Lag Correlation Discovery and Classification for Time Series and Data Streams

Georgios Dimitropoulos[1], Stella Papagianni[2] and Vasileios Megalooikonomou[2]
[1]*Department of Mathematics, University of Patras, Patras, Greece*
[2]*Department of Computer Science, University of Patras, Patras, Greece*
george.dimitropoulos1993@gmail.com, spapagianni@upatras.gr, vasilis@ceid.upatras.gr

Keywords: Data Mining, Data Streams, Time series, Classification, Lag Correlation, Incremental SVM Learning Algorithm, Stock Trend Prediction, Features Extraction.

Abstract: The high volume of data in dynamic places such as financial markets necessitates the implementation of effective mining methods to aid investors towards profitable decision making. The mining problems that are studied in this paper are lag correlation discovery and classification. Regarding the first, a new lag correlation algorithm for time series is proposed in this paper which is called Highly Sparse Lag Correlation (HSLC) and it is a combination of Boolean Lag Correlation (BLC) and Hierarchical Boolean Representation (HBR) algorithms. It aims to improve the time performance of Pearson Lag Correlation (PLC) algorithm. The classification algorithm that is employed is an incremental support vector machine (SVM) learning algorithm. To verify the effectiveness and efficiency of the proposed schemes, the lag correlation discovery algorithm is experimentally tested on electroencephalography (EEG) data that measures the electric activity of the brain, whereas the classification algorithm is tested on real financial databases. The HSLC algorithm that we propose, achieves better time performance over previous state-of-the-art methods such as the PLC algorithm and the incremental SVM learning algorithm that we adopt, increases the accuracy achieved by non-incremental models.

1 INTRODUCTION

The purpose of lag correlation discovery is to detect correlations with lag that are present in time series data. The most common way for detecting correlation is the Pearson correlation coefficient. In many applications, due to the fact that there is a great number of time series that have a high correlation, it is necessary for a method to accept only the pairs that have correlation greater than a high threshold (Pearson Lag Correlation (PLC) algorithm). The PLC algorithm can be used, not only for time series, but also for data streams. However, for online applications, the time performance of the algorithm is questionable. There is a rich body of literature on lag correlation algorithms for data streams. The Muscles algorithm proposed by Yi et al. (Yi, 2000) is able to adapt to changing correlations between data streams, handle a great number of long sequences efficiently and predict precisely the missing values. Another approach is the StatStream incremental algorithm proposed by Zhu et al. (Zhu, 2002) which is based on Discrete Fourier Transform (DFT) and a time interval hierarchy of three levels, to calculate efficiently high correlation among all pairs of streams. The Boolean Lag Correlation (BLC) algorithm proposed by Zhang et al. (Zhang, 2011) and the Hierarchical Boolean Representation (HBR) algorithm proposed by Zhang et al. (Zhang, 2009) calculate lag correlation among multiple data streams and are based on boolean representation. HBR is based on the fact that, in the majority of applications, only a small fraction of all possible pairs have a high correlation. Thus, in order to reduce the computational time, a technique that spots only the pairs which are correlated is used.

The purpose of classification algorithms on the other hand is to create a matching for a set of unknown features (test set) into some known categories (training set). There is a rich body of research work related to incremental SVM learning algorithms for classification. A framework for exact incremental learning and adaption of SVM classifiers has been studied by Poggio et al. (Poggio, 2001) and Diehl et al. (Diehl, 2003). Their approach

is general. The algorithm that is used is able to learn and unlearn manifold examples, adapt the current SVM to changes in regularization and kernel parameters and evaluate generalization performance.

On the application domain the macroeconomic environment is a field in which there has been extensive research regarding classification algorithms. The prediction of the financial indicators and the recognition of patterns in stocks are known as stock trend and it has been studied among others by Edwards et al. (Edwards, 2007). Kim (Kim, 2003) has used SVMs for the prediction of the trend of the daily stock price index of Korea (Korea composite stock price index-KOSPI).

In this paper, we focus on lag correlation and classification algorithms for time series and data streams respectively and we examine their applications on medical and financial data. The main contributions of this work can be summarized as follows:

- *Better time performance*: We propose and implement a new lag correlation algorithm for time series, the HSLC algorithm, which achieves a better computational time than the PLC algorithm.
- *Better accuracy*: We employ an incremental SVM learning algorithm which is proposed by Diehl et al. (Diehl, 2003) for the prediction of the KOSPI, in order to increase the accuracy that is accomplished by Kim (Kim, 2003) and conclude that the way of training models should be incremental.

In Section 2 the Highly Sparse Lag Correlation (HSLC) is presented, whereas in Section 3 the incremental SVM learning approach is analyzed. Both cases are experimentally studied in Section 4 followed by some remarks. Finally, in Section 5, the results of the experiments are discussed.

2 HIGHLY SPARSE LAG CORRELATION ALGORITHM

In the case of a large number of time series or data streams the set of all possible correlated pairs is rapidly increased. In order to be able to examine them for possible correlation it is vital to use algorithms which limit calculations to a smaller set and provide faster implementation through accurate approximations. The proposed approach, HSLC, is an algorithm that calculates lag correlation between time series. It can be used in applications where the number of highly correlated pairs is extremely smaller than the number of all possible pairs. The algorithm calculates lag correlation, using the following process:

Given k time series of length n,

1. Calculate the pairs that have lag correlation which is greater than a threshold.
2. Return these correlated pairs and the value of their lag l.

The HSLC algorithm is based on the HBR and takes into account the above observation, that the number of highly correlated pairs is extremely smaller than the number of all possible pairs. To cope with the related high computational cost, the algorithm uses a correlated pair detection technique based on the BLC algorithm including the Boolean Representation method. After that, the lag correlation is calculated for the set of pairs which were detected using the Pearson correlation coefficient. Before the presentation of the HSLC algorithm, we give some definitions:

Definition 2.1. (**Pearson correlation coefficient)** The Pearson correlation coefficient for two time series X and Y with the same length n for lag l with $0 \le l \le \frac{n}{2}$ is given by the formula:

$$r(lag) = \frac{\sum_{t=l+1}^{n} (X_t - \bar{X}) \cdot (Y_{t-l} - \bar{Y})}{\sqrt{\sum_{t=l+1}^{n} (X_t - \bar{X})^2} \cdot \sqrt{\sum_{t=1}^{n-l} (Y_t - \bar{Y})^2}} \quad (1)$$

where:

$$\bar{X} = \frac{1}{n-l} \cdot \sum_{t=l+1}^{n} X_t \quad (2)$$

$$\bar{Y} = \frac{1}{n-l} \cdot \sum_{t=1}^{n-l} Y_t \quad (3)$$

Definition 2.2. (**Boolean series)** The transformation of one time series X of length n into the corresponding Boolean series W of the same length n is given by the formula:

$$w_t = \begin{cases} 1, x_t > \bar{x} \\ 0, x_t \le \bar{x} \end{cases} \quad (4)$$

where:

$$\bar{x} = \frac{1}{n}\sum_{t=1}^{n} x_t \tag{5}$$

Definition 2.3. (**Correlation coefficient for Boolean series)** The calculation of the lag correlation coefficient for two Boolean series (W_i, W_j) is given by the formula:

$$\delta(l) = \max\left\{\delta_{pos}(l), \left|\delta_{neg}(l)\right|\right\} \; for\, l = 0,1,...,\frac{n}{2} \tag{6}$$

$$\delta_{pos}(l) = 1 - corr(l) \tag{7}$$

$$\delta_{neg}(l) = -corr(l) \tag{8}$$

$$corr(l) = \frac{\sum_{t=l+1}^{n} w_{t-l}^{i} XOR w_t^j}{n-l} \tag{9}$$

The proposed algorithm is described below:

HSLC algorithm

```
begin
Input: k time series X1,X2,...,Xk
detection_threshold, correlation_threshold, C.
//detection_threshold is the threshold that is used in order to find if the boolean series have a correlation, correlation_threshold is the threshold that is used in order to find if the time series, which were detected through the previous calculation of their corresponding boolean series, have a correlation and set C contains the pairs of time series (Xi,Xj) which have corresponding boolean series with correlation coefficient bigger than the detection_threshold //
Output: (Xi,Xj),max,lag
//(Xi,Xj) are the correlated pairs (if any), max is the maximum value of the Pearson correlation coefficient for every possible value of the lagl for the pairs (Xi,Xj) and lag is the value of l where the Pearson correlation coefficient takes its maximum value//
C ← ∅
for each time series Xi do
   Calculate X̄i; // X̄i is the mean value of Xi //
   Calculate Wi; // Wi is the transformed Boolean series of time series Xi which is calculated through the above formula (Definition III.2.)//
end for
for each pair of Boolean series Wi and Wj do
   max = 0; //maximum positive or negative correlation//
   for l = 0 to n/2 do
      //Calculate the Boolean correlation coefficient δ(l) by using the above formula (DefinitionIII.3)//
      If |δ(l)| > |max| then
         max = δ(l);
      end if
   end for
   if |max| = detection_threshold then
      //Add the pair (Xi,Xj) to the set C //
   end if
end for
for each pair of time series Xi and Xj that belong to the set C do
   max = 0; //maximum positive or negative correlation//
   for l = 0 to n/2 do
      //Calculate the Pearson correlation coefficient R(l) by using the above formula (Definition III.1)//
      If |R(l)| > |max| then
         max = R(l);
         lag = l;
      end if
   end for
   if |max| ≥ correlation_threshold then
      return (Xi,Xj),max,lag
   end if
end for
End.
```

For k time series of length n, the time complexity of the HSLC algorithm for the transformation of the initial time series into the Boolean ones is $O(kn)$. The corresponding time

complexity for the calculation of the correlation between the Boolean series is $O(1)$ as it only involves calculations between bits. So, the total time of detecting the correlated pairs is $O(kn)$. Finally, the time complexity for the calculation of the correlation between the pairs which belong to the set C using the Pearson correlation coefficient is $O(ck)$ where c is the number of the correlated pairs and it is very small. Thus, the aggregate time complexity of the algorithm is $O(kn)$ and it is a remarkable reduction in contrast to the time complexity of $O(k^2 n)$ that the naive PLC method achieves.

The above algorithm was tested on EEG data. For this EEG 22 different channels with a variety of sensors types and with a frequency of 256 Hz were used. An example of an input series follows:

Table 1: Features of input series.

Sensor id	*EEG*1
Frequency	256
Value	−1.8
Sensor Type	*Fp*2*AV*
Time Stamp	15 / 08 / 2016 17 : 30 : 55

3 SVM INCREMENTAL MODEL

3.1 Support Vector Machines and Incremental Learning

SVMs create patterns among the data of an initial set and subsequently, with the projection of formed data vectors (which are values that have been extracted through certain feature extraction) in space, allow to conclude over the data values that do not belong on the initial set. In this paper the Gaussian function is used (due to its speed), expressed in the form of

$$k(x,z) = e^{-\frac{\|x-z\|^2}{2\cdot scale^2}} \quad (10)$$

where $x \in R^{(m\times 12)}$, are the support vectors (margin, error and reserve), $z \in R^{(1\times 12)}$ is the input vector, *scale* is the support vectors' size and m is the support vectors' number.

Traditional data mining algorithms calculate their results for a pre-defined set of data. However, in the vast majority of applications, the set of data is not known from the beginning. Thus, it is vital for the algorithms to be adapted to the new upcoming data. This means that algorithms use the knowledge that had developed from their previous implementations and so are able to increase the precision in their results. This process is referred as incremental learning. There are a lot of algorithms which approach the above problem through SVMs employed by Poggio et al. (Poggio, 2001) and Syed et al. (Syed, 1999). In this work, we adopt the case that is studied Poggio et al. (Poggio, 2001).

3.2 Stock Prediction and Feature Extraction

The proposed model for stock trend prediction involves as a second step the extraction of features. These features should be able to be separated by the classification algorithm into two classes, the upwards and the downwards trend of the index respectively. These features are technical indicators, such as *William*&*R*, A/ D *Oscillator* , which are used for the creation of the SVM model that predicts the direction of the KOSPI and they have also been used by a variety of similar papers (e.g., Shin, 2005). In this paper the twelve indicators case was adopted and implemented following the approach described by Kim (Kim, 2003).

3.3 SVM Incremental Model

The streams in the aforementioned implemented model are the financial indicators previously mentioned. The constructed model is able not only to be adapted to the upcoming data but also maintain its existing knowledge through the adaption of the hyperplane of separation. More specifically, the incremental SVM learning algorithm (no instance memory) is implemented. Subsequently, our model classifies the test set into two classes, the upward and the downward trend of the indicators. Concurrently, this test set becomes part of the current training set, thus an increase of model's knowledge is achieved. Finally, the model is used to predict the trend (upward or downward) that the stock is going to have in the next days.

This model is imported into a system with a five-step process:

1. INPUT. The system receives the data streams which are the stocks with their values per day.
2. FEATURE EXTRACTION. The system receives the above sample streams and calculates the twelve indicators through a function which has been implemented in Matlab. We also calculate another feature, named trend feature. This feature has the value ($+1$, *if* $Close(t) < Close(t+1)$, upward trend) for time t, otherwise it has the value (-1, downward trend).
3. MODEL INCREMENT. The model receives the above indicators through overlapping windows of fixed size and is adapted to these in order to be able to increase its knowledge. We used the Microsoft StreamInsight Platform which is available at Microsoft StreamInsight (Microsoft StreamInsight , 2016), to handle the streams, in order to create the model of incremental learning and we run the incremental SVM learning algorithm which is proposed by Diehl et al. (Diehl, 2003) through the library Diehl (Diehl, 2011) which is implemented in Matlab. Every time that there is an increase in the knowledge through the algorithm, a new model is derived as an output. Thus, for time t, we have the model $T(i)$, where i is the set of samples which has been already processed by the system, and for a new input window, we have the model $T(i+w)$, where w is the predefined length of the input. The number of margin vectors (which are constantly increased as the system increases its knowledge), the number of reserve vectors (which are increased in a slower rate than the margin vectors) and the number of error vectors (which are only a few, during the whole implementation) are obtained as an output.
4. PREDICTION. After the adaption, the model is able to predict the trend of the stock for the next days as well as finding the class that they belong to. The results are compared with three other versions of the SVM algorithm, in order to estimate the accuracy of the predictions.
5. OUTPUT. Finally, the output receives the above results and presents the prediction of the trend (upward or downward).

An example of an input feature is the following, as retrieved from the Google/Finance (Google/Finance, 2016) database.

Table 2: Features of input streams.

Date	$12-Dec-14$
Open	1921.6
High	1926.6
Low	1915
Close	1921.7
Volume	36347000

This file had 2965 rows, where each row represents a different day. An input adapter in Microsoft StreamInsight was generated, which reads the input file (.csv format) and matches each row to a sample stream, whereas each column is an input feature with the field *Date* to be the key for the creation of the streams.

We have used the Microsoft StreamInsight platform for the creation of the streams and Matlab for their handling and for the implementation of our functions.

4 EXPERIMENTS

The experimental procedure includes two parts. In the first part, the problem of lag correlation discovery between time series is considered, applying the HSLC algorithm on EEG data. In the second part, experimentation with the problem of classification is conducted, by implementing the SVM incremental learning algorithm and testing it on KOSPI data streams.

4.1 Highly Sparse Lag Correlation Algorithm

The algorithms HSLC, PLC and BLC were implemented in Matlab and compared. The experiments are separated into two parts. In the first part we split the time series in windows and in the second part the lag correlation between them was calculated in their whole length. The precision of an algorithm Y in comparison to the theoretical correct results of an algorithm X is given by the formula:

$$precision(i) = \begin{cases} \dfrac{lag^i X}{lag^i Y}, lag^i X \le lag^i Y, \text{for the TP pairs} \\ \dfrac{lag^i Y}{lag^i X}, lag^i X > lag^i Y, \text{for the TP pairs} \\ 0, \text{for the } TP \text{ or } FN \text{ pairs} \end{cases} \quad (11)$$

where:

- TP (True Positives): The correlated pairs that were detected correctly by the algorithm Y in comparison to these that the algorithm X had detected.
- FP (False Positives): The pairs that were detected as correlated by the algorithm Y but not by X.
- FN (False Negatives): The pairs that were correlated according to X, but they were not detected by Y.
- lag^i: The output of the algorithms for the pair i.
- $i = 1, 2, ..., N$, where N is the number of all pairs that we examined, except from the TN (True Negatives).

The final value of the variable precision is the mean of the vector $precision(i)$.

4.1.1 Lag Correlation with the Use of Windows

The input time series, where each of them has 2000 values, are split in windows. In particular, we use a 128 -element sliding window in each channel. Thus, the lag correlation between the subsets of the time series is calculated for every slide of the window. The results of the experiments follow below:

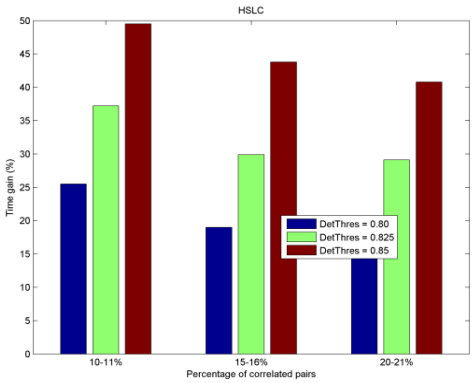

Figure 1: HSLC algorithm. Time gain in detection threshold and in percentage of the correlated pairs of the time series (Number of time series = 8).

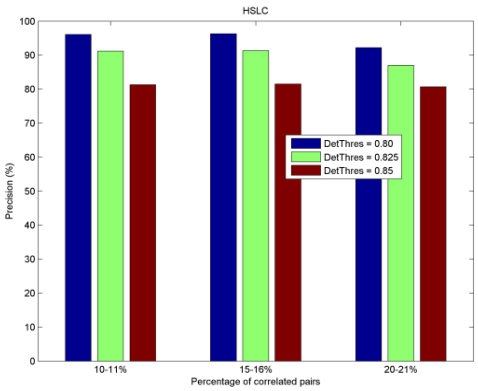

Figure 2: HSLC algorithm. Precision in detection threshold and in percentage of the correlated pairs of the time series. (Number of time series = 8).

By examining Figures 1 and 2 we reach the following conclusions:

- The time gain ranges from $18.3\% - 49.5\%$ and the precision ranges from $83.2\% - 95.3\%$
- It is noticed that when the detection threshold is increased, the time gain is also increased but the precision is decreased. This happens due to the fact that, when there is a higher detection threshold the algorithm detects fewer correlated pairs and needs fewer time for the calculation of the correlation coefficients. Thus, there is a trade-off between time gain and precision.
- When the percentage of the correlated pairs is increased, the time gain is decreased and the precision remains the same. This happens because when we have a higher percentage of correlated pairs the detection technique has a fewer time gain.

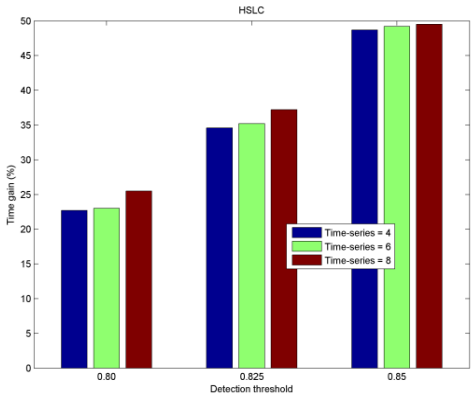

Figure 3: HSLC algorithm. Time gain in number of the time series and in detection threshold. (Percentage of correlated pairs = 10 − 11%).

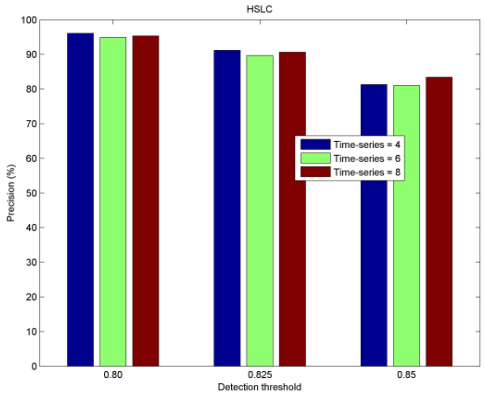

Figure 4: HSLC algorithm. Precision in number of the time series and in detection threshold. (Percentage of correlated pairs = 10 − 11%).

From Figures 3 and 4 we reach the following conclusion:

- When the number of time series is increased, we notice a slight increase in time gain while the precision remains nearly the same. This happens due to the fact that for a greater number of time series there are more possible correlated pairs and the detection technique leads to a greater time gain.

4.1.2 Lag Correlation without Windows

Here we calculate the lag correlation between time series in their whole length. We also implemented the BLC algorithm which is the detection part of the HSLC as it has a very high time gain in the case that there are no windows. The results of the experiments are the following:

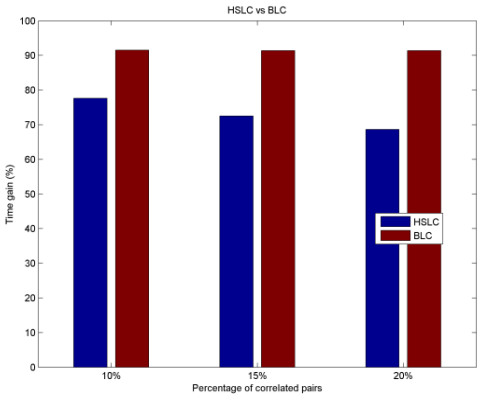

Figure 5: HSLC & BLC algorithms. Time gain in percentage of correlated pairs. (Number of time series=25).

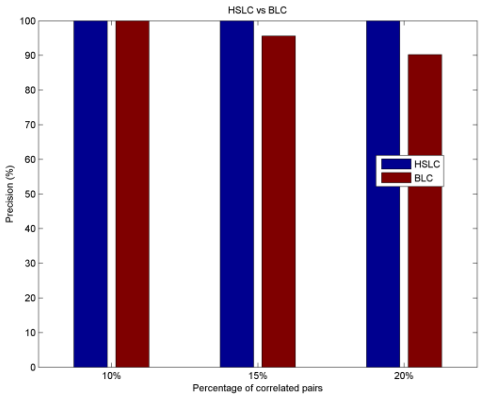

Figure 6: HSLC & BLC algorithms. Precision in percentage of correlated pairs. (Number of time series=25).

From Figures 5 and 6 we reach the following conclusions:

- The time gain for the HSLC and BLC is in the range of 68.6% − 77.6% and 91.4% − 91.5% respectively. The precision for the HSLC is 100% for all cases and for BLC it ranges from 90.2% − 100% . Subsequently, the HSLC has a smaller time gain than BLC, but is has an infinitesimal error in precision in contrast to BLC.
- In HSLC, when the percentage of correlated pairs is increased, the time gain in decreased and the precision remains the same. In contrast, in BLC, when the percentage of correlated pairs is increased, the time gain remains nearly the same and the precision is decreased. Thus, the algorithms have an opposite behavior in time gain and in precision.

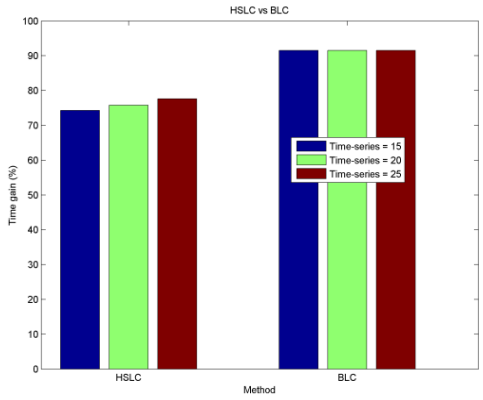

Figure 7: HSLC & BLC algorithms. Time gain in number of time series. (Percentage of correlated pairs $=10\%$).

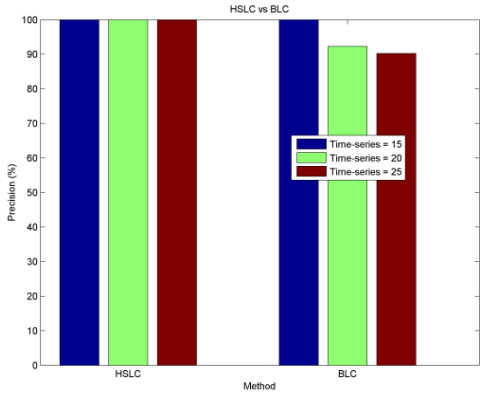

Figure 8: HSLC & BLC algorithms. Precision in number of time series. (Percentage of correlated pairs $=10\%$).

From Figures 7 and 8 we conclude that:

- In HSLC, when the number of time series is increased, we notice a slight increase in time gain while the precision remains the same. In contrast, in BLC, when the number of time series is increased, the time gain remains nearly the same and the precision is decreased.

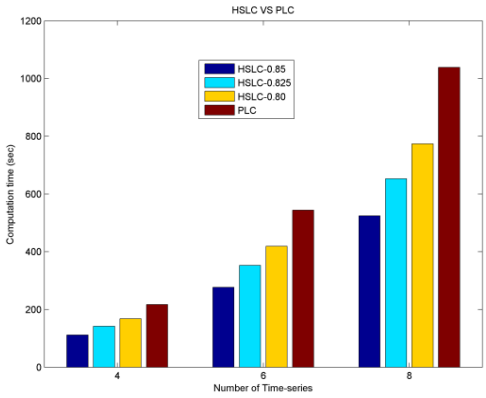

Figure 9: With windows. Comparison of the computational time of HSLC & PLC algorithms in number of time series. (Percentage of correlated pairs $=10-11\%$).

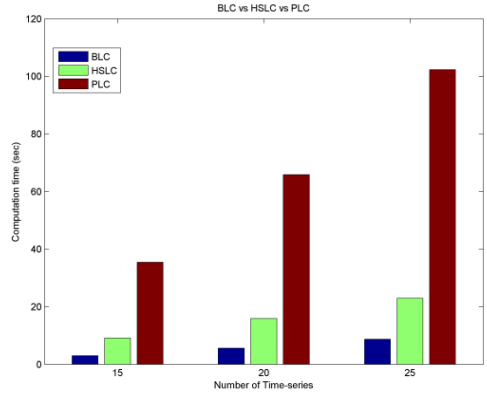

Figure 10: Without windows. Comparison of the computational time of BLC, HSLC & PLC algorithms in number of time series. (Percentage of correlated pairs $=10\%$).

From Figures 9 and 10 we reach the following conclusions regarding the general performance of the algorithms:

- In case that we use windows, the HSLC has a considerable reduction in computational time in comparison to PLC. The size of this reduction depends on the value that the user provides for the detection threshold by taking into account the tradeoff between time and precision.
- In case that we do no use windows, the HSLC has a dramatic reduction in computational time in comparison to PLC regardless of the value of the detection threshold. In this experiment we use the value 0.825. Thus, in this case we do not have to take into account a tradeoff, as

we achieve the maximum precision (100%) and a very high time gain with the above value of the detection threshold.

- In the implementation without windows, the BLC algorithm could be also used as it achieves a smaller computational time than the HSLC. However, it does not ensure an infinitesimal error as we noticed above.

4.2 Classification Experiments and SVM Incremental Model

In the implementation of our classification experiments, we compare our model with three other versions of the SVM algorithm. In our approach, we train the incremental SVM learning algorithm with the first 3000 examples and then the algorithm increases its knowledge with feedback from the next 3000 examples. Thus, the test set becomes part of the training set and our model is able to increase its knowledge. We use a window that has a size of 10 examples. The second approach is the classic SVM which is trained with the same first set of 3000 examples and afterwards tested with the next 3000 examples. The third version is the algorithm which we call static incremental SVM and also uses the same training and test sets of 3000 samples each. The difference in the latter case is the use of the SVM incremental learning algorithm instead of the classic SVM. However, it is only trained one time, in contrast to our approach. The fourth and last version is the online SVM algorithm which is trained with the same first 3000 examples and after that increases the initial training set with the following 3000 samples. The difference between this and our method is that this model is re-trained every time a new instance arrives (full instance memory). The following table shows the average accuracy every method achieved in predicting the price of stock for 3000 examples.

Table 3: Comparison of the accuracy of the SVMs algorithms.

ALGORITHM	ACCURACY
Classic SVM	52%
Static Incremental SVM	60%
Online SVM	62%
Incremental SVM Learning	74%

It can be observed that the incremental SVM learning algorithm achieves the highest accuracy. Finally, it is worth mentioning that our incremental model increases its accuracy with the increase of the training set. It achieves an accuracy of 71% for the first half of the training set and an accuracy of 77% for the second one.

5 CONCLUSION

In this work, we proposed two different data miming techniques for time series and data streams. The first is associated with the problem of lag correlation discovery of time series. The HSLC algorithm was proposed achieving remarkable reduction in the time complexity in contrast to the state-of-the-art method PLC, by concurrently preserving high accuracy in the results. Furthermore, it has an infinitesimal error for both the value of the lag and the value of the correlation coefficient for every detected correlated pair of series. In the second part of this paper, we examined the problem of classification of data streams and evaluated several approaches on a stock prediction case. Specifically, a model which is used for data mining on streams was presented by employing an incremental SVM learning algorithm on the KOSPI. The algorithm had better results on the accuracy in contrast to three other versions of the SVM algorithm and we concluded that the training models for the stock prediction problem should follow incremental iteration methodology. The experimental results also demonstrate that through efficient implementation of various mining techniques such as lag correlation discovery or classification on time series or on data streams, useful information can be acquired. This information comes through proper interpretation of patterns discovered by mining algorithms.

REFERENCES

Diehl, C. P. and Cauwenberghs, C. (2003) "SVM Incremental Learning, Adaption and Optimization", *Neural Networks, 2003, In Proc. of the International Joint Conference on*, Vol.4, pp 2685-2690, IEEE.

Edwards, R.D., Magee, J. and Bassetti, W.H.C. (2007) *Technical Analysis of Stock Trends*, 9th ed.

Kim, Kyoung-jae (2003) "Financial time series forecasting using support vector machines", *Neurocomputing Journal*, Vol. 55(1-2), pp. 307-319.

Poggio, T. and Cauwenberghs, C. (2001) "Incremental and Decremental Support Vector Machine Learning",

Advances in Neural Information Processing Systems 13: In Proc. of the 2000 Conference, Vol. 13, MIT Press.

Shin, K.S., Lee, T.S. and Kim, H. (2005) "An application of support vector machines in bankruptcy model", *Expert Systems with Applications Journal*, Vol. 28(1), pp. 127-135.

Syed, N. A., Liu, H. and Sung, K. (1999) "Incremental Learning with Support Vector Machines", *International Joint Conference on Artificial Intelligence*.

Yi, B.-K. , Sidiropoulos, N.D., Johnson, T., Jagadish, H.V., Faloutsos, C. and Biliris, A. (2000) "Online Data Mining for Co-Evolving Time Sequences", *In Proc. of the 16th International Conference on Data Engineering*, pp. 13-22.

Zhang, T., Yue, D., Gu, Y., Wang, Y. and Yu, G. (2009) "Adaptive correlation analysis in stream time series with sliding windows", *Computers & Mathematics with Applications Journal*, Vol. 57(6), pp.937-948.

Zhang, T., Yue, D., Wang, Y. and Yu, G. (2011) "A Novel Approach for Mining Multiple Data Streams based on Lag Correlation", *In Proc. of the Control and Decision Conference*, pp. 2377-2382.

Zhu, Y. and Shasha, D. (2002) "StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time", *In Proc. of the 28th International Conference on Very Large Data Bases*, pp. 358-369.

(2014). Google/Finance. In *Available* at *http://www.google.com/finance.*

(2016). Microsoft StreamInsight. In *Available* at *https://technet.microsoft.com/en-us/library/ee362541%28v=sql.111%29.aspx*.

Diehl, C. (2011). Github, inc. In *Available* at *https://github.com/diehl/Incremental-SVM-Learning-in-MATLAB.*