

Learning from data streams via online transduction

Shen-Shyang Ho and Harry Wechsler

Department of Computer Science
George Mason University
4400 University Drive
MSN 4A5
Fairfax, VA 22030

E-mail: {sho, wechsler}@cs.gmu.edu

Abstract

A practical issue in the existing transduction methods is expensive and inefficient computation compared to induction methods. This has hindered the use of transduction methods in temporal and real-time data mining.

In this paper, we introduce a fast incremental transductive confidence machine (TCM) based on adiabatic incremental support vector machine (SVM) such that critical information from current transduction trial is stored for later use. The algorithm is empirically shown to be computationally efficient and its performance is consistent with standard TCM implementation.

Besides being a classifier, TCM provides additional useful statistical information about the data that it processed. These information can be useful for temporal and real-time data mining. We demonstrate the feasibility and usefulness of using such statistical information for stream-based active learning.

1. Introduction

A simple interpretation of transduction is that the unknown values at some points of interest are estimated *directly* from the training data [16]. This is different from induction when a general rule is inferred from the training data and predictions are based on this general rule. The practical distinction between transduction and induction is whether we extract and store the general rule or not [19]. Current transduction methods are computationally expensive such that any improvement in performance (compared to induction, if any) cannot justify this expensive computation. Hence, transduction methods are seldom (or never) been taken seriously as viable (temporal or real-time) data

mining techniques.

One such transduction method, called transductive confidence machine (TCM), proposed by Gammerman and his colleagues [2] [3], predicts the classification of an object together with a measure of confidence using algorithmic randomness theory in an offline setting. Their first proposed algorithm is based on support vector machines (SVM). TCM (independent of the base classifier) is later proven to be well-calibrated (in the sense that in the long run, the number of wrong predictions is bounded) by Vovk [18] in an online setting, i.e., examples are presented to a learner one by one, with i.i.d. assumption.

Transduction in general, and TCM in particular, are computationally expensive and are only practical for small data sets. To improve the computational speed, Saunders et. al. [14] used a hashing function to split the training set into smaller subsets, of roughly equal size, which are used to construct a number of support vector machines in the TCM implementation. This solution, however, assumes the availability of all the training examples, which is not feasible in an on-line setting. A modification of TCM, called inductive confidence machine (ICM) [10], is also proposed which sacrifices some prediction accuracy for efficiency.

In this paper, we introduce a fast incremental TCM which integrates incremental SVM by *adiabatic increment*, i.e., preserving the Karush-Kuhn-Tucker conditions on all previously seen training examples [1], into TCM. This implementation, without splitting up the current training set into subsets and without sacrificing prediction accuracy, improves the computational efficiency of TCM significantly.

Besides being a classifier, TCM provides additional useful statistical information about the data that it processed. We show the feasibility and usefulness of using such statistical information for stream-based active learning.

In Section 2, we review some fundamental concepts in

algorithmic randomness and p-value for understanding of TCM. Section 3 describes the on-line (randomized) TCM and the idea of region predictor. In Section 4, we review different implementation strategies for incremental support vector machine and in Section 5, we briefly review the adiabatic increment for incremental SVM. In Section 6, we discuss the difference between online and incremental learning. In Section 7, we discuss online transduction and the use of incremental classifier to improve its computational efficiency. In Section 8, we introduce our implementation of the online incremental TCM. In Section 9, we perform experiments to compare the computational efficiency and performance of our algorithm with the standard TCM implementation and an implementation of TCM using a different incremental strategy. We also demonstrate its application on stream-based active learning. Throughout the rest of the paper, we assume basic familiarity of SVMs [16].

2. Randomness, p-value and Transductive Confidence Machine (TCM)

Vovk and et. al. [17] first proposed the application of (algorithmic) randomness on machine learning problems. The confidence measure used in TCM is based upon universal tests for randomness, or their approximation. A Martin-Löf randomness deficiency [8] based on such tests is a universal version of the standard statistical notion of p-values.

By reformulating the Martin-Löf randomness test, we have a function $t : Z^* \rightarrow [0, 1]$ called a *p-value function* if

1. Let P_n be a set of probability distribution in Z^n . For all $n \in \mathcal{N}$ and $r \in [0, 1]$ and all $P \in P_n$,

$$P\{z \in Z^n : t(z) \leq r\} \leq r \quad (1)$$

2. t is semi-computable from above, i.e. there exists a computable sequence of computable functions $t_i : Z^* \rightarrow [0, 1], i = 1, 2, \dots$ such that $t(z) = \inf_i t_i(z)$ for all $z \in Z^*$.

where Z is the set of all possible labeled examples, Z^n is the set of all sequences of n labeled examples of the form

$$\{z_1, z_2, \dots, z_n\} = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

where x_i is an object and y_i is its label for $i = 1, 2, \dots, n$ and Z^* is the set of all finite sequences of labeled examples [11]. This p-value function is practically equivalent to the standard statistical notion of p-values. A particular p-value for a sequence of labeled examples is also called the *randomness level* of that sequence [3].

In the literature on significance testing, the p-value is often defined as the probability of observing a point in the

sample space which can be considered as extreme as, or more extreme than, the observed sample [20]. This calculation requires a well defined stochastic ordering of the sample space which can be provided by a test statistic. The p-value serves as a measure of how well the data support or discredit a null hypothesis: the smaller the p-value, the greater the evidence against the null hypothesis. In other words, smaller p-values of the observed level of significance favor the alternative hypothesis and larger values favor the null hypothesis.

In order to construct a valid p-value function (the main component in a TCM), an ordering function called *strangeness measure* [3] (or nonconformity score [19]) is required. The strangeness, α_i , of a particular labeled example, z_i , corresponds to the uncertainty of that example with respect to all other labeled examples: the higher the measure, the higher the uncertainty. The strangeness measure used depends on the type of base classifier used to construct a TCM. To use support vector machine (SVM) as the base classifier to construct a TCM, the solution, a set of Lagrange multipliers, of the dual problem of the SVM optimization problem is used as the strangeness measure [2]. A strangeness measure for k-nearest neighbor is given in [11].

Consider a sequence of labeled examples, $\{z_1, z_2, \dots, z_{n-1}\}$ with their corresponding strangeness values, $\alpha_1, \alpha_2, \dots, \alpha_{n-1}$ and an unlabeled example, z_n , assigned a particular label and its strangeness value, α_n , we define a p-value function $t : Z^n \rightarrow [0, 1]$ which returns a p-value of z_n (assigned the particular label) by

$$t(z_1, z_2, \dots, z_n) = \frac{\#\{i = 1, \dots, n : \alpha_i \geq \alpha_n\}}{n} \quad (2)$$

which satisfies

$$P\{(z_1, z_2, \dots, z_n) : t(z_1, z_2, \dots, z_n) \leq r\} \leq r$$

for any $r \in [0, 1]$ and for any probability distribution P in Z^n , provided that the strangeness value of an example in the sequence is independent of its position in the sequence. This p-value computation is a simple approximation of the randomness level of the Martin-Löf's randomness test.

Below is the algorithm for the approximation of all p-values for all possible labels that an unlabeled example can be assigned. This algorithm is the essential procedure of a TCM:

Algorithm:

Input: training set $T = \{z_1, z_2, \dots, z_{n-1}\}$ and an unlabeled example $z_n = (x_n, ?)$:

1. FOR $y = 1$ to c (number of possible classes)
 - (a) Label z_n as class y
 - (b) Construct a classifier (e.g. *SVM*) using T and (x_n, y)

- (c) Use SVM to compute strangeness,
 $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_n\}$ for T and (x_n, y)
- (d) Use strangeness, α to compute p-value for
 (x_n, y) (Equation (2))

ENDFOR

- 2. Assign the label with the largest p-value to $z_n = (x_n, ?)$
- 3. Confidence = 1 - second highest p-value
- 4. Credibility = highest p-value

3. Online transductive confidence machine

The basic idea of online TCM is similar to the above procedure. A randomized version of TCM introduced by Vovk [18] is used here so that a prediction error bound for online TCM can be achieved. The computation of p-value for TCM and its randomized version is identical.

TCM is a way to define a *region predictor* [18] (or conformal predictor [19]) from a learning algorithm that can only return a prediction for each unlabeled test input. A region predictor is a function when given an unlabeled test input and a confidence level returns a set of possible predictions with a degree of confidence.

For a particular unlabeled example $z_n = (x_n, ?)$, a training set T , a set of all possible labels Y and $\delta \in (0, 1)$, we can obtain a region predictor:

$$\Gamma_{1-\delta}(T, z_n)$$

which contains $y \in Y$ that satisfy

$$\frac{\#\{i = 1, \dots, n : \alpha_i \geq \alpha_n^y\}}{n} > \delta \quad (3)$$

where $\alpha_1, \dots, \alpha_{n-1}$ are the strangeness measure of the training examples and α_n^y is the strangeness measure of z_n labeled y . Intuitively, we can say that z_n can take any of the labels in Γ at $1 - \delta$ confidence level.

“Uncertain” *region predictor* are region predictor that has more than one label. The region predictor can also be an empty set.

A modification of TCM called *randomized Transductive Confidence Machine (rTCM)* defines a randomized region predictor Γ such that for any label $y \in Y$,

- 1. Include label y in Γ when

$$\frac{\#\{i = 1, \dots, n : \alpha_i > \alpha_n^y\}}{n} > \delta$$

- 2. Do not include y in Γ when

$$\frac{\#\{i = 1, \dots, n : \alpha_i \geq \alpha_n^y\}}{n} \leq \delta$$

- 3. Otherwise, include y in Γ with probability

$$\frac{\#\{i = 1, \dots, n : \alpha_i \geq \alpha_n^y\} - n\delta}{\#\{i = 1, \dots, n : \alpha_i = \alpha_n^y\}}$$

y is included when the above expression is greater than a random number drawn from a uniform distribution on $[0, 1]$.

A rTCM is proven to have an error probability δ at trial $n = 1, 2, \dots$, and this error probability at each trial is independent of other trials [18].

A TCM predicts the label of an example as the one with the largest p-value with a confidence measure equals to one minus the second largest p-value and credibility equals to the largest p-value [3]. If the region predictor is of size larger than one, we know that the prediction has low confidence.

We will show the usefulness of these additional information about each example for stream-based active learning in Section 9. More details and properties of region predictor and [randomized] TCM can be found in [18].

4. Incremental support vector machine

Earlier implementations of incremental SVM make use of the properties of the support vectors and the distribution knowledge of the sample space [21] to reduce the size of the training set as more examples are added. In some cases, only the historical support vectors and the new examples are kept for training the classifier [15]. These methods require re-training the classifier using this new training data-set.

Recently, implementations of incremental SVM that avoid retraining all the training examples have been suggested [4] [7] [1].

Fung and Mangasarian [4] proposed a fast and simple incremental support vector machine that modifies the current linear classifier by “both retiring old data and adding new data”. Their method uses a non-standard SVM formulation that classifies points by assigning them to the closest of two parallel hyper-planes that are pushed apart as far as possible. However, their non-standard SVM does not provide the strangeness measure (Lagrange multiplier for each examples) required by TCM using SVM as the base classifier.

Kivinen et. al. [7] considered online learning in a Reproducing Kernel Hilbert Space using classical stochastic gradient descent within a feature space and derive a rate of convergence and error bound. However, the error bound does not corroborated well with their experiments [7].

We use the incremental SVM implementation by adiabatic increment [1] since

- 1. it can be easily integrated into TCM.

- it constructs the exact solution recursively as a new example is added.

We review the main idea of adiabatic increment in the next section.

5. Incremental SVM by adiabatic increments

Given a set of labeled examples

$$\{z_1, \dots, z_n\} = \{(x_1, y_1), \dots, (x_n, y_n)\}$$

where $y_i \in \{-1, 1\}$ for $i = 1, 2, \dots, n$ such that the two classes of examples are not linearly separable, an optimal hyperplane is constructed by minimizing the functional

$$W(\xi) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i \quad (4)$$

where w is the hyperplane normal vector, the constant $C > 0$ and $W(\xi)$ subjects to the constraints

$$y_i(w \cdot \Phi(x_i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, 2, \dots, n \quad (5)$$

where $\Phi(x_i)$ maps x_i into the feature space and b is an offset. Introducing Lagrange multipliers, $\alpha_i \geq 0, i = 1, \dots, n$, for each of the constraints above, the dual form of the above optimization problem is expressed as

$$\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n y_i y_j \alpha_i \alpha_j \Phi(x_i) \cdot \Phi(x_j) - \sum_{i=1}^n \alpha_i \rightarrow \min_{\alpha_i} \quad (6)$$

$$\sum_{i=1}^n y_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq C, i = 1, \dots, n \quad (7)$$

Apply the Lagrange method again on the dual problem, we get

$$W(\alpha, b) = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n y_i y_j \alpha_i \alpha_j \Phi(x_i) \cdot \Phi(x_j) - \sum_{i=1}^n \alpha_i + b \sum_{i=1}^n y_i \alpha_i \quad (8)$$

Replacing $y_i y_j \Phi(x_i) \cdot \Phi(x_j)$ with Q_{ij} (each element of the kernel matrix, Q),

$$W(\alpha, b) = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i Q_{ij} \alpha_j - \sum_{i=1}^n \alpha_i + b \sum_{i=1}^n y_i \alpha_i \quad (9)$$

From the slack-variables, ξ_i , we get the constraints $\alpha_i \leq C, i = 1, \dots, n$.

$W(\alpha, b)$ is minimized given the Karush-Kuhn-Tucker (KKT) conditions

$$g_i = \frac{\partial W}{\partial \alpha_i} = \sum_{j=1}^n Q_{ij} \alpha_j + y_i b - 1 \quad (10)$$

$$\begin{cases} > 0 & \alpha_i = 0 \\ = 0 & 0 \leq \alpha_i \leq C \\ < 0 & \alpha_i = C \end{cases}$$

$$\frac{\partial W}{\partial b} = \sum_{j=1}^n y_j \alpha_j \equiv 0 \quad (11)$$

The first KKT condition partitions the training data into three sets:

- the set of support vectors that lies on the margin, S , when $0 < \alpha_i < C$.
- the set of support vectors that lies in the margin, E , when $\alpha_i = C$.
- the remaining set, R , when $\alpha_i = 0$.

The main idea of adiabatic increment is that all the examples in the training set do not violate the KKT conditions when a new example z_{n+1} is added to this training set when its α_{n+1} , initially set to zero, is incremented. In order to do so, some ‘‘bookkeeping’’ [1] procedure is necessary.

When α_{n+1} , of the new example, is incremented, the ‘‘bookkeeping’’ procedure will update the values of g_i (Equation (10)) and α_i for $i = 1, 2, \dots, n, n + 1$. In doing so, an example may be transferred from one set to one of the other two sets accordingly.

More derivation details and implementation issues such as the computation of the Jacobian inverse, \mathcal{J} , by recursion, and maintaining of a smaller remaining set, R , to improve computational efficiency, are found in [1].

6. Online vs incremental learning

Before we introduce our TCM implementation in section 8, we will like to point out some distinctions between online and incremental learning. They should not be used interchangeably.

We need to distinguish between the learning *setting* and learning *algorithm*. Two questions can be asked:

- Is the learning *setting* (i) online or (ii) offline ?
- Is the learning *algorithm* (i) incremental or (ii) batch ?

In an online setting, data enter the learner sequentially, as packets of data or individual datum. This is different from the offline setting when the whole training set is assumed to be available to the learner.

For both the online and offline settings, the learning algorithm can be incremental. As the data are iteratively added to the training set to learn a general rule, the rule to be learnt during the current trial is built from some compact information of the previously learnt rule and the new data. This type of learning is *algorithmically incremental*.

Earlier methods [21] [15] on incremental SVM are not algorithmically incremental. They require the re-training of the classifier using all the available training examples at each trial. These are actually *batch algorithms used in an online setting*.

Later methods such as incremental proximal support vector classifier [4], Kivenen and et. al.'s online learning [7] and the incremental SVM by adiabatic increments [1] are all algorithmically incremental. We note that since SVM is a convex optimization problem, a correctly formulated incremental algorithm should always achieve the optimal solution.

7. Online transduction and incremental classifier

In an online setting, the general rule (induced from the current training set) may be modified whenever a new packet of data or an individual datum becomes available. It becomes awkward to talk about the prediction accuracy of a particular rule. Vovk and et. al. [19] point out that “it makes more sense to emphasize the overall frequency of accurate prediction” during the online process. Under this assumption, the error bound of (online randomized) TCM is shown [18] to be independent of other (previous or future) trial(s). In other words, the error bound for the complete online transduction process is not affected by any rule inferred (if any) at any particular point of time.

As we pointed out in Section 1 (Introduction) that the practical distinction between transduction and induction is whether we extract and store the general rule or not. More specifically, there are some critical information required by the transduction methods that is also computed in the induction methods.

Recent developments in incremental (inductive) classifier, such as the adiabatic incremental SVM, has given us some hints on how to go about improving the computational efficiency of transduction.

We conjecture that any algorithmically incremental classifier (e.g. [4] [7]) can either be modified to become viable transduction methods or integrated into existing transduction methods (e.g. TCM) to speed up the transduction process in an on-line setting and still maintain its performance.

So, instead of extracting and storing the general rule from the training set and repeating this process in the online setting, we could compute and store critical information efficiently from the current training set to be used for

any transduction process later.

8. Incremental Online Transductive Confidence Machine

Let T be the current training set with $n - 1$ labeled examples, $\{\alpha, b\}$ be the current solution where α is the set of Lagrange multipliers and b is the offset, \mathcal{J} be the current Jacobian inverse, Q be the kernel matrix, S, E be the current two support vector sets (on and within margin, respectively) and R be the current remaining vector set.

By integrating the procedure of the adiabatic incremental learning into the algorithm for TCM (Section 2), we get:

Algorithm (One transduction process for a new unlabeled example):

Input: $z_n = (x_n, ?), T, \{\alpha, b\}, \mathcal{J}, Q, S, E, R$.

1. Keep a copy of $\{\alpha, b\}, \mathcal{J}, Q, S, E, R$ for re-initialization at the end of each FOR loop
2. FOR $y = 1$ to c (number of possible classes)
 - (a) Label z_n as y (i.e. $z_n = (x_n, y)$)
 - (b) Initialize $\alpha_{z_n}^y$ to zero.
 - (c) Compute $Q_{z_n j}$ for all $j \in S$ (the set of support vectors) and include it into Q .
 - (d) Compute g_{z_n} using Equation (10).
 - (e) If $g_{z_n} > 0$,
 - i. $\alpha_{z_n}^y = 0$, and goto (g)
 - (f) Elseif $g_{z_n} \leq 0$, apply the largest possible increment $\Delta\alpha_{z_n}^y$ so that (the first) one of the following conditions occurs:
 - i. $g_{z_n} = 0$: Add z_n to S , update R accordingly, goto (g).
 - ii. $\alpha_{z_n}^y = C$: Add z_n to E and goto (g).
 - iii. Do “bookkeeping” among S, E and R . Update \mathcal{J} if S changes
 - iv. Repeat (f) until no more changes to S, E and R .
 - (g) Compute p-values for $z_n = (x_n, ?)$ labeled y using $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_{n-1}, \alpha_{z_n}^y\}$.
 - (h) Test whether y is in the region predictor of $z_n = (x_n, ?)$ with confidence level $1 - \delta$ using $rTCM$.
 - (i) Re-initialize $\{\alpha, b\}, \mathcal{J}, Q, S, E, R$ to their respective input values.

ENDFOR

3. Assign the label with the largest p-value to example z_n .
4. Calculate the confidence as $1 -$ second largest p-value.

In the next section, we use the acronym “TCM” to mean the *online (randomized) TCM using SVM as the base classifier*.

9. Experiments

This section describes experimental data concerning the comparative efficiency for our novel adiabatic incremental TCM method using SVM, and its utility for stream-based active learning.

9.1. Comparison of computational cost and classification performance of three different TCM implementations

We generated a binary data-set using Musicant’s NDC (normally distributed clustered) data-set generator [9]. The data is generated as clusters of normally distributed points in \mathbb{R}^n with an adjustable linear separability. The values of each dimension are scaled to range in $[-1, 1]$. The data-set consists of 3000 points in a 32 dimensional input space and the experiments are performed in MATLAB 6.5.

We compare the computational cost and performance of (A) the original TCM, (B) the incremental TCM by keeping the historical support vectors, and (C) the incremental TCM by adiabatic increment that is described in Section 8. As we mentioned in Section 6, incremental SVM can be used in an offline setting. We use the adiabatic incremental SVM (that has comparable computational cost as other QP implementations of SVM in an offline setting) as the base classifier in (A) and (B). For all experiments, δ is set to 0.1 i.e. 90% confidence level. For SVM, we use a Gaussian kernel and C is set to 10.

The computational cost is based on the number of iterations of step 1(f), the adiabatic step, which is the most expensive step in the three implementations. The upper graph in Figure (1) shows the total computational cost of the three implementations at each instance when a new example is included into the training set of the learner. We see that our implementation (C) requires much less number of total iterations and the rate of increment has a much gentler slope than implementation B while the the total iterations of original TCM increases exponentially. From the lower graph in Figure (1), we see that at each streaming data instance, the number of iterations remains small and stable for implementation (C).

We note here that each iteration does not corresponds to a fix time unit since at each iteration some matrix computations are required such that the computation cost depends mainly on the size of the inverse Jacobian, \mathcal{J} (i.e. the number of support vectors) and the number of training examples. Maintaining a smaller set R and assuming a “concept drift” setting (that is likely to occur in real world problems) are some ways to provide an upper bound for the computational cost at each iteration.

We observe from Figure (2) that using adiabatic increment for TCM and original TCM have very similar learning

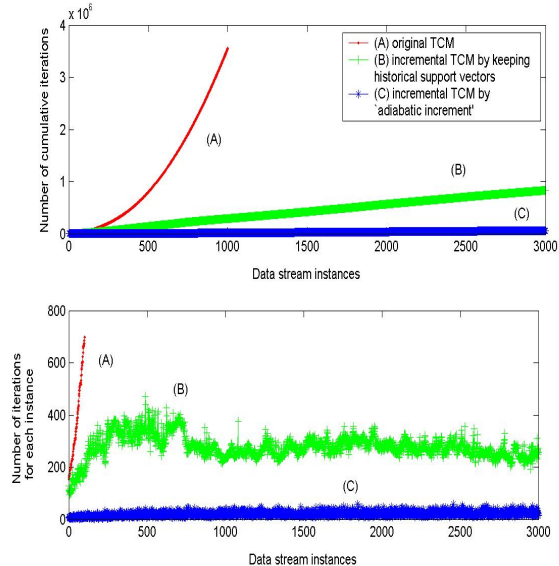


Figure 1. Comparison of the (upper graph) total number of updating iterations and (lower graph) number of updating iterations at each data stream instance required for (A) original TCM, (B) incremental TCM by keeping historical support vectors and (C) incremental TCM by adiabatic increment. (Results for (A) only shown for 1000 instances in the upper graph and 100 instances in the lower graph as the rate of increment is so high that the visualization of the others curves is affected if results for all instances are shown.)

curves for accuracy, error and uncertainty predictions. The slight differences are due to the randomized nature of our TCM implementation.

We also note from Figure (2) that the number of uncertain predictions are much higher than the accurate predictions for implementation (B) while its error predictions are higher than the other two implementations. This is the result of information lost when data instances that are not support vectors previously are discarded, which may become useful later.

Theoretically, the data samples are no longer i.i.d. and the error bound for TCM is no longer valid. Empirically, we observed that despite the fact that the training examples used at each trial are smaller than the other two implementations in a long data stream (i.e. the total computational cost will be lower in a long data stream), its classification performance is dubious.

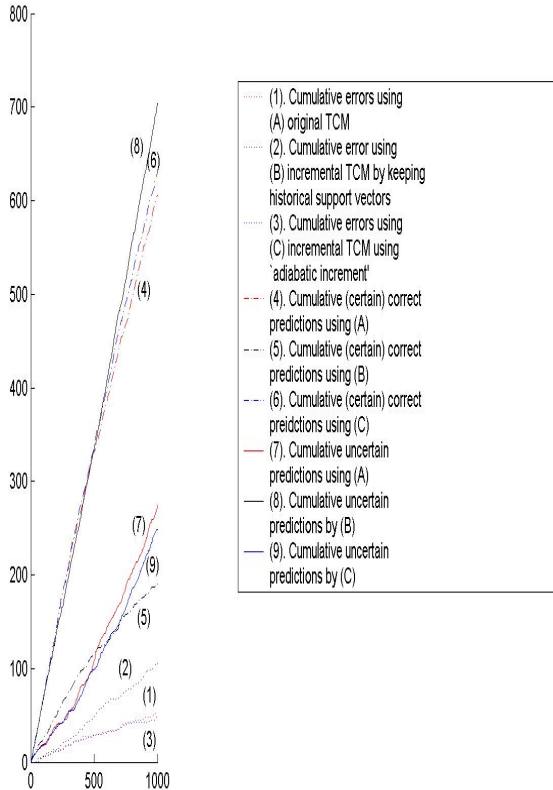


Figure 2. Comparison of the training performance of (A) original TCM, (B) incremental TCM by keeping the historical support vectors and (C) incremental TCM using adiabatic increment. (A) and (C) has very similar learning curves. Slight difference is due to the randomized nature of the algorithm.

9.2. Application of TCM: Stream-based active learning

TCM has been used to perform active learning [5] (in a data stream [6]). For stream-based active learning, unlabeled examples are provided to the learner one by one, and the learner must decide whether or not to request its label and add it to the training set.

The goals for active learning are two fold : (i) less computation (due to smaller training sets) without penalizing the performance of the classifier, and (ii) reducing the human labeling efforts and cost.

We recall that in Section 3, we describe a region predictor as a function, when given an unlabeled test input

and a confidence level, returns a set of possible predictions with a degree of confidence and uncertain region predictor are region predictor that has more than one label. Our stream-based active learning method takes into account the *deviation from uncertain region predictor* of each unlabeled example provided to the learner. This deviation measure makes use of the p-values computed for all possible labels for an unlabeled example by TCM.

Given a binary classification problem, $\mathcal{U}(e) = |p_i - p_j|$, where p_i is the p-value for an unlabeled example e labeled as i and p_j is the p-value for the example labeled as j , is a measure of the deviation from an uncertain region predictor. As $\mathcal{U}(e)$ increases from 0 to 1, the example e becomes more likely to be a particular label.

Selection Criteria: For a given example e , if $\mathcal{U}(e) \leq \eta$ (a threshold value), add e to the training set, T .

The theoretical justification and empirical studies of stream-based active learning (including multi-class problems) can be found in [6].

Here, we use the binary classification problem, Image (18 dimension data-set with 20 folds of training/testing partitions), from the benchmark collection in [12], to demonstrate data stream active learning using TCM.

In our experiment, the learner is first provided with one randomly chosen example from each class. Subsequently, the learner is provided with a stream of unlabeled data. At each trial, a new unlabeled point from the stream is introduced to the learner. The data stream is of length 1300.

In Table 1, we compare the number of examples selected by the active learner and their prediction accuracy at the end of the data stream. About one-fifth of the total data is selected using active learning and the performance is almost as good as using all the examples in the data stream. The total number of examples selected is much less than random sampling and the accuracy estimate of active learning is comparable.

Method	No of selected examples	Accuracy Estimate
Standard	1300	95.65±0.56
Random	652.00±14.98	94.04±0.88
Active	273.30±49.27	94.33±1.05

Table 1. Comparison of 1) using all samples (Standard), (ii) random sampling (Random) and (iii) active learning (Active). Accuracy and number of selected examples are compared. Estimates are averaged over 20 trials.

Figure (3) shows that active learning using TCM has very competitive performance compared to random sampling when the same number of examples are selected.

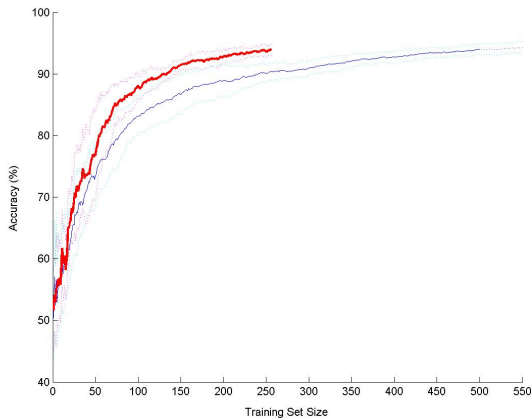


Figure 3. Our active learning (upper solid curve with error bars (dotted curve)) shows very competitive performance compared to the random sampling (lower solid curve) when the same number of examples are selected.

10. Conclusions

The use of transductive learning in temporal and real-time data mining is hindered by the fact that existing transduction methods are expensive and inefficient computation compared to inductive learning techniques. One objective of this paper is to encourage temporal data-mining researchers to explore the use of transduction methods in their works. In order to do so, we introduce a fast incremental transduction method from an existing incremental algorithm and a transduction method to overcome the inefficiency problem of transduction.

Our transduction method is a fast incremental TCM based on adiabatic incremental SVM. Our method is empirically shown to be computationally efficient and its performance is consistent with the standard TCM implementation. Besides being a classifier, TCM provides additional useful statistical information about the data that it has processed. We demonstrate the usefulness and feasibility of such statistical information for stream-based active learning. These information can be useful for other temporal and real-time data mining.

We also show that incremental SVM by keeping historical support vectors when integrated into TCM do not have consistent performance compared to the standard TCM im-

plementation. Hence, the integration of incremental algorithm into transduction methods must be done with caution to achieve both a reduction in the computational cost and a competitive performance.

Transduction can, then, become a viable technique for learning in data stream.

Acknowledgement

The first author would like to thank Dr. Volodya Vovk for the interesting discussions. The first author would also like to thank Dr Alex Gammerman and Dr. Volodya Vovk for the manuscript of their book “Algorithmic learning in a random world”.

References

- [1] Cauwenberghs, G. and Poggio, T. Incremental support vector machine learning, *Advances in Neural Information Processing Systems 13*, MIT Press, 409-415, 2000.
- [2] Gammerman A., Vovk. V. and Vapnik. V. Learning by Transduction. *Uncertainty in Artificial Intelligence*, *Proc of the Fourteenth Conference (1998)*, Madison, Wisconsin, July 1998, 148-155. 1998.
- [3] Gammerman A. and Vovk V. Prediction algorithms and confidence measures based on algorithmic randomness theory, *Theoretical Computer Science* 287, 209-217, 2002.
- [4] Fung G. and Mangasarian O.L. Incremental support vector machine classification. *Proceedings of the second SIAM International Conference on Data Mining*, SIAM, 247-260, 2002.
- [5] Ho S.-S. and Wechsler H. Transductive Confidence Machine for active learning, *Proceedings of Int. Joint Conf. on Neural Networks*, 2, 1435-1440, 2003.
- [6] Ho S.-S. and Wechsler H. Stream-based active learning using algorithmic randomness theory. (Manuscript in preparation).
- [7] Kivinen J., Smola A. and Williamson R. Online learning with kernels, *Advances in Neural Information Processing Systems 14*, MIT Press, 785-792, 2001
- [8] Li, M., Vitanyi, P. *An Introduction to Kolmogorov Complexity and Its Applications*, 2nd Edition. Springer-Verlag, 1997.
- [9] Musicant, D. R. NDC: Normally Distributed Clustered Datasets, *Computer Sciences Department, University of Wisconsin, Madison*, <http://www.cs.wisc.edu/dmi/svm/ndc/>, 1998.

- [10] Papadopolous H., Vovk V. and Gammerman A. Qualified predictions for large data sets in the case of pattern recognition. Proceedings of the International Conference on Machine Learning and Applications (ICMLA '02), 159-163, CSREA Press, 2002.
- [11] Proedrou, K., Nouretdinov, I., Vovk, V., Gammerman, A. Transductive confidence machine for pattern recognition. ECML 2002, Helsinki, Finland, August 19-23, 2002. Proceedings. Vol. 2430. Springer-Verlag, pp. 381-390, 2002
- [12] Raetsch G., Onoda T. and Mueller K.R. Soft margins for Adaboost, Machine Learning 42, 3, 287-320, 2001.
- [13] Saunders, C., Gammerman, A., Vovk, V. Transduction with confidence and credibility. In: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31-August 6, 1999. Vol. 2. pp. 722-726, 1999.
- [14] Saunders C., Gammerman A. and Vovk V. Computational efficient transductive machines, Algorithmic Learning Theory, 11th International Conference, Sydney, Australia, December 11-13, 2000, Lecture Notes in Computer Science, 1968, Springer, 325-333, 2000.
- [15] Syed N., Liu H. and Sung K.-K. Incremental learning with support vector machines, Proc Workshop on support vector machines at IJCAI-99, Stockholm, Sweden, 1999.
- [16] Vapnik, V. Statistical Learning Theory, Wiley-Series, 1998.
- [17] Vovk V., Gammerman A. and Saunders C. Machine-learning applications of algorithmic randomness. Proceedings of the Sixteenth Int. Conf. on Machine Learning (ICML 1999), 444-453, 1999.
- [18] Vovk V. On-line confidence machines are well-calibrated. Proc. 43th IEEE Symposium on Foundations of Computer Science, 187-196, 2002.
- [19] Vovk V, Gammerman A. and Shafer G. Algorithmic learning in a random world (Manuscript), July 2004.
- [20] Weerahandi, S. Exact statistical methods for data analysis. Springer-Verlag, 1994.
- [21] Xiao, R., Wang, J. and Zhang, F. An approach to incremental svm learning algorithm, 12th IEEE International Conference on Tools with Artificial Intelligence (IC-TAI'00), Vancouver, British Columbia, Canada, November 13 - 15, 2000.