept drift. In most cases, it refers to the second one. There are two main methods to detect concept drift: slip window and concept drift detection.

The sliding window is widely used in concept drift detection. Sliding windows maintain a fixed or variable amount of data. By adding new data to the window and moving the previous data out, the data in sliding windows can approximately represent the latest data distribution. An update of the model is determined by a change in the data distribution in sliding windows. The Flora method proposed by G. Widmer is an earlier sliding window algorithm which uses decision rules as a learning model and updates these rules according to the data distribution in sliding windows [25]. Hulten G. also proposed the CVFDT algorithm, and these two algorithms are both single sliding windows [26]. However, the window size is such a core issue that a large window is not sensible enough, especially for gradual drift. In contrast, the small window is so reasonable that models are over-updated.

Zhu Qun proposed a two-layer window algorithm detecting concept drift according to the change of data in two sliding windows [27]. It can obtain better performance compared to a single sliding window. However, the window size is also a problem. Due to the uncertainty of concept drift, the fixed-size window is not enough. Therefore, a variable-size window is considered [28]. The OnlineTree2 algorithm Núñez M. proposed uses a decision tree model. It utilises different size windows in each decision tree leaflet node to improve the processing precision of concept drift [29].

After the concept drift occurs, the original model will have a significant error on new data. Therefore, processing the concept drift after detecting it is also very important. There is still a lack of a method for decreasing concept drift. Retraining the whole model is the most common. However, it is time-consuming and expensive. The appearance of transfer learning provides a new idea to solve the concept drift problem.

2.3. Domain Adaptation

Domain adaptation, a particular category of transfer learning, aims to learn a model that reduces the data shift between the source and target distributions and is considered a new approach for concept drift. Most domain adaptation algorithms bridge the source and target domains by estimating instance importance or learning domain-invariant representations using labelled source domain and unlabelled target domain data [30–33]. Domain adaptation has been a success in diverse applications across many fields, such as computer vision and natural language processing, due to its good performance.

As discussed, the existing ISVM algorithms have two categories: adding the previous support vectors to new data and training them together or using previous SVMs as a part of the ensemble model. These two algorithms are effective and easy to implement in incremental learning. However, the first one has little effect on concept drift because previous support vectors may not benefit current data. An ensemble model can solve it because the previous SVM would be assigned a small weight when the diversity between previous and current data is significant. In the extreme case, its weight may be 0, which means the knowledge contained in the previous SVM is useless. However, this is not proper because all previous knowledge is forgotten. Due to the above problems, an ISVM algorithm with domain adaptation is proposed in this study [34]. We view previous data and current data as the source domain and target domain and adapt the previous SVM to current data before the final ensemble. Furthermore, considering using fewer models to maintain more precious knowledge, we select a model combined with the biggest diversity.
3. Methodology

Before introducing our method, we first introduce some key concepts and definitions mentioned in this study.

Definition 1: **Transfer learning** is a machine learning method which reuses a pre-trained model as the starting point for a model on a new task [35].

Definition 2: **Domain adaptation** is an essential part of transfer learning, which aims to build machine learning models that can be generalised into a target domain and deal with the discrepancy across domain distributions [36].

Definition 3: **Catastrophic forgetting** is a phenomenon of the tendency for knowledge of the previously learned task(s) to be abruptly lost as information relevant to the current task(s) [37].

Definition 4: A **Hilbert Space** is an inner product space that is complete and separable with respect to the norm defined by the inner product [38].

Definition 5: A **RKHS (Reproducing Kernel Hilbert Space)** is a Hilbert space $H$ with a reproducing kernel whose span is dense in $H$. We could equivalently define an RKHS as a Hilbert space of functions with all evaluation functionals bounded and linear [38].

Definition 6: A **Mercer kernel** refers to the kernel function determined by the Mercer theory that can be used for SVM [38].

3.1. Support Vector Machine

SVM is an algorithm based on statistical theory, and it improves the ability to learn generalisation by seeking the minimum structured risk [39]. The SVM classifier is constructed from samples and depends on an RKHS with a Mercer kernel [38–42]. In many areas, deep learning has proven to achieve better results than traditional machine learning algorithms by training many samples, usually reaching tens of thousands of samples. However, sometimes, the amount of data stored falls far short of the requirements of deep learning. SVM can generally achieve good results in problems with limited data [43–45]. In addition, SVMs can be extended to incremental SVMs without requiring an integration strategy. In this paper, the proposed ISVM_DD (incremental support vector machine with domain adaptation) is based on ISVM. Some notions of SVM [46] are shown in Table 1.

<table>
<thead>
<tr>
<th>Notions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(X,d)$</td>
<td>a compact metric space</td>
</tr>
<tr>
<td>$Y$</td>
<td>label space, $Y = {-1, 1}$</td>
</tr>
<tr>
<td>$h$</td>
<td>two-class classifier which labels each point, $x \in X$ with a value $y \in Y$, $h : X \rightarrow Y$</td>
</tr>
<tr>
<td>$p$</td>
<td>probability distribution on $Z = X \times Y$</td>
</tr>
<tr>
<td>$(x,y)$</td>
<td>random variable on $Z = X \times Y$</td>
</tr>
<tr>
<td>$R(g)$</td>
<td>classification error of a classifier $h$, $R(g) = p(h(x) \neq y)$</td>
</tr>
<tr>
<td>$K$</td>
<td>kernel, $X \times X \rightarrow \mathbb{R}$</td>
</tr>
<tr>
<td>$H_k$</td>
<td>reproducing kernel Hilbert space $K$</td>
</tr>
<tr>
<td>$C$</td>
<td>regularisation parameter</td>
</tr>
<tr>
<td>$C(X)$</td>
<td>the space of continuous function on $X$ with the norm $|f|<em>\infty = sup</em>{x \in X}</td>
</tr>
<tr>
<td>$k$</td>
<td>$k = sup_{x \in X} \sqrt{K(x,x)}$</td>
</tr>
</tbody>
</table>
Based on the above notions, some basic definitions were given. A function is identified as a Mercer kernel when one \( K \) is continuous, symmetric, and positive semi-definite; and two, for any finite set of distinct points \( x_i, x_j \in X \), the matrix \( K(x_i, x_j) \) is positive semi-definite. \( H_k \) is defined as the closure of the linear span of the set of functions \( \{ K(x, \cdot): x \in X \} \) with the dot product \( \langle \cdot, \cdot \rangle_{H_k} = \langle \cdot, \cdot \rangle_k \) satisfying \( \langle K_{x_i}, K_{x_j} \rangle_k = K(x_i, x_j) \). The reproducing property takes the form \( \langle K_{x}, f \rangle_k = f(x), \forall x \in X, \forall f \in H_k \). For a function \( g: X \rightarrow R \), the sign function is defined as \( \text{sign}[g(x)] = 1 \) if \( g(x) \geq 0 \), and \( \text{sign}[g(x)] = -1 \) if \( g(x) < 0 \).

The above reproducing property tells that \( \log \| f \|_\infty \leq k \| f \|_K, \forall f \in H_k \). The SVM classifier associated with the Mercer kernel \( K \) is defined as \( \langle f \rangle_k \), and \( f_z \) is defined as Equation (1):

\[
f_z = \arg\min_{f \in H_k} \frac{1}{2} \| f \|^2_K + C \sum_{i=1}^{n} \xi_i
\]

s.t. \( y_i f(x_i) \geq 1 - \xi_i \),

\( \xi_i \geq 0 \),

\( 1 \leq i \leq m \)

3.2. Incremental Support Vector Machine with Adaptation Strategy

The above SVM algorithm cannot learn under concept drift. This paper extended it with a domain adaptation strategy and proposed an ISVM_DD algorithm.

The SVM classifier is \( (\vec{W}, b) \) and discriminant function is \( f(\vec{X}) = \vec{W} \cdot \vec{X} + b \), and the classification decision function is \( L(\vec{X}) = \text{sign}(f(\vec{X})) \). In incremental learning, data are in the form of a sequence, which means there is constantly a new chunk of data. Suppose the first chunk of data is the source domain. If a new chunk of data has the same data distribution as the first one, the new chunk of data will also be divided into the source domain. In contrast, a new chunk of data with different data distribution will be defined as the target domain. We assumed that if two domains are related, then the \( \vec{W} \) of them are similar. Therefore, we introduce \( \mu \| \vec{W}_i - \vec{W}_s \|^2 \) to the \( f(\vec{X}) \) to adapt the SVM from the source domain to the target domain. The \( \| \vec{W}_i - \vec{W}_s \|^2 \) refers to the diversity between the source and target domain models, and if the diversity is larger, its value will be larger. The parameter \( \mu \) controls the degree of adaptation.

Suppose the SVM classifier built from source domain data is \( (\vec{W}_s, b_s) \). Select \( n \) samples from the target domain, and then use source domain knowledge and these \( n \) samples to get an adaptive SVM classifier \( (\vec{W}_t, b_t) \). We transferred this adaptation process to the following optimisation problem as Equation (2):

\[
\min_{\vec{W}_i, b_i} \frac{1}{2} \| \vec{W}_i \|^2 + C_i \sum_{i=1}^{n} \xi_i^t + \mu \| \vec{W}_i - \vec{W}_s \|^2
\]

s.t. \( y_i^t ((\vec{W}_i \cdot \vec{x}_i^t) + b_i) \geq 1 - \xi_i^t \),

\( \xi_i^t \geq 0 \), \( i = 1, 2, \cdots, n \)

To solve this optimisation problem, we introduced the Lagrange parameter \( \vec{\alpha} = (\alpha_1, \alpha_2, \cdots, \alpha_n)^T \) and \( \vec{\beta} = (\beta_1, \beta_2, \cdots, \beta_n)^T \). The Lagrange form corresponds to the optimisation problem, and Equation (2) can be transformed into Equation (3):

\[
L(\vec{W}_i, b_i, \xi_i^t, \vec{\alpha}, \vec{\beta}) = \frac{1}{2} \| \vec{W}_i \|^2 + C_i \sum_{i=1}^{n} \xi_i^t + \mu \| \vec{W}_i - \vec{W}_s \|^2 - \sum_{i=1}^{n} \alpha_i [y_i^t ((\vec{W}_i \cdot \vec{x}_i^t) + b_i) - 1 + \xi_i^t] - \sum_{i=1}^{n} \beta_i \xi_i^t
\]
We can solve Formula (2) by solving its dual problem, as shown in Equation (4):

$$\max_{\alpha} \min_{\vec{W}_t, b, \vec{c}_i} L(\vec{W}_t, b, \vec{c}_i, \vec{a}, \vec{\beta})$$

(4)

To solve Equation (4), we need to solve $\min_{\vec{W}_t, b, \vec{c}_i} L(\vec{W}_t, b, \vec{c}_i, \vec{a}, \vec{\beta})$. Firstly, setting the partial derivative of the (3) to the original variable $\vec{W}_t, b$ and $\vec{c}_i$ is 0, and representing variable $\vec{W}_t, b$ and $\vec{c}_i$ by $\vec{a}$:

$$\frac{\partial L(\vec{W}_t, b, \vec{c}_i, \vec{a}, \vec{\beta})}{\partial \vec{W}_t} = \vec{W}_t + 2\mu(\vec{W}_t - \vec{W}_s) - \sum_{i=1}^{n} a_i y_i x_i = 0$$

(5)

$$\frac{\partial L(\vec{W}_t, b, \vec{c}_i, \vec{a}, \vec{\beta})}{\partial b} = -\sum_{i=1}^{n} a_i y_i = 0$$

(6)

$$\frac{\partial L(\vec{W}_t, b, \vec{c}_i, \vec{a}, \vec{\beta})}{\partial \vec{c}_i} = C_t - a_i - \beta_i = 0$$

(7)

Then, taking the result of (5)–(7) back into (3) and simplifying Equation (4) to Equation (8):

$$\max_{\vec{a}} \left[ \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j y_i y_j (x_i^t x_j^t) \right] + \sum_{i=1}^{n} a_i \left( 1 - 2y_i \frac{(x_i^t \vec{W}_t)}{2\mu + 1} \right) + \frac{\mu}{2\mu + 1} \left| \vec{W}_s \right|^2$$

s.t. $\sum_{i=1}^{n} a_i y_i = 0$

$C_t - a_i - \beta_i = 0$

$a_i \geq 0, \beta_i \geq 0$

(8)

At the same time, the $\vec{W}_t$ can be represented as (9):

$$\vec{W}_t = \frac{\sum_{i=1}^{n} a_i y_i x_i + 2\mu \vec{W}_s}{2\mu + 1}$$

(9)

Then, we can obtain the dual problem of Equation (8) as Equation (10):

$$\min_{\vec{a}} \left[ \frac{1}{2(2\mu + 1)} \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j y_i y_j (x_i^t x_j^t) - \sum_{i=1}^{n} a_i \left( 1 - 2y_i \frac{(x_i^t \vec{W}_t)}{2\mu + 1} \right) - \frac{\mu}{2\mu + 1} \left| \vec{W}_s \right|^2 \right]$$

s.t. $\sum_{i=1}^{n} a_i y_i = 0$

$0 \leq a_i \leq C_t, i = 1, 2, \ldots, n$

(10)

Due to the adaptive discriminant function $f(\vec{x}) = \vec{W}_t \cdot \vec{x} + b_t$, and the hyperplane is $\vec{W}_t \cdot \vec{x} + b_t = 0$. We introduced kernel function $K(\cdot)$ to Equation (10), and $b_t$ is as Equation (11):

$$b_t = y_i - \sum_{i=1}^{n} \frac{a_i y_i}{2\mu + 1} K(\vec{x}_i \cdot \vec{x}_j) - \frac{2\mu}{2\mu + 1} K(\vec{W}_s \cdot \vec{x}_j)$$

(11)

Then, use the SMO (sequential minimal optimisation) algorithm to solve this problem [47].
3.3. Model Ensemble Strategy

Based on the above SVM algorithm and adaptation strategy described, the detailed steps of the model ensemble algorithm are presented in Algorithm 1. It is hypothesised that $k$ datasets arrive in chronological sequence. The algorithm first builds an SVM $f_1$ from the first dataset $D_1$, and preserving $f_1$ in the model set could store $M$ models at maximum. Then, when a new dataset $D_2$ arrives, an SVM $f_2$ is built from $D_2$. All previous SVMs are adapted in a model set to $D_2$. Finally, an ensemble of these adapted SVMs and $f_2$ are conducted by a dynamic weight mechanism. Weight updating is according to the error of each SVM on the current dataset. SVM with smaller errors would be assigned with a larger weight. The error of $k$th SVM $f_k$ on the current dataset was calculated according to Equation (12), $m_t$ was the number of instances in the current dataset $D_t$:

$$
\varepsilon_k = \frac{\text{count}(f_k(x_i) \neq y_i)}{m_t}, \quad 1 \leq i \leq m_t
$$

(12)

We performed the transformation of $\varepsilon_k$ to $B_k$, and the smaller $\varepsilon_k$ is the bigger $B_k$ would be. It was calculated according to Equation (13):

$$
B_k = \frac{1}{2} \log \frac{1 - \varepsilon_k}{\varepsilon_k}
$$

(13)

Then, the weight of $f_k$ was updated according to Equation (14):

$$
w^t_k = w^{t-1}_k \times B^t_k
$$

(14)

Then, the final ensemble model $F^t$ at time $t$ was as Equation (15):

$$
F^t = \text{sign}\left(\sum_{k=1}^{M} f_k(X^t) \times w^t_k\right)
$$

(15)

The accuracy of $F^t$ on $D^t$ was calculated according to Equation (16):

$$
\text{accuracy}(F^t) = 1 - \frac{\text{count} \left( F^t(x^t_j) \neq y^t_j \right)}{m_t}, \quad j = 1, \ldots, m
$$

(16)

Algorithm 1: Model ensemble strategy.

Inputs:
- Dataset arrived before time $t, D = \{D^1, D^2, \ldots, D^{t-1}, D^t\}$
- Sequence of $m_i$ examples at time $i$
- New arriving dataset at time $t$: $D_t$
- Learning algorithm SVM

Initialise: $w^0_k = 1$

Do:
- for each $i = 1, 2, \ldots, t - 1, t$
  - Training $f^t$ from $D^t$;
  - if $i \leq M$:
    - add $f^t$ into model set directly;
    - update $w^t_k$ according to $w^{t-1}_k$ and $\varepsilon^t_k$
  - else:
    - add $f^t$ into model set temporarily
    - remove one $f$ out to maximum $\text{div}(S)$
    - update $w^t_k$ according to $w^{t-1}_k$ and $\varepsilon^t_k$

Output: New model set
4. Results and Discussion

Two computational experiments are conducted to assess the performance of the algorithm proposed and the effect of different kinds of concept drift. The algorithm is then compared with SVM and ISVM without domain adaptation. Five real datasets are used to compare the performance of algorithms, and two synthetic datasets are used to compare the effect of different concept drifts. In this study, Matlab software was used to analyse the data distribution. Python was used to generate Synthetic data and train models.

4.1. Dataset

4.1.1. Real Data

Five generic real datasets are used to test the performance of the proposed algorithm ISVM_DD. The five datasets have different dimensions and scales:

1. Clean Data is a dataset with high dimensions and a small scale. It consists of 476 instances, and each instance has 166 features. For all instances, there are two kinds of classes;
2. Credit Data is a dataset with high dimensions and a large scale. It consists of 6000 instances, each with 65 features about credit card information. For all instances, there are two kinds of classes about whether or not to approve credit;
3. Mushroom Data is a dataset with low dimensions and a large scale. It consists of 5644 instances, and each instance has 23 features of the characteristics of a mushroom. For all instances, there are two classes about whether it is poisonous;
4. Spambase Data is a high-dimension and medium-scale dataset. It consists of 4601 instances, each with 57 features about keywords of a message. For all instances, there are two classes about whether it is spam;
5. Waveform Data is a dataset with a middle dimension and medium scale. It consists of 3345 instances, and each instance has 40 features of the characteristics of the waveform. For all instances, there are two classes of the waveform catalogue.

All of the real datasets are generated with five data chunks, shown in Table 2:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Example Number</th>
<th>Feature Number</th>
<th>Label</th>
<th>Chunk Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clean Data</td>
<td>475</td>
<td>166</td>
<td>2</td>
<td>95</td>
</tr>
<tr>
<td>Credit Data</td>
<td>6000</td>
<td>16</td>
<td>2</td>
<td>1200</td>
</tr>
<tr>
<td>Mushroom Data</td>
<td>5600</td>
<td>22</td>
<td>2</td>
<td>1120</td>
</tr>
<tr>
<td>Spambase Data</td>
<td>4600</td>
<td>57</td>
<td>2</td>
<td>920</td>
</tr>
<tr>
<td>Waveform Data</td>
<td>3300</td>
<td>40</td>
<td>2</td>
<td>660</td>
</tr>
</tbody>
</table>

4.1.2. Synthetic Data

Since we cannot obtain real data distribution, synthetic data are used to compare the performance of different types and degrees of concept drift. We selected two additional concept data, SEA (streaming ensemble algorithm), moving hyperplane concepts, and CIR (circle) concepts.

1. SEA moving hyperplane concepts [21] have three features $x_1$, $x_2$, and $x_3$. Their value is between 0 and 10. Feature $x_1$ and feature $x_2$ are relevant, while $x_3$ is a noisy feature with a random value. The class label of data in this concept is determined by Equation (17).

$$ax_1 + bx_2 \leq \theta$$  \hspace{1cm} (17)

To simulate the concept drift, set different values of $\theta$.

2. CIR concepts [21] apply a circle as the decision boundary in a 2-D feature space. To simulate the concept drift, we set a different radius of the circle, that is, Equation (18).
\[(x_1 - a)^2 + (x_2 - b)^2 \leq \theta\] (18)

In the experiment, each dataset is divided into two parts according to the degree of concept drift. SEA_ONE and CIR_ONE have a small changing degree, while SEA_TWO and CIR_TWO have a great changing degree. Under these conditions, each dataset was randomly generated. The result is shown in Table 3.

<table>
<thead>
<tr>
<th>Drift Type</th>
<th>Fixed-Parameter</th>
<th>Drift Rate</th>
<th>Drift Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEA_ONE</td>
<td>A = 1, b = 1</td>
<td>Low</td>
<td>(\theta = 10, 8, 6, 9, 11)</td>
</tr>
<tr>
<td>SEA_TWO</td>
<td></td>
<td>High</td>
<td>(\theta = 12, 6, 11, 4, 15)</td>
</tr>
<tr>
<td>CIR_ONE</td>
<td>A = 0, b = 0</td>
<td>Low</td>
<td>(\theta = 3, 2, 1, 4, 6)</td>
</tr>
<tr>
<td>CIR_TWO</td>
<td></td>
<td>High</td>
<td>(\theta = 1, 6, 2, 7, 3)</td>
</tr>
</tbody>
</table>

The difference in distribution between SEA_ONE and SEA_TWO is shown in Figure 1. For SEA_ONE, \(\theta\) is 10, 8, 6, 9, 11 in turn, while for SEA_TWO, \(\theta\) is 12, 6, 11, 4, 15. It can be seen that the dataset in SEA_ONE is closer than the dataset in SEA_TWO.

![Figure 1](image1.png)

**Figure 1.** Concept distribution of SEA_ONE and SEA_TWO with different \(\theta\).

The difference in distribution between CIR_ONE and CIR_TWO is shown in Figure 2. For CIR_ONE, \(\theta\) is 3, 2, 1, 4, 6 in turn, and for CIR_TWO, \(\theta\) is 1, 6, 2, 7, 3 in turn. It can be seen that the dataset in CIR_ONE is closer than the dataset in CIR_TWO.

![Figure 2](image2.png)

**Figure 2.** Concept distribution of CIR_ONE and CIR_TWO with different \(\theta\).
4.2. Result Analysis

4.2.1. Results on Real Data

Catastrophic forgetting is a critical standard for judging an incremental learning algorithm. If catastrophic forgetting occurs, the incremental learning algorithm cannot maintain previous knowledge, which means it is not learning incrementally.

In this experiment, we randomly divide the experimental data into a training set and a test set, of which 90% is the training set and 10% is the test set. The data of the training set are randomly divided into five subsets, S1–S5, which are added to the model incrementally.

In this experiment, we test the performance of the model on both new data and previous data after each fine-tuning of the model on new data. The training and testing results of the proposed ISVM_DD algorithm on five datasets are shown in Tables 4–8. Train 1 represents the model training. Train 2, train 3, train 4, and train 5 represent the fine-tuning of the model on new data S2, S3, S4, and S5.

Table 4. Result of ISVM_DD on clean data.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Accuracy on Clean Data (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train 1</td>
</tr>
<tr>
<td>S1</td>
<td>57.89%</td>
</tr>
<tr>
<td>S2</td>
<td>—</td>
</tr>
<tr>
<td>S3</td>
<td>—</td>
</tr>
<tr>
<td>S4</td>
<td>—</td>
</tr>
<tr>
<td>S5</td>
<td>—</td>
</tr>
</tbody>
</table>

Table 5. Result of ISVM_DD on credit data.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Accuracy on Credit Data (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train 1</td>
</tr>
<tr>
<td>S1</td>
<td>86.08%</td>
</tr>
<tr>
<td>S2</td>
<td>—</td>
</tr>
<tr>
<td>S3</td>
<td>—</td>
</tr>
<tr>
<td>S4</td>
<td>—</td>
</tr>
<tr>
<td>S5</td>
<td>—</td>
</tr>
<tr>
<td>Test</td>
<td>83.83%</td>
</tr>
</tbody>
</table>

Table 6. Result of ISVM_DD on mushroom data.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Accuracy on Mushroom Data (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train 1</td>
</tr>
<tr>
<td>S1</td>
<td>95.12%</td>
</tr>
<tr>
<td>S2</td>
<td>—</td>
</tr>
<tr>
<td>S3</td>
<td>—</td>
</tr>
<tr>
<td>S4</td>
<td>—</td>
</tr>
<tr>
<td>S5</td>
<td>—</td>
</tr>
<tr>
<td>Test</td>
<td>91.05%</td>
</tr>
</tbody>
</table>

Table 7. Result of ISVM_DD on spambase data.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Accuracy on Spambase Data (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train 1</td>
</tr>
<tr>
<td>S1</td>
<td>83.58%</td>
</tr>
<tr>
<td>S2</td>
<td>—</td>
</tr>
<tr>
<td>S3</td>
<td>—</td>
</tr>
<tr>
<td>S4</td>
<td>—</td>
</tr>
<tr>
<td>S5</td>
<td>—</td>
</tr>
<tr>
<td>Test</td>
<td>72.28%</td>
</tr>
</tbody>
</table>
Table 8. Result of ISVM_DD waveform data.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Accuracy on Waveform Data (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train 1</td>
</tr>
<tr>
<td>S1</td>
<td>92.67%</td>
</tr>
<tr>
<td>S2</td>
<td>—</td>
</tr>
<tr>
<td>S3</td>
<td>—</td>
</tr>
<tr>
<td>S4</td>
<td>—</td>
</tr>
<tr>
<td>S5</td>
<td>—</td>
</tr>
<tr>
<td>Test</td>
<td>92.52%</td>
</tr>
</tbody>
</table>

The first row of Table 4 shows the classification accuracy of the five models on the first dataset, S1. It can be seen that from model1 to model5, the classification accuracy of S1 is consistent, which means there is no catastrophic forgetting of information. The last row shows the test accuracy of the model on the latest dataset after each fine-tuning. It can be seen that each model can achieve good accuracy on the test set of the latest dataset, which means the model has learned new information after fine-tuning.

It can be seen from S1 in Table 4 that the training accuracy from training 1 to training 5 keeps stable, which means there is no catastrophic forgetting of information. Similarly, there is no catastrophic forgetting of information for other subsets from Table 4 to Table 8. The knowledge learned from the old dataset is not forgotten when learning new datasets.

Tables 4–8 show that with the input data (S1, S2, S3, S4, and S5) arriving and learning new datasets, the accuracy of the test set is gradually improving, which means it can learn new knowledge from new datasets. This also proves that as the training increases, more knowledge is learned.

The result in Tables 4–8 shows the knowledge learned from the old dataset is not forgotten when learning new datasets. With the input data arriving and learning from new datasets, the test accuracy is improving gradually, which means it can learn new knowledge from new datasets. This also proves that as the training increases, more knowledge is learned.

The average test accuracy of SVM, ISVM, and ISVM_DD on five datasets is also compared, and the result is shown in Figure 3 and Table 9.
The knowledge learned from the old dataset is not forgotten when learning new datasets. This also proves that as the training increases, more new datasets, the accuracy of the test set is gradually improving, which means it can learn new knowledge from new datasets. This also proves that as the training increases, more new datasets, the accuracy of the test set is gradually improving, which means it can learn new knowledge from new datasets. This also proves that as the training increases, more new datasets, the accuracy of the test set is gradually improving, which means it can learn new knowledge from new datasets. This also proves that as the training increases, more new datasets, the accuracy of the test set is gradually improving, which means it can learn new knowledge from new datasets. This also proves that as the training increases, more new datasets, the accuracy of the test set is gradually improving, which means it can learn new knowledge from new datasets. This also proves that as the training increases, more new datasets, the accuracy of the test set is gradually improving, which means it can learn new knowledge from new datasets. This also proves that as the training increases, more new datasets, the accuracy of the test set is gradually improving, which means it can learn new knowledge from new datasets.

Table 9. Result of algorithms on five datasets.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Clean Data</th>
<th>Credit Data</th>
<th>Mushroom Data</th>
<th>Spambase Data</th>
<th>Waveform Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>52.53%</td>
<td>81.34%</td>
<td>91.86%</td>
<td>75.67%</td>
<td>92.79%</td>
</tr>
<tr>
<td>ISVM</td>
<td>52.32%</td>
<td>81.52%</td>
<td>92.23%</td>
<td>77.38%</td>
<td>93.42%</td>
</tr>
<tr>
<td>ISVM_DD</td>
<td>52.63%</td>
<td>84.08%</td>
<td>95.03%</td>
<td>80.21%</td>
<td>94.31%</td>
</tr>
</tbody>
</table>
It shows that ISVM_DD gains performance improvements of about 3% on 4 out of 5 datasets. It means incremental learning could perform better when data has concept drift than batch learning. The comparison between ISVM and ISVM_DD shows that domain adaptation could decrease concept drift.

4.2.2. Results on Synthetic Data

From the results of real data, the performance of different datasets differs. It is somewhat related to the type and changing rate of concept drift. However, the type and changing rate of concept drift in real data could not be known, so two synthetic datasets are used to investigate the influence of concept drift type and changing rate.

There are two kinds of concept drift: SEA moving hyperplane concepts and circle concepts, and the corresponding datasets are SEA and CIR. Each dataset is divided into two parts according to the changing rate of concept drift. SEA_ONE and CIR_ONE have low changing rates, while SEA_TWO and CIR_TWO have high changing rates. The result is shown in Table 10 and Figure 4.

Table 10. The average accuracy of different types and different degrees of concept drift.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>SEA_ONE</th>
<th>SEA_TWO</th>
<th>CIR_ONE</th>
<th>CIR_TWO</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>89.07%</td>
<td>88.65%</td>
<td>81.33%</td>
<td>80.32%</td>
</tr>
<tr>
<td>ISVM</td>
<td>90.46%</td>
<td>88.74%</td>
<td>83.27%</td>
<td>81.03%</td>
</tr>
<tr>
<td>ISVM_DD</td>
<td>94.77%</td>
<td>90.24%</td>
<td>84.88%</td>
<td>82.69%</td>
</tr>
</tbody>
</table>

Table 10 shows that for different concept drifts, all these three algorithms show different performances, and ISVM_DD performs better than SVM and ISVM on all four datasets. However, for concept drift with a high changing rate, ISVM and ISVM_DD decrease average accuracy.

Figure 4 shows that for different types of concept drift, such as SEA and CIR, the algorithm ISVM-DD shows a similar difference. However, for different drift degrees, it offers significantly different performance. Figure 4 shows a slight drift degree, and the algorithm shows good performance, while for a great drift degree, the performance has an obvious decrease. When the drift degree is great, the distance between two concepts is so significant that one adaptation is not enough to connect them.

5. Conclusions

Incremental learning is an effective approach to addressing increasing and changing data. However, concept drift is still a crucial problem limiting incremental learning. In
this paper, to decrease the effect of concept drift in incremental learning, we proposed an ISVM_DD algorithm. On the one hand, the proposed ISVM_DD is able to acquire new knowledge from newly added training data while retaining the knowledge learned before without catastrophic forgetting through domain adaptation. On the other hand, by the model selection, the proposed ISVM_DD can forget obsolete information without corrupting still valid, already learned information. The experiment indicates the ISVM_DD achieve better performance compared to previous algorithms.

Furthermore, we studied the effect of the type and degree of concept drift on ISVM_DD. We found that the type and changing rate of concept drift will affect the performance, especially the degree. When dramatic concept drift occurs, the performance of ISVM_DD will decrease. It may be because just once domain adaptation could not connect the source domain and target domain, which is called a transitive transfer learning problem [48]. In the future, we will investigate multiple domain adaptation among some intermediate domains to decrease dramatic concept drift.

Author Contributions: Conceptualisation, J.T.; methodology, J.T.; validation, J.T.; formal analysis, J.T.; writing—original draft preparation, J.T.; writing—review and editing, L.L. and K.-Y.L.; supervision, L.L.; project administration, L.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the National Natural Science Foundation of China under grants 72171172 and 62088101; Shanghai Municipal Science and Technology, China Major Project under grant 2021SHZDZX0100; Shanghai Municipal Commission of Science and Technology, and China Project under grant 1951132101.

Institutional Review Board Statement: No applicable.

Informed Consent Statement: No applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

3. Gâlmeanu, H.; Andonie, R. Weighted Incremental-Decremental Support Vector Machines for concept drift with shifting window. *Neural Netw.* 2022, 152, 528–541. [CrossRef] [PubMed]
15. Liang, Z.; Li, Y.F. Incremental support vector machine learning in the primal and applications. *Neurocomputing* 2009, 72, 2249–2258. [CrossRef]
40. Wu, Y.; Ma, X. Alarms-related wind turbine fault detection based on kernel support vector machines. *J. Eng.* 2019, 18, 4980–4985. [CrossRef]
43. Arslan, G.; Madran, U.; Soyoğlu, D. An Algebraic Approach to Clustering and Classification with Support Vector Machines. *Mathematics* 2022, 10, 128. [CrossRef]
44. Liu, X.; Zhao, B.; He, W. Simultaneous Feature Selection and Classification for Data-Adaptive Kernel-Penalized SVM. *Mathematics* 2020, 8, 1846. [CrossRef]

45. Gonzalez-Lima, M.D.; Ludeña, C.C. Using Locality-Sensitive Hashing for SVM Classification of Large Data Sets. *Mathematics* 2022, 10, 1812. [CrossRef]

