Benchmarking a Coevolutionary Streaming Classifier under the Individual Household Electric Power Consumption Dataset

Alexander Loginov[1], Malcolm I. Heywood[1], and Garnett Wilson[1]

[1]*Faculty of Computer Science, Dalhousie University, Halifax, NS. Canada*

Article originally appears in IJCNN'16 under IEEE copyright
Please cite http://ieeexplore.ieee.org/document/7727557/

Abstract

The application of genetic programming (GP) to streaming data analysis appears, on the face of it, to be a less than obvious choice. If nothing else, the (perceived) computational cost of model building under GP would preclude its application to tasks with non-stationary properties. Conversely, there is a rich history of applying GP to various tasks associated with trading agent design for currency and stock markets. In this work, we investigate the utility of a coevolutionary framework originally proposed for trading agent design to the related streaming data task of predicting individual household electric power consumption. In addition, we address several benchmarking issues, such as effective preprocessing of stream data using a candlestick representation originally developed for financial market analysis, and quantification of performance using a novel 'area under the curve' style metric for streaming data. The computational cost of evolving GP solutions is demonstrated to be suitable for real-time operation under this task and shown to provide classification performance competitive with current established methods for streaming data classification. Finally, we note that the *individual household electric power consumption* dataset is more flexible than the more widely used *electricity utility prediction* dataset, because it supports benchmarking at multiple temporal time scales.

1 Introduction

Investopedia[1] gives the following definition for a 'Technical Indicator' or TI: "Any class of metrics whose value is derived from generic price activity in a stock or asset. Technical indicators look to predict the future price levels, or simply the general price direction, of a security by looking at past patterns." As such, a significant body of research has appeared in which evolutionary methods have been deployed for designing TI for use with a Decision Tree (DT) for the purposes of specifying the actions of a trading agent under both foreign exchange (FX) and stock markets. The actions typically represent one of three decisions: buy, hold or sell. Moreover, the related task of financial forecasting employs the TI—DT agent to make predictions regarding the state of the market at the next time step, i.e. will the market move up, down or experience no change. The TI can be evolved independently of the DT (i.e., different TI might act as better indicators for risk versus return [6], [11]) or, in a more recent development, coevolved collectively in a single process [15], [12], [14], [18], [17], [19]. The coevolutionary approach implies that performance is only ever expressed in terms of the ultimate performance objective (wealth creation) as opposed to introducing surrogate performance metrics that attempt to characterize what a good TI should represent.

Despite the wide range of research conducted in coevolving TI—DT agents for financial trading and forecasting tasks, there has been little utilization under the related task of streaming data classification [10]. In this work, we conduct an initial study to assess the potential for deploying frameworks for designing TI—DT agents under the task of predicting movement in power utility values, i.e. predicting whether the consumption of a utility (e.g., water, gas, electricity) will decrease or increase at the next time step relative to the recent past (a task that is also synonymous with that required for sentiment analysis in the analysis of document texts).

On the face of it, evolving TI—DT agents for financial decision making represents a particularly promising starting point. Both stock and FX markets represent environments in which the underlying process is potentially non-stationary, implying that it is particularly important to address the issue of when to rebuild agents. In the context of streaming data classification, making decisions under non-stationary tasks is frequently equated with

[1] http://www.investopedia.com/terms/t/technicalindicator.asp

'shift' and 'drift' [5, 10]. Models may either take the form of ensembles of multiple decision makers and / or incrementally react to each and every sample from the stream. A body of research has also been developed for characterizing stream content statistically and relating this to change detection (e.g. Hoeffding Trees) [1, 7]. Moreover, because there is only ever one decision maker constructed, the issue of which model to deploy for the purposes of 'anytime classification' is potentially more straight forward to answer than under evolutionary methods.

For the purposes of this initial study we concentrate on the following points: 1) identifying what needs to be addressed to facilitate the application of a previously proposed framework for coevolving TI—DT agents under financial trading environments to the power consumption task; 2) introduce the multi-temporal resolution electricity utilization data set assumed for benchmarking purposes; and, 3) propose an 'area under the curve' style metric capable of characterizing performance over the entire stream using a single scalar.

With this in mind, Section 2 summarizes FXGP a previously proposed approach for coevolving TI—DT decision makers. In doing so, we emphasize how FXGP addresses the issues of change detection, anytime operation as well as coevolving TI and DT. Section 3 introduces the data set and preprocessing performed to support benchmarking at multiple temporal resolutions. Section 5 presents the results of the benchmarking study, with conclusions made in Section 6.

2 FXGP Algorithm Overview

The FXGP algorithm is used to predict the changes in the direction of movement of a measured parameter, i.e. the analysis of currency pairs or stock prices. Specifically, TI are used to discover temporal features, whereas each DT expresses a set of rules (defining a trading strategy). After executing the TI—DT programs a decision is made to use one of a discrete number of actions. Under the streaming classification task the coevolution of TI—DT individuals will be used to predict the movement in the next time step of the sequence (e.g., up, down, no change), or a process synonymous with financial forecasting [15], [12], [14]. In this section we provide an overview of the main parts of the FXGP algorithm [17–19].

The FXGP algorithm includes three major steps, each responsible for one of the following tasks (Figure 1):

1. **Training Step:** Coevolution of decision tree (DT) population and technical indicator (TI) population over a training data subset N_t.
2. **Validation Step:** Identification the best TI–DT combination (hereafter 'TI–DT champion') over a validation data subset N_v.
3. **Labelling Step:** Deployment of a champion agent during which the labels are suggested by the champion TI—DT pairing. Retraining is only triggered when some error threshold is exceeded (retrain signal).[2]

The data partitions associated with Train, Validate and Label steps do not overlap. Moreover, the formulation of the Validation and Label steps for the streaming data classification task have been simplified from those used for trading tasks. The DT and TI population exist in a symbiotic cooperative coevolutionary relationship (Figure 2). Thus, each DT utilizes some subset of the available TI individuals and performance (fitness) is only expressed at the level of the DT.

[2]Under the context of trading agent design this might take the form of a combination of multiple parameters (maximum drawdown, number of consecutive losses, etc).

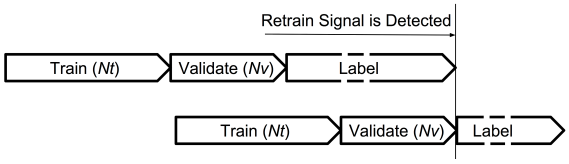

Figure 1: FXGP Train—Validate—Label cycle. The DT and TI populations are first symbolically co-evolved over a Training data subset. A single DT with linked TI is identified over a Validation data subset. The resulting champion agent is then deployed until the retrain criterion is met. Train and validation partitions can dynamically grow based on the number of exemplars already encountered. The Train—Validate—Label cycle is then repeated until all data is labeled.

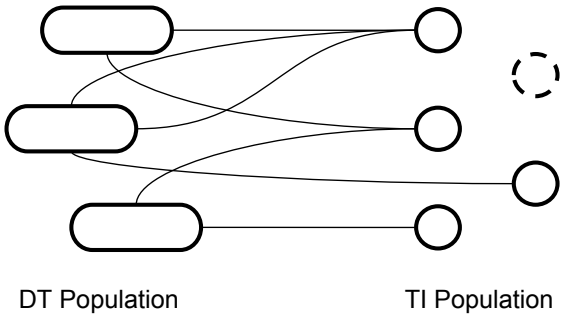

Figure 2: Symbiotic co-evolutionary relation between TI–DT Populations. Subsets of TI individuals are explicitly called by DT individuals. Any TI without at least one DT reference are deleted (see TI with dash).

Table 1: TI Header.

Field	Description
TI_{links}	How many times TI is linked to DT population
TI_{type}	Value or MA
TI_{period}	MA duration
TI_{shift}	Number of time steps back in a history

2.1 Training

Each training cycle begins with the complete reinitialization of DT and TI population content, i.e. all previous DT and TI content is discarded and new individuals randomly generated [17]. The DT population has a fixed size, whereas the size of a TI population can vary between generations. The TI population is initialized before the DT population. Each TI includes a header that stores individual properties of a TI (Table 1) and a TI program. The TI program assumes a register level transfer language with two registers, or 'linear GP' [4] (Table 2). The $R[0]$ register is an output register and contains a TI value after program execution.

The moving average of a TI (MA) at moment t is calculated as Eq. (1),

$$MA_t = \frac{\sum_{j=n}^{t} V_j}{TI_{period}} \tag{1}$$

where V_t is a TI value and $n = t - TI_{period}$

The TI header fields (Table 1) are initialized as follows:

- TI_{links} counter is set to 0.
- TI_{type} is randomly initialized with 0 (Value) or 1 (MA).
- TI_{period} is randomly initialized with an integer that satisfies the following condition: $1 < TI_{period} \leqslant period$ or ignored if TI_{type} is Value.
- TI_{shift} is randomly initialized with an integer that satisfies the following condition: $0 \leqslant TI_{shift} \leqslant shift$.

Table 2: TI functions. $R[x]$ denotes the content of the register x and $R[y]$ denotes the content of either: 1) the register y, 2) an attribute value, or 3) a an attribute c3...c6 value TI_{shift} time steps back in a history (where $x \neq y$ and x and $y \in \{0, 1\}$).

Function	Definition
Addition	$R[x] \leftarrow R[x] + R[y]$
Subtraction	$R[x] \leftarrow R[x] - R[y]$
Division	$R[x] \leftarrow R[x] \div 2$

Table 3: DT Header. Data structure for recording DT specific properties.

Field	Description
DT_{size}	DT size, nodes
DT_{score}	DT score (partition specific detection rate)

Table 4: Complete FXGP parameter list for DT and IT populations.

Parameter	Value	Description
hitMax	10	Max number of errors to re-training
tvMax	20000	Max total size of train and validation partitions, candlesticks
tvMin	336 or 672	Min total size of train and validation partitions, candlesticks
tvRatio	0.33	Validation to train partitions ratio
T_{max}	10000	Max number of generations
plateau	200	Plateau size, generations
shift	4	Max time shift, candlesticks
gap	0.25	Populations gap
trees	100	DT population size
nodes	5	Max DT size, nodes
drLim	0.6	Score limit
indicators	100	Initial TI population size
length	8	Max length of a TI program, steps
period	72	Max MA period

Initialization of the DT population is conducted after that of the TI population. Each DT can contain a different number of rules, DT_{size}, chosen stochastically over the interval: $1 < DT_{size} \leqslant nodes$. Each DT node is represented as one of the following conditional statements [19]:

- $IF\ (X_t > Y_t)\ THEN\ \langle arg_1 \rangle\ ELSE\ \langle arg_2 \rangle$
- $IF\ ((X_t > Y_t)\ and\ (X_{t-n} < Y_{t-n}))\ THEN\ \langle arg_1 \rangle$
  $ELSE\ \langle\ arg_2 \rangle$

where n is a shift back in a history relative to the current location, t; $X_t \neq\ Y_t$ and X_t and Y_t are randomly selected and can be an attribute value or a TI. Every time that a TI is linked to a new DT the corresponding TI_{links} counter is incremented. Likewise, arg_1 and arg_2 are randomly assigned as either a pointer to a different node, or one of the class labels appropriate to the streaming classification task. Table 3 summarizes the two DT headers characterizing properties specific to each DT.

Every generation, the *gap* DT individuals with the lowest DT_{score} are deleted and replaced. Any TI associated with a deleted DT have their corresponding TI_{links} counter decremented. Any TI with TI_{links} equal to 0 is considered useless and deleted, decreasing the TI population size (Fig. 2). Following the deletion of worst DT (and implicated TI), mutation is used to add $gap \times\ (TI_{population\ size})$ TIs and then add $gap \times\ DT_{size}$ DTs. Evolution stops when the maximum number of generations, T_{max}, is reached or when the best DT_{score} was not improved within *plateau* generations.

2.2 Validation

The validation process is used to identify the best TI–DT agent. Validation is performed over the partition of data that follows the training partition (Figure 1). If the DT_{score} of a champion agent after validation is higher than *drLimit*, then the selected agent is deployed for data labelling. If the best DT_{score} is less than *drLimit*, both DT and TI population are discarded and training is repeated. This process is simpler than employed under a trading agent context [17–19].

Table 5: Dataset 0 – Original individual household electric power consumption. Attribute information.

Attribute	Description (one-minute measurements)
a1	date
a2	time
a3	global active power: household global minute-averaged active power (in kilowatt)
a4	global reactive power: household global minute-averaged reactive power (in kilowatt)
a5	voltage: minute-averaged voltage (in volt)
a6	global intensity: household global minute-averaged current intensity (in ampere)
a7	sub metering 1: energy sub-metering No. 1 (in watt-hour of active energy). It corresponds to the kitchen, containing mainly a dishwasher, an oven and a microwave (hot plates are not electric but gas powered)
a8	sub metering 2: energy sub-metering No. 2 (in watt-hour of active energy). It corresponds to the laundry room, containing a washing-machine, a tumble-drier, a refrigerator and a light
a9	sub metering 3: energy sub-metering No. 3 (in watt-hour of active energy). It corresponds to an electric water-heater and an air-conditioner

Table 6: 'Dataset 1' dataset. Attribute information (30 minutes grouping).

Attribute	Description
b1	date (first 'a1' value within 30 minutes interval)
b2	time 't' (first 'a2' value within 30 minutes interval)
b3	sum of a3 within 30 minutes interval
b4	sum of a4 within 30 minutes interval
b5	sum of a5 within 30 minutes interval
b6	sum of a6 within 30 minutes interval
b7	sum of a7 within 30 minutes interval
b8	sum of a8 within 30 minutes interval
b9	sum of a9 within 30 minutes interval
label	'1' $IF\ b3(t+1) \geq b3(t)$ '-1' otherwise

2.3 Labelling

Prediction is performed using the TI–DT champion agent identified during Validation. Retraining is triggered when the number of wrongly classified instances reaches *hitMax*. When the *hitMax* limit is reached, the labelling stops and the whole Train—Validate cycle is repeated. The exemplars encountered during labelling defines the total size of the train and validate partitions unless it is less than *tvMax* (Table 4). Such a process is again distinct from that assumed under a trading agent context [18, 19].

Indeed, the task of change detection represents an ongoing topic of interest to streaming data classification [5, 10]. In this work, change detection is synonymous with the concept of an error threshold. Only on detecting change is a new policy constructed. This is distinct from continuous frameworks for model building [17], where the weights of a neural network might be adapted on a continuous basis.

We also note that in order to operate under real-time conditions and therefore have a champion available on an 'anytime' basis, it is necessary for the Train–Validate cycle to complete before the next exemplar in the sequence appears. If a new TI–DT champion is not available within this period then the champion previously available is required to continue to produce labels.

3 Data

In this work we apply the FXGP algorithm to the prediction of movement at the next time step for individual household electric power consumption. The original dataset consists of consumption measured at one-minute intervals over a period of 47 months (December 2006 — November 2010) [16].[3] This dataset has a much higher resolution with respect to the units of time than the frequently employed 'Australian New South Wales Electricity Market' data [9]. The high level of sampling available, implies that multiple datasets can be constructed from the same source data by assuming different periodicities for constructing source statistics.

Table 5 summarizes the attributes of the original dataset, hereafter 'Dataset 0'. This data is then preprocessed and converted into five separate datasets: 'Dataset 1' through 'Dataset 5'. All the resulting datasets assume an ARFF format that is accepted by frameworks for data stream mining such as MOA [3]. Specifically, the goal is to predict the movement in the 'global active power' attribute (a3, Table 5).

Dataset 1 was obtained by summing the one-minute measurements within consecutive 30 minute intervals and labelling as shown in Table 6. Dataset 1 will represent the base case dataset.

Dataset 2 represents a characterization of the original 'a3' attribute using the aggregation of 'open-high-low-close' information over the 30 minute period (Table 7). This corresponds to the widely employed 'candlestick' representation for price data in financial or stock data, i.e. a preprocessing step that potentially reduces the amount of noise in the original measurements. Note that the criteria for the label are still relative to the definition for attribute 'b3' in Dataset 1. Dataset 3 assumes the same preprocessing of the attributes into 30 minute candlestick's, but casts the labelling task into one of three categories (less, approximately the same, or more) as opposed to one of two (Table 8). In this work, 'approximately the same' is taken to imply within $\pm 10\%$ of the previous value,

[3] http://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption

Table 7: Dataset 2 – Attribute information (30 minute candlesticks, 2 class classification). Note that attributes c3 through c6 represent those potentially indexed by a TI

Attribute	Description
c1	date (first 'a1' value within 30 minutes interval)
c2	time 't' (first 'a2' value within 30 minutes interval)
c3	first 'a3' value within 30 minutes interval
c4	highest 'a3' value within 30 minutes interval
c5	lowest 'a3' value within 30 minutes interval
c6	last 'a3' value within 30 minutes interval
label	'1' $IF\ b3(t+1)\ \geq\ b3(t)$ '-1' otherwise

Table 8: Dataset 3 – Attribute information (30 minutes candlesticks, 3 class classification). Note that attributes c3 through c6 represent those potentially indexed by a TI

Attribute	Description
c1	date (first 'a1' value within 30 minutes interval)
c2	time 't' (first 'a2' value within 30 minutes interval)
c3	first 'a3' value within 30 minutes interval
c4	highest 'a3' value within 30 minutes interval
c5	lowest 'a3' value within 30 minutes interval
c6	last 'a3' value within 30 minutes interval
label	'1' $IF\ b3(t+1)\ >\ 1.1 * b3(t)$ '-1' $IF\ b3(t+1)\ <\ 0.9 * b3(t)$ 'o' otherwise

resulting in a approx. equal total representation of each class (Table 9). Naturally, enforcing a tighter constraint would result in the 'equal to' class becoming less frequent.

Dataset 4 and Dataset 5 were processed in the same way as Dataset 2 and Dataset 3 respectively, but with candlesticks estimated over consecutive 15 minutes intervals. Table 9 provides a summary of the static properties of each dataset. The datasets are available publicly for research purposes.[4]

4 Streaming data AUC style metric

Metrics for performance evaluation of streaming data are in itself the subject of active research. That is to say, not only is the model under continuous development throughout the stream, but the underlying process creating the data is potentially non-stationary. As a consequence, it is desirable that the performance metric should be able to characterize performance over the *course* of the stream. Specifically, we note three distinct types of metric:

- **Prequential error metrics:** characterize 'accuracy' and explicitly include a decay/ forgetting factor relative to older instances [8]. We note that such metrics unfortunately become less meaningful as the degree of class representation becomes imbalanced [20].
- **Measures of (label) autocorrelation:** are designed to reflect the sensitivity of a streaming classifier to label autocorrelation [2]. Specifically, even if a stream is balanced (in the representation of multiple classes) as a whole, this does not necessarily mean that it is well mixed. In particular, if classes appear locally in batches with the same class label, then sophisticated models of classification might perform significantly worse than very simple single-bit predictors. The 'No class' classifier results reported later reflect such a metric (discussed further in Section 5.1).
- **Rate based metrics:** provide for the incremental estimate of the confusion matrix throughout the stream [10, 20]. As such, any scalar performance metric derived from the confusion matrix can be plotted as a function of progress through the stream and consequently characterize incremental development of a classifier over the course of the stream.

A Rate based metric is assumed in this work, in particular the multi-class Detection Rate (DR) formulation [10]. Previous experience with this metric demonstrated better discriminatory properties than prequential error metrics over a cross-section of streaming classification datasets [20]. In the following, the multi-class DR is summarized and a corresponding formulation for quantifying overall stream performance in terms of the 'area under the curve' (AUC) is described. The motivation for the latter is that all the above classes of metric express performance as a 'trajectory' (of the metric) against time. In order to explicitly quantify the difference between trajectories associated with different classification algorithms, the quality of any given trajectory needs to be re-expressed as a single scalar value before application of statistical tests. To date, general practice has been to assume the last numerical value from the (streaming data) performance metric as the scalar value [1, 7]. This approach emphasizes performance towards the end of the stream as opposed to recognizing the quality of results *throughout* the stream. The purpose of the AUC metric proposed here is to more accurately capture (as a single scalar value) the performance of a classifier over the duration of the stream as a whole.

[4] `http://web.cs.dal.ca/~mheywood/Data/`

Table 9: Datasets summary.

Dataset	Attributes	Instances	Classes	Class distribution
Dataset 0	9	2075259	n/a	n/a
Dataset 1	9	68320	2	32893, 35427
Dataset 2	6	68320	2	32893, 35427
Dataset 3	6	68320	3	23946, 25181, 19193
Dataset 4	6	136632	2	65924, 70708
Dataset 5	6	136632	3	46649, 48622, 41361

4.1 Multi-class DR for streaming data

The following definition for the online estimation of multi-class detection rate will be assumed [10, 20]. First, a per class detection rate is defined as

$$DR_c(t) = \frac{tp_c(t)}{tp_c(t) + fn_c(t)} \tag{2}$$

where t is the exemplar index, and $tp_c(t)$, $fn_c(t)$ are the respective online counts for true positive and false negative rates for class 'c' up to this point in the stream.

The multi-class Detection Rate now has the form

$$DR(t) = \frac{1}{C} \sum_{c=[1,..,C]} DR_c(t) \tag{3}$$

4.2 AUC style metric for streaming data

The 'area under the curve' (AUC) metric is typically deployed within the context of summarizing the properties of a 'receiver operating characteristic' (ROC) [13]. A ROC, in turn, is used to quantify the robustness of a model in relation to a pair of 'orthogonal' performance metrics[5] with the goal of defining the best 'operating point' trading off the two metrics.

In the case of streaming data we note that we desire the performance metric (multi-class DR of Eq. (3)) to be maximized at *all* time steps. Thus, the 'area under the curve' over the duration of stream has the form

$$streamAUC = \frac{1}{T} \sum_{t=[1,..,T]} DR(t) \tag{4}$$

Note that this represents an arithmetic process for approximating the 'area under the curve,' so interpretations of AUC values specific to ROC source data should not be made [13].

5 Experimental Setup and Results

Each preprocessed dataset was divided in two parts. The data from 2006 was used to define the FXGP parameterization, whereas the data from January 2007 to November 2010 was used for the data stream experiments. Thus, the first TI–DT champion was deployed, starting from January 1, 2007, whereas the last week of the 2006 was used to train and validate agents during the very first Train—Validate—Label cycle (336 and 672 candlesticks for 30-minute and 15-minute candlesticks respectively) in case of FXGP and to do initial training of MOA classifiers.

5.1 Impact of candlestick preprocessing

The FXGP algorithm assumes data in the form of price time series, i.e. data preprocessed as candlesticks (datasets Data 2... Data 5, Table 9). Several models from the open-source MOA framework [3] are used to characterize the effectiveness (or otherwise) of the data pre-processing into candlesticks. Specifically, the 'No Change' classifier, Naive Bayes and Hoeffding Trees will be used. The No Change classifier represents a 1-bit state-machine in which the 'prediction' reflects that of the last prediction as long as the last prediction was correct. A missed prediction results in the state changing to predict the new class. Such a predictor does not make any use of the attribute information, only knowledge of the labels. Previous research has demonstrated that such a naive model is capable of surprisingly strong performance when there is a low amount of mixing (turnover) in consecutive labels [2]. Both the Naive Bayes and Hoeffding Tree classifiers represent well known algorithms for streaming data classification and appear in a number of monographs, e.g. [1, 7].

The results obtained in the case of all available attributes (Dataset 1) and 30-minute candlesticks (expressed relative to a single attribute) are shown in Figure 3. Naturally, the No change classifier is not impacted by the preprocessing of the attribute data (it never employs any attribute information). Conversely, the Naive Bayes classifier does not show any particular preference, whereas predictions using the Hoeffding Tree do improve when using candlestick preprocessing. Expressing this using the stream AUC metric from Eq. (4) Section 4 indicates that this can be quantified in terms of a 4% improvement (Table 10). Hereafter we will assume candlestick preprocessing for the remainder of the study (Dataset 2 through 5).

[5] Typically, true positive rate and false positive rate are assumed, but precision and recall are also widely used [13].

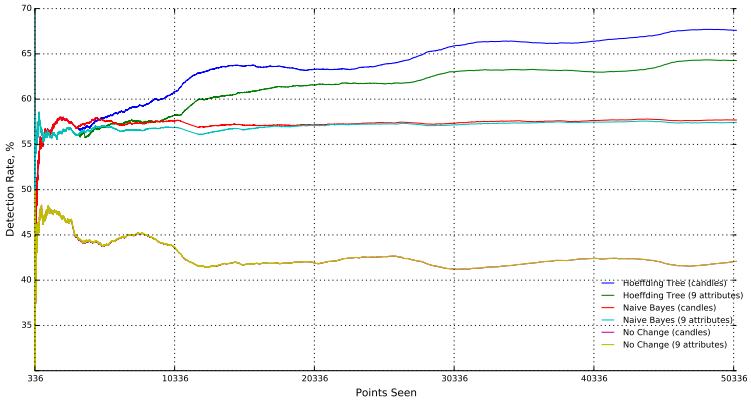

Figure 3: MOA. Full set of attributes (Dataset 1) vs 30 minutes candlesticks (Dataset 2). 'Candles' and '9 attributes' denote attribute type employed.

Table 10: AUC summary statistics. Raw data versus Candle preprocessing

Classifier	Dataset	AUC
MOA Hoeffding Tree	Dataset 2	0.638
MOA Hoeffding Tree	Dataset 1	0.613
MOA Naive Bayes	Dataset 2	0.573
MOA Naive Bayes	Dataset 1	0.571
MOA No Change	Dataset 2	0.426
MOA No Change	Dataset 1	0.426

5.2 Dataset 2 and 3 — 30 minute candlesticks

Each experiment includes a single run for MOA ('Hoeffding Tree', 'Naive Bayes' and 'No Change') and 100 independent runs for FXGP. Hence, results for GP reflect a spread from the worst of the 100 runs to the best. The results for binary classification of 30-minute candlesticks are shown in Figure 4 and the results of ternary classification (30-minute candlesticks) are shown in Figure 5. Table 11 details the streaming AUC statistic for both binary and ternary classification tasks.

Given the formulation adopted for labelling the data, adding a third class will only increase the potential for label mixing relative to the binary case, hence the reduction in performance as measured by the stream AUC statistic reflects this bias. Indeed, all classifiers return a reduction in detection rate when going from the binary to ternary formulation. The relative ranking of the models (in terms of Detection Rate) between each formulation of the dataset remains unchanged; in particular, No change $<$ Naive Bayes $<$ Hoeffding Tree $<$ FXGP.

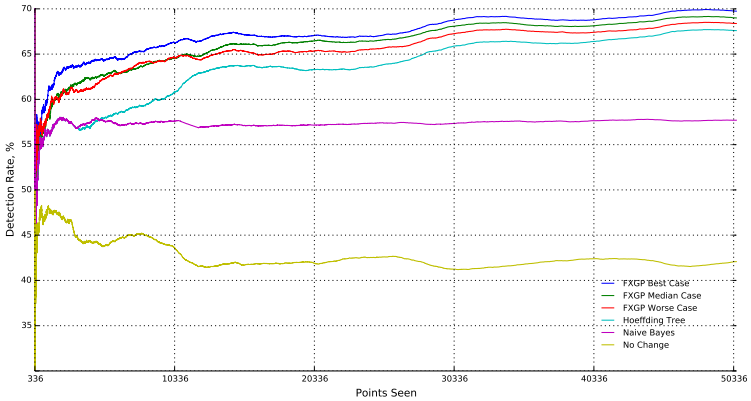

Figure 4: Dataset 2: 30 minute candlesticks, binary classification task.

Table 11: AUC summary statistics, Dataset 2 and 3: 30 minute candlesticks, binary and ternary classification

Classifier	binary (Dataset 2)	ternary (Dataset 3)
FXGP best	0.673	0.554
FXGP median	0.663	0.536
FXGP worst	0.657	0.532
MOA Hoeffding Tree	0.638	0.493
MOA Naive Bayes	0.573	0.459
MOA No Change	0.426	0.354

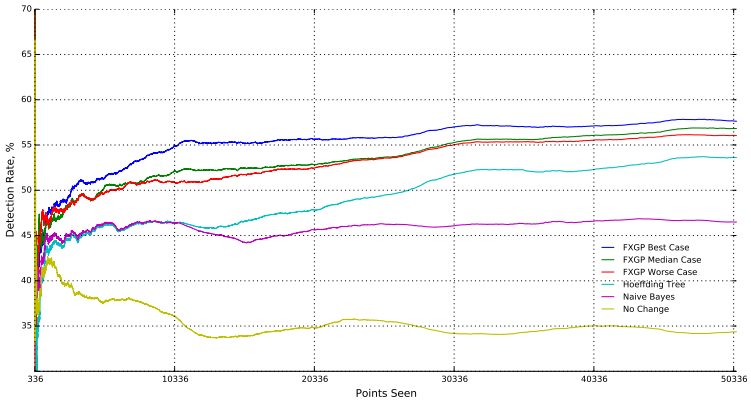

Figure 5: Dataset 3: 30 minute candlesticks, ternary classification task.

5.3 Dataset 4 and 5 — 15 minute candlesticks

The results of assuming preprocessing using the 15-minute candlesticks are shown in Figure 6 and Figure 7, for the binary and ternary classification tasks respectively. Table 12 summarizes the resulting quantification as reflected by the stream AUC metric.

The No class classifier again represents the worst case detection rate throughout. Likewise, the relative ranking of the other models is unchanged from the case of candlesticks estimated over a 30 minute interval. However, what is now interesting is that comparing FXGP performance using 15 minute candlesticks and 30 minute candlesticks returns an increase in performance when estimating the candlestick over the shorter period. Indeed, testing for the significance of this using an unpaired two-tailed Student's t-test (99% confidence interval) indicates that a statistically significant improvement appears in the case of FXGP on Dataset 4 versus 2 and Dataset 5 versus 3.[6] Also of note is that the eventual DR at the end of the stream might be higher in the case of the 30-minute scenarios

[6]Corresponds to a p-value $< 2.2 \times 10^{-16}$ in both cases w.r.t. the 100 runs.

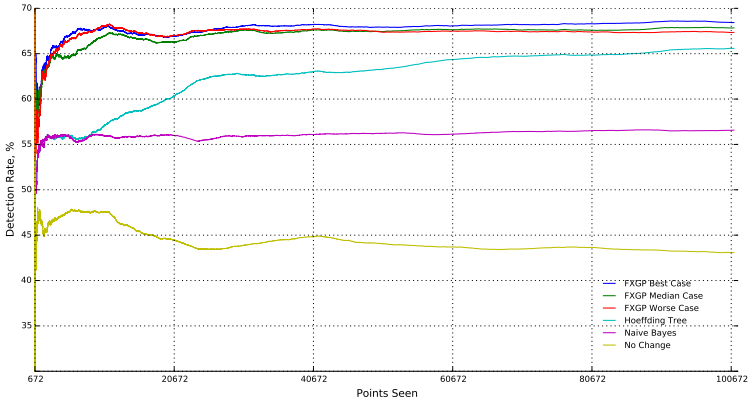

Figure 6: Dataset 4: 15 minute candlesticks, binary classification task.

Table 12: AUC summary statistics, Dataset 4 and 5: 15 minute candlesticks, binary and ternary classification

Classifier	binary (Dataset 4)	ternary (Dataset 5)
FXGP best	0.678	0.566
FXGP median	0.672	0.557
FXGP worst	0.673	0.550
MOA Hoeffding Tree	0.626	0.503
MOA Naive Bayes	0.561	0.468
MOA No Change	0.443	0.358

under FXGP (compare Figures 4 to 6 and likewise Figures 5 to 7). However, in the case of the 15-minute scenarios the point in the stream at which, say, a 55% DR is first reached is much earlier. Given that there is little / no regression in DR, the corresponding stream AUC is significantly higher.

5.4 Quantifying the role of retraining

Sections 5.1 to 5.3 implicitly assumed that the individual household electric power consumption dataset would benefit from retraining or an explicitly online / streaming approach to model building. In order to provide some quantification for this perceived benefit we explicitly turn off the retraining step for FXGP. Thus, the first 'Train—Validate' cycle is performed (relative to the same one week of data) and thereafter the TI–DT champion identified during the Validate stage is deployed to make the predictions thereafter. Figure 8 reflects the distribution of DR across the remainder of the stream (30 minute candlestick, binary classification). Note that for comparative purposes, DR for the curves for Hoeffding Tree, Naive Bayes and No Change classifier still reflect training *throughout* the stream, whereas the FXGP curves reflect performance without retraining. Previously, FXGP was able to return worst case performance that exceed the best baseline model. Now, without retraining, best case performance fails to reach that of the Hoeffding Tree and worst case performance is considerably worse than Naive Bayes (although still better than the No Change baseline). Indeed, the worst case profile reflects a detection rate of $\approx 50\%$ or no better than labelling the data all one class. The wide variation in performance of FXGP reflects the difficulty in choosing a model when it is not possible to guarantee that the training data is representative of the underlying task (as is the case when non-stationary properties exist). Moreover, note that this is the same set of GP models that when retraining is enabled perform better than all baseline streaming classifiers. In short, change detection and retraining is central for correcting GP as the stream progresses.

5.5 Quantifying the cost of permitting retraining

FXGP does not train incrementally, but when retraining is triggered, then the content of TI and DT populations are both reset and evolution begins from a completely new initialization of random individuals. Such an approach was assumed following a benchmarking study comparing incremental evolution (some or all the population is retained between evolutionary cycles) to the 'flush-and-restart' methodology assumed here [17]. Naturally, this

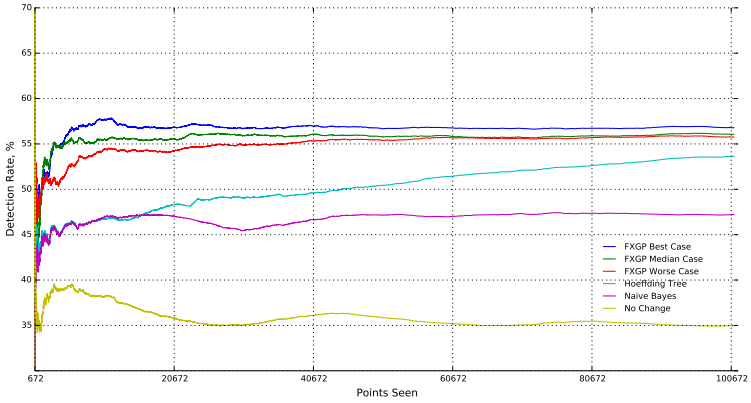

Figure 7: Dataset 5: 15 minute candlesticks, ternary classification task.

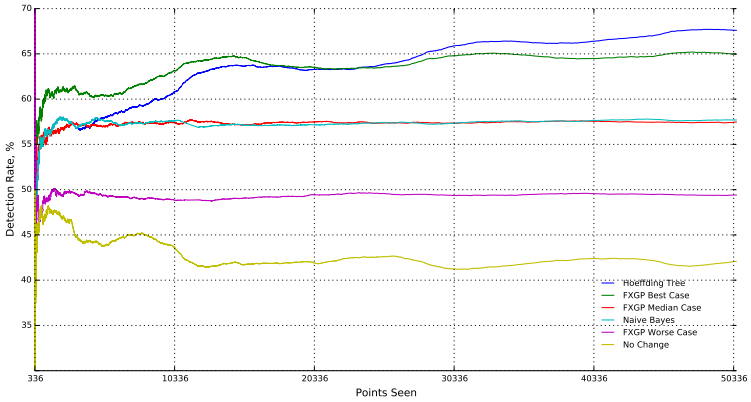

Figure 8: Dataset 2: 30 minute candlesticks, binary classification task — 100 runs of FXGP *without* retraining. Hoeffding Tree, Naive Bayes and No Change classifier retain retraining.

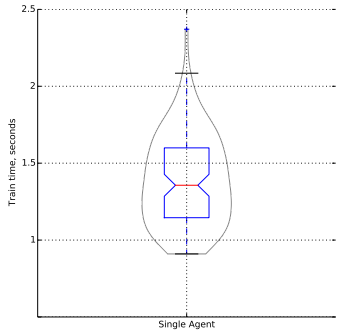

Figure 9: CPU time in seconds necessary to coevolve a completely new TI—DT champion. Distribution estimated over 100 runs.

also has implications for the (computational) cost of rebuilding a champion solution, potentially setting a limit to the degree to which real-time operation can be supported. We therefore quantify this cost from the perspective of the CPU time to coevolve an entirely new TI—DT pair (Figure 9).[7]

Given that the time between retraining events is several orders of magnitude lower than the interval between new data samples (15 or 30 minutes) it is readily apparent that FXGP is capable of real-time operation under this task domain. Naturally, the number of retrigger events is a function of the number of classes (difficulty of the task) and cardinality of the data stream, but in all cases remains $< 1\%$ of stream content. Thus, between ≈ 200 to ≈ 1000 retraining events are sufficient to maintain synchronization with the non-stationary properties of the stream. However, without retraining, it is generally not possible to identify good predictors (Figure 8).

6 Conclusion

This work makes three specific conclusions: Firstly, GP can be applied to streaming data classification tasks and remain computationally feasible without recourse to specialist hardware / software support (e.g. no use was made of multi-threading or multi-core execution), while the performance of the classifier is competitive with that of other algorithms. Secondly, we demonstrate the utility of an AUC style metric for providing a more general way to quantify performance of a classifier over the duration of the stream as a whole. The AUC metric could potentially be applied to any number of streaming performance indicators. Thirdly, we demonstrate the effectiveness of preprocessing data using a candlestick representation. Naturally, such a representation assumes that there is sufficient data present in the stream for construction of each candlestick. If the data was only available at, say, 30 minute intervals and this was also the rate at which predictions were required, then preprocessing using candlesticks would not be appropriate.

[7] 2.8 GHz iMac, Intel Core i7 CPU.

Acknowledgment

The authors gratefully acknowledge support from the NSERC CRD program (Canada).

References

[1] A. Bifet. *Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams*, volume 207 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2010.

[2] A. Bifet, I. Žliobaitė, B. Pfahringer, and G. Holmes. Pitfalls in benchmarking data stream classification and how to avoid them. In *Machine Learning and Knowledge Discovery in Databases*, volume 8188 of *LNCS*, pages 465–479, 2013.

[3] Albert Bifet. MOA: Massive Online Analysis Learning Examples. *Journal of Machine Learning Research*, 11:1601–1604, 2010.

[4] M. Brameier and W. Banzhaf. *Linear Genetic Programming*. Springer, 2007.

[5] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4):12–25, 2015.

[6] P. Fernandez-Blanco, D. Bodas-Sagi, F. Soltero, and J. Hidalgo. Technical market indicators optimization using evolutionary algorithms. In *ACM Conference Companion on Genetic and Evolutionary Computation*, pages 1851–1858, 2008.

[7] J. Gama. *Knowledge Discovery from Data Streams*. CRC Press, 2010.

[8] J. Gama, R. Sabastiao, and P. P. Rodrigues. On evaluating stream learning algorithms. *Machine Learning*, 90:317–346, 2013.

[9] M. Harries. Splice-2 comparative evaluation: Electricity pricing. Technical report, University of New South Wales, 1999.

[10] M. I. Heywood. Evolutionary model building under streaming data for classification tasks: opportunities and challenges. *Genetic Programming and Evolvable Machines*, 16(3):283–326, 2015.

[11] A. Hirabayashi, C. Aranha, and H. Iba. Optimization of the trading rule in foreign exchange using genetic algorithm. In *ACM Conference on Genetic and Evolutionary Computation*, pages 1529–1536, 2009.

[12] H. Iba and C. C. Aranha. *Practical applications of evolutionary computation to financial engineering*, volume 11 of *Adaptation, Learning, and Optimization*. Springer, 2012.

[13] N. Japkowicz and M. Shah. *Evaluating Learning Algorithms: A classification perspective*. Cambridge University Press, 2011.

[14] M. Kampouridis. An initial investigation of choice function hyper-heuristics for the problem of financial forecasting. In *IEEE Congress on Evolutionary Computation*, pages 2406–2413, 2013.

[15] M. Kampouridis and E. Tsang. EDDIE for investment opportunities forecasting: Extending the search space of the GP. In *IEEE Congress on Evolutionary Computation*, pages 2019–2026, 2010.

[16] M. Lichman. UCI machine learning repository, 2013.

[17] Alexander Loginov and Malcolm I. Heywood. On the impact of streaming interface heuristics on GP trading agents: an FX benchmarking study. In *Proceeding of the ACM Genetic and Evolutionary Computation Conference*, pages 1341–1348, 2013.

[18] Alexander Loginov and Malcolm I. Heywood. On the Utility of Trading Criteria Based Retraining in Forex Markets. In *Applications of Evolutionary Computation (EvoFIN)*, volume 7835 of *LNCS*, pages 192–202, 2013.

[19] Alexander Loginov and Malcolm I. Heywood. On evolving multi-agent FX traders. In *Applications of Evolutionary Computation (EvoFIN)*, volume 8602 of *LNCS*, pages 203–214, 2014.

[20] A. Vahdat, J. Morgan, A.R. McIntyre, M.I. Heywood, and A.N. Zincir-Heywood. Evolving GP classifiers for streaming data tasks with concept change and label budgets: a benchmarking study. In *Handbook of Genetic Programming Applications*, chapter 18, pages 451–480. Springer, 2015.