Evolutionary model building under streaming data for classification tasks: Opportunities and challenges*

Malcolm I. Heywood†

April 21, 2016

Abstract

Streaming data analysis potentially represents a significant shift in emphasis from schemes historically pursued for offline (batch) approaches tothe classification task. In particular, a streaming data application implies that: 1) the data itself has no formal 'start' or 'end'; 2) the properties of the process generating the data are non-stationary, thus models that function correctly for some part(s) of a stream may be ineffective elsewhere; 3) constraints on the time to produce a response, potentially implying an anytime operational requirement; and 4) given the prohibitive cost of employing an oracle to label a stream, a finite labelling budget is necessary. The scope of this article is to provide a survey of developments for model building under streaming environments from both the perspective of evolutionary and non-evolutionary frameworks. In doing so, we bring attention to the challenges and opportunities that developing solutions to streaming data classification tasks are likely to face using evolutionary approaches.

1 Introduction

Model based evolutionary computation (EC) is taken to to be synonymous with supervised learning, i.e., finding a mapping from a typically higher dimensional independent variable, $\vec{x}$, to a typically lower dimensional dependent variable, y, with credit assignment performed relative to label information. The goal of this article is to articulate the challenges and opportunities that follow for model based EC as we increasingly encounter the phenomena of streaming data. A shortlist of properties that make machine learning (ML) under the streaming task more of a challenge than when encountered under a non-streaming setting potentially includes some combination of the following:

- the underlying process creating the data might well be non-stationary / dynamic, implying that the dataset is subject to concept shift / drift. A classifier effective on one part of a stream is therefore not necessarily useful elsewhere. Conversely, the majority of 'offline' supervised learning frameworks assume that data is independent and identically

*This technical report is prepared for compliance to NSERC open source publication requirements. The original article should be referenced in all cases: M. I. Heywood (2015) ***Genetic Programming and Evolvable Machines***. 16(3):283–326 http://link.springer.com/article/10.1007/s10710-014-9236-y

†Faculty of Computer Science, Dalhousie University, NS. Canada. mheywood@cs.dal.ca

distributed (e.g., [59]). This is the basis for building models from a static sample (the training partition), validating against an independent sample of data, and testing for the generalization of trained models to a third independent partition.

- datasets are not necessarily finite, or at the very least, they are sufficiently large to preclude multiple passes, implying that a single pass constraint has to be assumed. When combined with the non-stationary nature of the stream this implies that learning needs to be a continuous 'online' activity, with solutions available at anytime during the progression of the stream.

- the cost of labelling the data implies that only a fraction of the data can be labelled. This places an additional requirement on the model to also detect change in the stream, and / or actively request label information on demand or make use of unlabelled data cf. online semi-supervised learning;

- the continuously evolving nature of stream content has implications for learning under class imbalance, as it is generally not possible to a priori 'stratify' the dataset.

Previous reviews or monographs have typically considered the streaming data task from the specific perspective of 'classical' ML without considering how features from model based EC might be applicable [17, 76, 77, 156]. Conversely, several monographs in evolutionary computation have surveyed the issue of EC in 'dynamic environments' (e.g., [51, 141]), but make little reference to parallel developments from ML or the streaming task in particular. The goal of this work is to attempt to provide some perspective on what can be gained from drawing on the rich literature available from both sources of study with respect to evolving model based solutions to streaming data tasks.

The following survey assumes two distinctions. Firstly, we draw a distinction between the goal of frameworks for model based EC (such as genetic programming (GP), learning classifier systems (LCS) or neuro-evolution) and optimization (as in, say, evolutionary strategies). In particular, the availability of gradient information and a closer coupling between representation space and search / error space provide additional information for evolutionary optimization (under dynamic environments) that do not exist under model based EC. Hence, the action of variation operators (acting on the representation space) can have a wide range of effects on the search space in GP (e.g., [116]), whereas the real-valued representations assumed under optimization tasks ensure a much closer coupling. Moreover, optimization tasks focus on the accurate 'tracking' of optima and place little emphasis on testing against unseen data or generalization.

Secondly, most of the research in machine learning for streaming data is associated with classification as opposed to function approximation / regression. This, on the face of it, is quite surprising. There is a long tradition of using model based EC for forecasting / prediction. However, the goal of forecasting is to predict the next instance of a sequence of data, after each prediction it is assumed that the true value for the dependent variable is known, and the process iterates.[1] Streaming data scenarios frequently do not conform to this mode of operation because a labelling budget is generally enforced, thus raising the issue of when to request label information. Having made this distinction, we will make reference to contributions from evolutionary optimization and (symbolic) regression when they directly contribute to streaming data issues as characterized above.

[1]Multi-step prediction implies that several predictions are made before the true values are known.

In order to develop the concept of model based EC for streaming data, a generic characterization for the streaming data task is introduced (Section 2), where this also leads to a discussion of the implications of streaming data on the bias–variance tradeoff. Section 3 introduces at a high-level various considerations pertinent to the task of model building under streaming data from three standpoints: general ML requirements, ensemble ML requirements, and generic EC requirements. Section 4 presents benchmarking issues specific to the streaming data context, divided into two broad themes: evaluation methodologies and benchmarking datasets. Section 5 reviews progress to date vis-à-vis advances in streaming data algorithms in general and opportunities for using model based EC. A concluding discussion with recommendations for future research follows in Sections 6 and 7 respectively.

2 Characterizing the streaming data task

Several distinct approaches have been taken to characterizing the nature of streaming data. The first attempts to place constraints on the relationship between training and test partitions, or identify the necessary conditions under which model building and generalization can take place (Section 2.1). This has been a particular concern of classical ML. Conversely, the focus of EC has (historically) been more associated with characterizing the types of change, with the goal of then establishing what properties the EC framework for model building under dynamic environments should retain (Section 2.2). A final subsection reviews the implications of concept change on the bias–variance dilemma frequently used to characterize properties of credit assignment in ML model building (Section 2.3).

2.1 Statistical frameworks

The independent (vector of inputs) and dependent discrete class labels – or $\vec{x}$ and y respectively – are frequently characterized using a statistical model (e.g., [81]). Thus, the data stream is defined as a continuous sequence of $(\vec{x}(t), y(t))$ pairs. However, label information (the dependent variable) is typically only provided by engaging an 'oracle' (e.g., human expert), in which case the stream is characterized by $\vec{x}(t)$ alone with $y(t)$ being available for a subset of t. Moreover, it is assumed that the 'training' and 'test' data correspond to consecutive finite length sequences. Such an assumption leads to the *shared distribution* assumption, i.e., both training and test partitions need to be generated by the same distribution, $p(\vec{x}, y)$, for generalization to take place.

The shared distribution assumption implies that the underlying process responsible for creating the data has the form:

$$p(\vec{x}, y) = P(y|\vec{x}) \times p(\vec{x}) \tag{1}$$

where $p(\vec{x})$ denotes the distribution of input data, as in attributes / features. Thus, feature change can be stationary or non-stationary and when non-stationary the transitions might be smooth (implying multiple processes are simultaneously present) or abrupt. $P(y|\vec{x})$ denotes the conditional dependencies between input and label distributions. Conditional changes imply that the label switches for the *same* input as the stream progresses. Dual changes imply that both $p(\vec{x})$ and $P(y|\vec{x})$ undergo variation (as the stream progresses).

The shared distribution assumption comes from the constraint that training and test distributions are still assumed to be suitably similar or $p_{tr}(\vec{x}, y)$ and $p_{ts}(\vec{x}, y)$ are statistically equiv-

alent. This is not to say that all stream behaviours will fit this framework, just that for the purposes of defining a basis for measuring generalization, such a constraint is adopted. Not all frameworks for learning under streaming data contexts subscribe to this constraint. Indeed, should two consecutive sequences of data not conform to this constraint, we have a requirement for 'change' or 'novelty' detection.

The shared distribution framework – or more generally a Bayesian causal framework – provides the basis for identifying six reasons for a shift between training and test distributions [168]:

- *Simple covariance shift:* change solely due to temporal variation in the input distribution, $p(\vec{x})$.
- *Prior probability shift:* for the same distribution of inputs, $p(\vec{x})$, the probability of labels, $P(y)$, vary. The implication of this is that (some subset of) data that was at one time labelled as say class 0, is associated with a different class label.
- *Sample selection bias:* represents a pre-processing or measurement bias. For example, user surveys as collected from a single medium (e.g., twitter) potentially result in a bias towards collecting information from a specific demographic. Thus, any prediction based on such a survey (the training partition) would likely not reflect the overall opinion of a voting public (the test partition).
- *Imbalanced data:* is a form of "data shift by design". That is to say, minor / major classes are intensionally over / under sampled in order to increase / decrease the sensitivity of a classifier to specific classes. However, the relative frequency of each class may vary over the course of a stream. The implication being that any sample will not necessarily represent all classes. This is also referred to as skewed data (Section 5.6).
- *Domain shift:* case of variation due to a lack of 'object invariances' in the original input attributes $\vec{x}$. Examples of this might be sensitivity to lighting intensity or more generally a requirement for movement invariant attributes.
- *Source component shift:* the same concept might be described from multiple sources (as in sensor fusion). However, the differing sources might result in the same concept being described at different points in time using different subsets of attributes.

2.2 Types of change

Under a general EC setting, multiple authors have characterized the types of change that a non-stationary task might assume [1, 25, 48, 51, 112, 141]. Three distinct scenarios appear, as summarized by Table 1. In the case of an underlying process described by random changes, the point at which a change takes place is unrelated to the observed variable, $p(\vec{x})$, or a previous change, $p(y, \vec{x})$. Processes of this type might represent sources of breakdown, such as motor or sensor failure. At the other extreme (bluebottom row, Table 1), the underlying property describing the stream is a *stationary* function of the current (and possibly previous) instance(s) of the observed variable, $p(\vec{x})$. Thus, variation in observed or dependent variables are entirely predictable and possibly of a cyclic nature. Conversely, the second scenario corresponds to processes that are 'complex'. Thus, there is an underlying relationship between changes $p(y, \vec{x})$, and previous states, $p(\vec{x})$, but the relation is too complex to identify, i.e., chaotic systems.

Table 1: Types of change to the underlying process. Adapted from [51].

Type of change	Characteristics
Stochasic process	changes that are independent of previous state or change
Complex process	non-random but subject to multiple feedback paths within the process (i.e., chaotic)
Deterministic process	non-random and predictable change

Hence, as the parameters describing a complex systems vary, then periods of predictable behaviour may exist before a phase change takes place and the environment essentially becomes non-stationary. Naturally, it might be possible to make predictions about complex processes, but only over relatively short predictive horizons, e.g., as in weather forecasting. We also note that the three categories of this table are essentially extremes sampled from a continuum of processes. Delimiting when one process categorization 'morphs' into another is likely to be quite subjective.

Morrison and Branke also emphasize the relative size and frequency of change [25, 141]. Both properties have implications for the stability versus plasticity tradeoff (Section 3), where from the perspective of model based evolution, this is related to the balance between exploitation and exploration in credit assignment. Abbass *et al*. emphasize a binary categorization in which sources of change are either 'model boundary' changes or sample changes [1]. Model boundary changes are associated with changes to the underlying process responsible for generating the data (e.g., cyclic or stochastic variation in the generating process), whereas sample changes are associated with biases introduced by issues unrelated to the underlying data generation process (e.g., noise, class distribution / skew (within a finite sampling period)) or factors associated with the ML interface to the data stream.

2.3 Bias–variance tradeoff

Setting aside the online versus batch concept of data access / credit assignment, streaming data scenarios can be considered an example of learning from a 'large' dataset. The bias / variance characterization of error from a machine learning algorithm in general are summarized as follows (e.g., [59, 129]):

- Error variance: measures the sensitivity of a model to a particular subset of data or, equivalently, the sensitivity to the underlying model complexity. Thus, under classification tasks, variance characterizes how much decision boundaries change as a function of data / model initialization.

- Error bias: measures the degree to which the typical response of a model (i.e., w.r.t. all data) varies from the desired value. Thus, bias is associated with basic topology (of a model) and variance characterizes a specific parameterization.

With these definitions in mind, most ML algorithms have concentrated on variance reduction [23], particularly given that in the case of classification tasks the bias–variance tradeoff is multiplicative and non-linear [59]. Thus, classification using Naive Bayes is known to emphasize variance reduction alone, whereas methods for combining multiple weak learners are potentially effective at both variance and bias reduction [23, 129]. Over a series of benchmarks, Brain and Webb demonstrate that the overall contribution of error variance under classification

tasks decreases as the size of the dataset increases, whereas the contribution of bias terms may actually increase [23]. Gama notes that such a finding may also be significant to the context of model building under streaming data [76, 77]. However, this does not imply that variance reduction should be ignored. Indeed, given that streaming data implies that it is necessary to construct models from small subsets of the data (e.g., a sliding window), then variance reduction is still the principal objective of recent ML approaches to streaming data classification (e.g., [203]).

2.4 Discussion

Naturally, the above characterizations are independent of the approach to model building. However, given the much tighter coupling between representation and credit assignment assumed in the case of classical ML, and the increasing incidence of Bayesian frameworks, it is not surprising that a conditional probability model lies at the centre of ML characterizations for the streaming task. Conversely, EC as applied to dynamic environments has placed more emphasis on the types of change. Taken purely from a historical perspective, most benchmarking results to date assume two basic characterizations of the concept shift task: discrete switches between different underlying processes or some form of continuous variation in the underlying process.

One area that is often an oversight in the characterization of concept drift in the streaming data task is that of variation in attribute support. On the face of it, this is implicit in the concept of, say, change as applied to the independent variable, $\vec{x}$. In practice, most benchmarking assumes that all attributes are employed, whereas associating different subsets of attributes with different generating processes results in multiple forms of the curse of dimensionality [111, 196]. Recently an ε-greedy approach was proposed with linear classifiers in which new samples were taken from the stream in order to periodically resample the attribute space [189]. We note that GP will naturally support stochastic resampling of the attribute space as part of the training cycle.

3 Identifying generic ML properties for the stream learning task

The properties considered appropriate to the task of model identification under streaming data have been developed independently by the generic ML versus EC communities. We again begin by establishing two ML perspectives – generic ML issues and somewhat more specific issues relative to ensemble methods – before reviewing properties pertinent to EC model building under streaming data.

3.1 Generic ML perspective

Relative to offline (or batch) model identification, as performed against a single training–validation–test partition, the streaming data task presents various additional generic challenges summarized as follows [57, 80, 76]:

Computational: both time and memory efficiency should be constant (and preferably linear) with respect to the data throughput of the stream. Assuming that the stream is labelled, this has at least two interpretations: continuous / online updating (e.g., [70, 149]) versus some form a batch / incremental update policy (e.g., [137, 154]). Batch update policies are only likely

to approach this objective 'on average'. However, depending on the data rate of a streaming application, this may be perfectly acceptable. Conversely, if the stream is not labelled, then credit assignment is potentially limited to the rate at which labels are provided.[2] Section 5.3 investigates this issue further in the case of change detection without the aid of labels. We also recognize that not all attributes necessarily carry an equal evaluation cost. Anytime interruptible ML algorithms attempt to incrementally construct a model given prior a computational budget for attribute evaluation [66]. Thus, a decision tree representation might be assumed on account of bluethe ability to build models without necessarily utilizing all attributes.[3] Indeed, attribute evaluation costs as well as model accuracy are used to bias tree construction.

Single pass: given that data is being received on a continuous basis, a model can only index data over some finite interval relative to the current time step, t. This is generally taken to imply that either a sliding window or consecutive blocks of non-overlapping data characterize the interface to the stream. Once there is a shift in the location of the sliding window / data block, it is not possible to revisit data previously encountered. An exception to this is when the learning algorithm introduces a finite archive of data that is in some way pertinent to the identification of useful models (e.g., active learning). A tradeoff naturally exists in terms of: 1) archive size versus real-time operation; and 2) the 'age' versus diversity of exemplars retained in such an archive. This topic will be discussed more explicitly within the context of EC methods in Sections 3.3 and 5.1.2.

Anytime operation: independent from the process of model identification (continuous or batch), the ML framework must be able to provide estimates for the dependent variable, y, at any point in the stream. From an EC perspective this implies that a 'champion' individual / ensemble must always be available. This naturally raises questions regarding how such a champion is identified, given that a convenient definition for a validation partition might not be readily apparent (Section 3.3.3).

Generalization: is relative to the point in the stream that a prediction is made. Depending on the type of stream, various forms of generalization might be applicable. In the case of a stationary process, the single pass constraint would imply that we might desire generalization (under test) to approach that of batch learning with the same computational cost. Conversely, for non-stationary streams it might be expected that model accuracy should be maintained until a 'significant' change in the stream is detected, i.e., the shared stationary distribution assumption of Section 2.1 is suitably well maintained. After a significant change is detected, updates to the model should be performed before a new solution can be identified and operation resumed.

Stability versus plasticity: relates to the tradeoff in balancing the capacity to react to change versus losing the capability to generalize to the underlying process, i.e., the balance between new and prior knowledge. Thus, distinctions need to be made regarding what of previous model(s) should be retained versus introducing completely new model(s) [65, 88]. From a streaming data perspective the limited / incremental nature in which data is presented results in virtual concept drift[4] (necessitating updates to current knowledge) versus 'actual' concept drift in which case replacing previous knowledge is more appropriate [65, 194]. Several authors note that there may also be an inherent correlation with the stability–plasticity dilemma and the rate at which credit assignment is performed [65, 84, 138]. Thus, ML algorithms based on exemplar-wise updating tend to emphasize plasticity over stability, whereas algorithms which

[2] A caveat being semi-supervised learning under streaming data, Section 5.4.

[3] See model building with embedded versus wrapper or filter frameworks for attribute selection [114].

[4] See also 'sample selection bias', Section 2.1.

update relative to a batch / chunk of data tend to emphasize stability.[5]

3.2 Ensemble ML perspective

Ensemble methods represent a framework for combining the predictions from multiple base learners into a single decision [129]. Indeed, both bagging and boosting represent an obvious starting point for developing ensembles under streaming contexts [149, 152]. The motivation for applying ensembles to streaming data (in addition to the stronger performance of ensemble based learning) is that changes can more easily be accommodated through the addition or removal of ensemble members than through incremental refinement to a single model, i.e., incremental refinements to a single model might be more sensitive to epistatic interactions. An early characterization of ensemble methods as applied to streaming data assumed three categories of which only one category reflected addition / subtraction of ensemble members [113]. However, recent research has concentrated on approaches that actively grow / replace ensemble membership. Moreover, it may even be possible to maintain multiple ensembles in order to: 1) react to cyclic behaviours in the stream [160]; or 2) explicitly maintain multiple ensembles with different diversity properties [139]. A summary of potential design decisions associated with ensembles as applied to streaming data might therefore include:

- Ensemble base learner diversity: Different base learners or mixtures of base learners comprising the ensemble might be considered depending on the type of stream [65, 102].
- Incremental change to current knowledge (e.g., [107, 160]) as opposed to the outright dropping of previous knowledge (e.g., [68, 169]);
- Adapting the weight associated with learners versus no weighting: Weight adaptation represents an intermediate level of refinement in which the models denoting the ensemble remain unchanged but their relative contribution to the voting is modified [2, 90, 154]. Conversely, weightless frameworks emphasize plasticity and tend to drop weaker ensemble members immediately (e.g., [81, 169]);
- Classifier weight adaption versus data instance based weight adaptation: The weighing of votes from an ensemble is generally a function of either classifier performance [54, 65, 154] or of the data from which a member of the ensemble was constructed (e.g., [20, 152]);
- Identification of ensemble member for replacement: Various heuristics have been proposed for targeting the ensemble member for replacement when performance as a whole is deemed to be poor, e.g., replace the oldest [169] or member with least 'contribution' [107, 172].
- Role of diversity on ensembles: Within an environment undergoing change, diversity provides faster reaction times to a change, but does not necessarily facilitate fast convergence to the new concept [138, 167]. One implication of this might be that the amount of diversity / plasticity needs to in some way 'match' the amount of concept drift / shift in the stream. This result mirrors the scenario in GP where conflicts can appear between fitness improvement and maintaining population diversity (e.g., [64]).

[5]We note that this in itself is a function of assumptions made regarding parameterization. At some point decreasing the size of data chunks will result in performance approaching that of exemplar-wise updating. Observations of this type have informed the use of differing pairwise sliding window durations (Section 5.1.1) and evolved temporal features, e.g., [126, 188].

Table 2: Basic design decisions for ML ensembles under streaming data

Task	Parameters
Constructing new ensemble member	Diversity of base learner [65, 102] Sample stream data using Boosting versus Bagging [149, 152]
Identify ensemble member for replacement	Age based heuristics [169] Performance based heuristics [107, 172]
Class imbalance	Effect of sampling biases [34, 55, 86, 190]
Drift management	Incremental updating of current models [107, 160] Adapt voting weights [2, 90, 154]
Shift management	Outright replacement of one or more ensemble member [81, 68, 169]
Diversity management	Impact on capacity for change [26, 138, 167]

- Change detection is now potentially a property of the behaviour of the ensemble: With different models sensitive to different properties of a stream, analysis of variance has been proposed as a mechanism by which members of an ensemble can be targeted for replacement [203].
- Separation of duties: Different aspects of the decision making process can be distributed to specialist parts of a wider framework, potentially resulting in hybrid architectures. For example, the combination of entropy based change detection (relative to sliding window content) with random forest style ensembles [2] or the pairing of ensembles based on Hoeffding decision trees with Kalman filter style change detection [18].
- Hybrid frameworks: If labels are freely available or available at a sufficiently low cost, it becomes feasible to pursue the incremental refinement of all classifiers currently available as well as test for the introduction of an entirely new classifier to the ensemble [27]. Adopting such a dual strategy has the advantage of providing the ability to accurately track concept drift as well as reacting to concept shift (gradual versus sudden change).

Table 2 summarizes the role of different design decisions when constructing an ensemble. Various redundancies are apparent, in that several different design decisions might have the same effect. There has been little work on identifying best practices for ensemble design in streaming data, in part due to the lack of formal results (discussed further below). However, there is also a potential opportunity for deploying hyper-heuristics for the purpose of more explicitly searching the space of algorithms for ensemble design (see Section 7).

Many open issues remain, not least regarding appropriate metrics for quantifying ensemble diversity, thus making it difficult to answer questions regarding credit assignment. Some results exist for static tasks, including the role of ensemble voting margins in determining generalization error [159]. Moreover, the ensemble misclassification rate can be expressed in terms error associated with individual members and a diversity term. The diversity term can be characterized as being 'good' or 'bad' [26], a result that has a corresponding observation in EC [191]. Under non-stationary data, it has been established that reducing the absolute value for the ensemble margin produces an equivalent increase in diversity [167].

A second open question is in regard to the method assumed for combining the outcome from multiple models under a non-stationary task. Specifically, bounds on the expected loss

associated with different ensemble frameworks are only available for the training partition (e.g., Adaboost [75]), whereas concept shift / drift will likely invalidate such an estimate. Various constraints have been proposed in an effort to identify "well behaved" loss functions, e.g., convex versus non-convex. However, attempts to provide formulations for ensemble methods under non-stationary data currently lack transparency [56]. Finally, we note that once a change is detected, the issue of whether to retain in whole or in part the material from the current ensemble, versus dropping all models and retraining from scratch, can potentially be addressed by maintaining multiple ensembles with *different* diversity properties [139]. Maintaining such diversity under label budgets remains an open question.

3.3 Generic Model-based EC perspective

In adopting an evolutionary approach to model building a frequently made observation is that we need to resist the tendency for the population to 'converge' while also introducing mechanisms that explicitly promote the capacity to react to change. Such a statement comes from general ML observations regarding 'stability versus plasticity' of ensemble methods as applied to online learning tasks with concept drift [138], experiences from evolutionary methods as applied to dynamic optimization tasks [22, 51], and empirical studies from related tasks such as neural evolution under partially observable non-Markovian reinforcement learning tasks [135, 166]. With this in mind, three basic properties will be adopted for the purposes of the following discussion: evolvability / plasticity, memory, and diversity.

Evolvability / plasticity characterizes the efficiency by which 'useful' phenotypic variation is generated, given the current state of the environment. As pointed out in the introduction, unlike ML or EC as applied to (dynamic) optimization tasks, the mapping between representation and search spaces is not tightly coupled under GP. Thus, it is not generally possible to provide an ordering of the representation space (as is possible under real-valued representations) [116]. Instead evolvability is related to the development of neutrality, support for evolving the genotypic to phenotypic mapping, and modularity (Section 3.3.1). Modularity, for example, enables model based evolution to delimit the scope of variation operators, thus clarifying credit assignment [67]. Moreover, when changes to the task have structure (as opposed to random variation) then support for modularity facilitates faster rates of adaptation [151].

We recognize two potential sources of **memory**: the (population of) candidate models and the (subset of) task instances from which models are evolved. Thus, as EC maintains multiple solutions in parallel, the potential exists for switching between different candidate solutions during the stream. We note that in the case of slowly changing environments, the population as a whole acts as a source of genotypic memory [186]; especially if the form of variation is limited to past instances of the environment or combinations thereof. Conversely, success under rapidly changing environments implies that a population is capable of adapting to multiple environments simultaneously [186].

Supporting such a capability implies that appropriate mechanisms will be required to 'manage' such complexity. In the following we explicitly note contributions from **diversity maintenance**, modularity (cf., evolvability) or multiple forms of reward [191]. Reward is related to both the organism and the environmental context. From the perspective of streaming data, the latter represents the subset of data against which evolution is performed, i.e., not all exemplars are equally informative. In the following survey we highlight the role of *active learning* (cf. coevolution) in identifying useful instances of data from the stream to learn from. The implication being that both candidate solution and task instance receive some form of reward (Section

3.3.2).

3.3.1 Evolvability

A working definition for *evolvability* was recently proposed as "the capability of a system to generate adaptive phenotypic variation under certain environmental conditions and to transmit it via an evolutionary process" [94, 96]. Turney characterizes this in terms of two equally fit individuals, A and B [173]. If the children of A are likely to be fitter than children of B, then individual A is more 'evolvable' than that of B. This implies that evolvability is not a direct function of fitness (as in performance) based selection, but rather a function of other genotypic properties, e.g., neutrality, the genotypic-to-phenotypic mapping, or the inherent plasticity of a representation (Baldwin effect).

Wang and Wineberg build on Turney's observations and propose to characterize evolvability through [191]: 1) the ability to improve fitness (as opposed to the absolute value of fitness, or an EC perspective); and 2) the amount of genotypic variation (a biological perspective). With this in mind a three-population model is adopted in which each population assumes a different performance metric: 1) absolute scalar fitness (core population), 2) offspring outperforming their parents (fitness change; subpopulation), and 3) offspring which are most genotypically distinct (genotypic change; subpopulation). In doing so, the authors are explicitly rewarding evolvability as well as absolute fitness. Moreover, the authors demonstrate that diversity is maintained through evolvability and not diversity for diversity's sake. Wang and Wineberg also make use of Price's equation to adapt the size of the two secondary populations, where adaption of population size in dynamic environments is a recurring theme. Specifically, adapting population size in order to maintain a constant rate of genotypic substitution potentially represents a mechanism for adapting to a slowly changing environment in GP [95, 182].

Hu and Banzhaf make the observation that a continuous background level of neutrality in GP facilitates 'bursts' of (phenotypic) variability once the environment undergoes a change [94]. Neutrality can also be viewed as a memory mechanism by which previously useful or entirely new genotypic material can be switched in and out of the phenotype. Wagner *et al*. utilize such a mechanism in a GP framework for forecasting [188]. Sources of phenotypic variation are potentially related to the *plasticity* of the genotype-to-phenotype mapping [187] and might therefore be characterized in terms of the amount of: variability and neutrality.

Providing explicit support for genotype-to-phenotype mappings in EC in general may then have beneficial properties under dynamic environments [61]. The GE^2 framework provides a scheme for evolving pairs of genotypes, one to specify a 'meta grammar' and the second a 'solution grammar' [148]; the latter defining specific GP solutions in terms of the meta grammar. Naturally, the use of a meta grammar introduces additional paths through which the same ultimate phenotype can be discovered. Moreover, additional paths also exist for the introduction of neutral genetic material or gene duplication. Later work extended the solution grammar to provide a much stronger representation for evolving constants, a characteristic frequently overlooked in GP, and demonstrated to provide significant advantages under stock trading tasks [50]. Open questions include defining the most effective scheme for coevolving meta grammar and the individuals using an instance of the meta grammar. Other researchers have also demonstrated the utility of GP with genotypic-to-phenotypic mappings under dynamic environments, this time with an emphasis on identifying the most appropriate instruction types [195].

The capacity to support modularity represents one of the earliest factors explicitly articu-

lated as pertinent to model building under dynamic task domains [163]. However, for such modularity to demonstrate more than a mere sum of its parts, it is necessary for there to be non-trivial interdependencies between modules or 'near decomposibility' [192]. This then has implications for the type of search process that are capable of efficiently manipulating modules. Watson and Pollack make the case for multiple levels of selection [192]. Recently, the (neuro-) evolution of modularity itself was demonstrated when the cost of 'connectivity' was also included in the reward scheme [36]. Moreover, when the properties of the environment (goals) undergo structural variation, then the rate of evolutionary variation when the representation supports modularity has been shown to be significantly faster than with non-modular representations [103]. Further schemes explicitly reporting on the utility of modularity in dynamic environments include: gene-duplication under neuro-evolution [30], and a role for automatically defined functions [14].

3.3.2 Memory

Memory – relative to evolution of models – for the most part is associated with maintaining multiple candidate solutions simultaneously. Dempsey *et al.* recognize two forms: implicit and explicit [51]. Typically, **explicit memory mechanisms** are related to reusing previously evolved individuals as seeds when events are encountered that trigger the archiving of individuals for later use [63]. This brings us back to evolvability, i.e., sufficiently strong parents give rise to offspring with even better adaptations. Moreover, explicit memory mechanisms also imply a need for memory management to protect the development of potentially useful genetic material for *later* use.

Multiple works based on genetic algorithms assume some form of distance metric based on genotypic similarity (e.g., [140, 198]). From the perspective of EC both genotypic and phenotypic measures of diversity appear to be sensitive to the application on which they are applied [28]. Related research with ensemble algorithms under streaming data indicates that the explicit maintenance of diversity across multiple models is beneficial, although caveats apply (Section 3.2). Conversely, several forms of neuro-evolution applied to non-Markovian policy discovery tasks have been able to make explicit use of genotypic metrics for diversity maintenance (e.g., speciation in [135, 166]); however, evaluation under streaming data tasks has not to date been performed. Mechanisms for explicitly maintaining diversity will be discussed further in Section 3.3.3.

Implicit memory mechanisms relate to memory through genotypic redundancy. Under the guise of genetic algorithms, a lot of attention has been paid to diploid representations [121, 174, 198]. Again, these have not tended to carry over to the case of GP. Part of this is likely due to the widespread abundance of code bloat (introns) in GP [24, 116]. As such, introns[6] can act as an alternative mechanism by which previously functional code are reintroduced (see also neutrality under 'Evolvability', Section 3.3.1). Such an implicit memory mechanism has been shown to be effective under GP applied to dynamic time series forecasting [188]. Indeed, Dempsey *et al.* note that such alternative implicit memory mechanisms have the additional advantage of avoiding the need to introduce suitable 'dominance functions' to define under what conditions to switch between the dominant versus recessive material. A third form of implicit memory is also available in the form of the population model itself [155] (i.e., multi-

[6]Introns, although non-coding for proteins in biology, appear to describe RNA that play an important role in gene regulation in eukaryotes [35]. In the case of GP, there is generally little or no distinction between genotype and phenotype, and more non-functional code observed than functional code [24, 116].

ple individuals retain the same 'building blocks'), albeit configured in different ways – thus, searching multiple parts of the search space simultaneously. However, this is only useful if suitable mechanisms can be found for maintaining population plasticity and / or diversity (see Sections 3.3.1 and 3.3.3 respectively).

Finally, memory has more recently been conceived of from the perspective of *data archiving*, where this represents a form of **active learning** (Figure 2). In this case, model building is performed against the content of the data archive ($DS(i)$, Figure 2) as opposed to all the stream content ($SW(t)$, Figure 2). Under an active learning context the goal is to solve the dual learning task of caching what is most informative to learn from, as opposed to merely treating all data as equally relevant (see also 'scaffolding' [65]). More pragmatically, the data archive serves to decouple the rate at which evolution is performed from the rate at which stream content updates. This is particularly important given that the cost of model construction might preclude operation at the rate data passes through the stream. Updates to the data archive content ($DS(i)$, Figure 2) can potentially be decoupled from the rate at which new material enters the sliding window (or $SW(t)$, Figure 2). Moreover, from a streaming data perspective, the data archive can also provide a mechanism for addressing the issue of class-imbalance (skewed data). For example, all classes might be represented in the data archive with equal frequency, whereas there is no guarantee that all classes appear in the stream with equal frequency.

Finally, we note that at all points in time, a model is available for suggesting data labels, but such models can also be used as the basis for potentially requesting label information. This implies feedback loops exist between the population of models, data archive, and interface to the stream. In effect bootstrapping is being performed relative to the certainty with which a model provides label information and / or data sampling is performed. Such an approach is potentially distinct from that typically assumed by ML approaches to streaming data. Specifically, it is generally assumed that the cost of incremental refinement of models using ML is sufficiently low and / or entirely new models are constructed when changes are detected relative to the most recent sample of data from the stream. These topics will be developed further in Section 5.3.1.

3.3.3 Diversity

The convergence property of schema theory explicitly points to the loss of diversity in population based frameworks (e.g., [116]), whereas sufficient diversity needs maintaining in order to provide the basis for reacting to changes [138, 139]. With this in mind we adopt the two categories of diversity maintenance as initially proposed by Dempsey *et al.* [51] and revisit from the explicit perspective of evolutionary model building under dynamic tasks:

Reactive: approaches imply that decreases in fitness are associated with a need to increase the rate of mutation [37]. This naturally implies that some measure of fitness is readily available (possibly implying that the stream is labelled) or that label free change detection is possible (see Section 5.3). Moreover, on detecting a change, evolving the entire population from a completely new initialization might be appropriate [126, 195], whereas in other applications an incremental reseeding of a new population using material from the previous population is recommended [49, 188].

Continuous maintenance: implies that schemes are introduced that attempt to maintain the diversity of the population on a continuous basis. Grefenstette's concept of 'random immigrants' [87] introduces a fixed number of randomly initialized individuals at each generation and as such is sufficiently generic to be applicable to GP, albeit lacking efficiency (most off-

spring are likely to be replaced). More interesting is the concept of age biases. Under genetic algorithms, age biases have demonstrated their utility in dynamic [85] and static (multimodal) environments [93]. Under dynamic environments, an age bias was introduced to prefer the 'middle aged' over the young or old.[7] Conversely, under static environments the role of aging was to ensure a fair competition between similarly 'developed' individuals in combination with the continuous introduction of new offspring through stochastic sampling alone. More recently, aging and fitness sharing metrics have been compared under streaming classification tasks [6, 7]. Both appear to be effective, with a preference for fitness sharing under Markov style environments, whereas a combination of aging and fitness sharing appears to be more effective when the underlying task is subject to gradual variation. However, fitness sharing again assumes that a suitable performance metric is available, e.g., a labelled stream.

The introduction of evolutionary multi-objective (EMO) fitness functions in GP has also resulted the adoption of explicit mechanisms for maintaining diversity [15, 134]. This is particularly relevant in the case of unbalanced data in which solutions take the form of multiple GP individuals. Thus, diversity is useful in ensuring that when constructing ensembles of classifiers, each classifier complements the performance achieved to so far (e.g., [132]). Without this, performance on infrequent classes is potentially penalized in favour of performance on dominant classes. Indeed, recent results indicate that the development of GP models under streaming data identify classifiers for the most frequent classes first and only later add the less frequent classes as the stream progresses [8].

Finally, frameworks for maintaining diversity through the use of multiple populations have been proposed, albeit not specifically with respect to streaming data (e.g., [74]). Thus, different populations have access to different partitions of the dataset, as in ensemble boosting methodologies (Section 3.2). The restricted partitioning and distribution of the data as seen by different populations helps maintain diversity and therefore increases the likelihood of building ensembles from uncorrelated errors [100]. Combining with appropriate pruning heuristics may help maintain the relevancy of independent populations [15], although achieving this under the constraints enforced by the streaming data context is still an open question.

3.4 Discussion

The interrelated nature of evolvability, memory, and diversity points to the evidence of multiple feedback loops that potentially impact the quality of solutions discovered by model based EC (Section 3.3). Moreover, it is apparent from the number of directions presently taken by ensemble based ML (Section 3.2), that there are multiple (possibly redundant) dimensions to the 'design space' associated with streaming algorithm design. This then makes the design of systems for streaming data tasks especially challenging. One potential path for addressing this, albeit not yet considered in the wider literature, is through evolving hyper-heuristics (or the evolution of the evolutionary algorithms), e.g., [150]. Current research has not considered this avenue, so we revisit the opportunity in the context of future research (Section 7).

4 Benchmarking issues

As noted in the introduction, frameworks for streaming data have been dominated by application scenarios associated with classification style tasks. The following discussion of evaluation

[7] Younger / older individuals should only be maintained if they were suitably fit.

methodologies and datasets will reflect this bias.

4.1 Evaluation

Performance metrics employed under the streaming data scenario also face a set of unique challenges [79, 17, 80]. Thus, performance is not a static concept, but relative to the point in the stream at which the evaluation is performed. Hence rather than just being interested in some overall measure of accuracy, we might also be interested in the time taken to adapt to a change. The following shortlist specific scenarios have been reported in the literature:

- Online 'running' average of each exemplar before updating the model [12]. This leads to the concept of prequential error [46], or given a suitable loss function, $L(\cdot, \cdot)$, the model's prediction, $\hat{y}_t$ and the actual label, y_t:

$$p_0 = \sum_{t=1}^{n} L(\hat{y}_t, y_t) \tag{2}$$

  Sensitivity to specific positions within the stream can be further reinforced through estimating over a sliding window (as opposed to the entire stream) or introducing an exponential weighting term (fading factors) [79, 80]. Equivalences have been demonstrated between (batch) Bayesian error estimation and (weighted) prequential error [79].

- Performance of the model classifier is evaluated relative to a 'block' of the future or both future and past exemplars from the stream (e.g., [138, 160]). The goal of such a dual performance metric is to assess the degree of forgetting and / or model specificity that has taken place between concept shifts in the stream. Such metrics are naturally most informative in the case of streams with an abrupt transition between concepts.

- Evaluate against streaming data not seen during model construction (e.g., [169, 107]). For example, training could actually be performed relative to a sample taken from the data stream, with test performed against the entire stream content or that not explicitly sampled [7].

Naturally, the use of accuracy style metrics implies that the drawbacks common to evaluation under static non-streaming scenarios will be apparent [101]. In the case of streaming data as applied to classification tasks, the Kappa statistic is of particular relevance, in which case performance of the proposed algorithm is evaluated relative to that of a suitable naive model of classification:

$$\kappa = \frac{p_0 - p_c}{1 - p_c} \tag{3}$$

where p_0 is the aforementioned prequential error of the proposed model, and p_c is the probability that the naive model makes a correct prediction. As $\kappa \to 1$, the proposed model approaches the ideal, whereas $\kappa \to 0$ implies that the proposed model's predictions increasingly coincide with that of the naive model.

Under classification tasks Bifet *et al.* note that assuming, say, a Naive Bayes classifier or even a simple majority class 'coin toss', would be informative for the case of data conforming to the i.i.d. assumption [21]. However, as noted in the preceding sections, streaming data does

not conform to such an assumption. Indeed, streaming data might well demonstrate periods with a common trend. From a class label perspective this means that 'periods' can appear with sequential exemplars having the same label. Under such conditions a 'no-change' or autocorrelation rule represents a much more effective basis for a naive classifier under streaming data, i.e., given a class label $y_t = c$, assume all following exemplars are class c until there is a change in class label, then update the naive classifier's 'prediction' to reflect the new class. Such a heuristic was observed to perform better than many streaming classification algorithms on the widely used Electricity benchmark dataset [21].

Finally, we present the following comments regarding streaming data evaluation under class imbalance. A prequential error estimator, $P_e(t)$, as estimated across all sequence history (e.g., [79]) has the form:

$$P_e(t) \leftarrow \frac{e(t) + (t-1) \times P_e(t-1)}{t} \tag{4}$$

where t is the sequence index, $P_e(t=0) = 0$, and $e(t)$ is the error on exemplar t.

Naturally, given such a definition, prequential error will weight all classes equally resulting in an accuracy style metric. Unfortunately accuracy style metrics are not particularly informative when the distribution of class labels are imbalanced, whereas 'rate' style metrics might be much more appropriate [101]. For instance, taking the specific case of detection rate, the following definition might be assumed for streaming data contexts [180]:

$$DR_c(t) = \frac{tp_c(t)}{fp_c(t) + tp_c(t)} \tag{5}$$

where $tp_c(t)$ and $fp_c(t)$ are the corresponding true positive and false positive counts w.r.t. class c up to exemplar t in the stream.

Likewise the corresponding overall (average) detection rate takes the form $DR(t) = \frac{1}{C} \sum_{c \in C} DR_c(t)$, i.e., only when all classes are correctly classified (relative to point t in the stream) will $DR(t) = 1$. Thus, a degenerate classifier that labels the entire stream as the majority class has a detection rate of $\frac{1}{C}$. Such a formulation naturally assumes that the number of classes is known a priori, implying that a suitable heuristic would need adopting when additional classes are encountered [180]. In short, the average detection rate, $DR(t)$, can be estimated incrementally through the course of the stream and, unlike the prequential error estimator, reflects a model's sensitivity to class imbalance.

4.2 Datasets

There are three broad categories of dataset employed for benchmarking machine algorithms as applied to streaming tasks: artificial, real-world time varying, and artificially modified real-world datasets. The following discussion summarizes the sources and basic motivations for each case.

4.2.1 Artificial datasets

Purely artificial datasets represent the most widely occurring scenario employed for empirical benchmarking purposes. Naturally, the principle motivation is that specific forms of variation can be introduced to test particular aspects of a ML algorithm. The potential drawback is that

there is no guarantee that real-world datasets should necessarily be limited or place the same emphasis on the properties appearing in the artificial datasets.

Two generic schemes have for the most part been adopted, either change as a gradual drift or as a sudden shift [199]. However, as established in Section 2.2, categorizing the types of change is in itself a multifaceted task, thus artificially generated datasets should reflect more than one aspect. The library provided by Minku *et al.* provides broad coverage in terms of facets included within an artificial dataset [136, 138]. Specifically, in developing a set of datasets for benchmarking different aspects associated with the task of learning under dynamic environments, Minku *et al.* propose a separation of properties that characterize different aspects of *drift generation* [138]:

Isolated properties: distinguish between two micro properties associated with drift generation:

1. **Severity:** that is the amount of change (in magnitude terms), and can be associated with either the dependent or independent variable. Some authors associate concept *drift* with gradual changes and concept *shift* with abrupt changes to the underlying process (e.g., [199]).
2. **Speed:** describes the time taken for replacement of one generating process by another.

Group properties: distinguish between processes used to compose an isolated drift into a sequence or wider pattern of behaviour. Three properties are identified:

1. **Predictability:** is taken to imply that sequences of the generating process are either random or follow a specific pattern.
2. **Frequency:** determines whether a drift / shift is periodic or not.
3. **Recurrence:** establishes whether previously encountered drifts / shifts are possible, and if so whether they are cyclic or not.

Thus, a total of six task types are described (essentially reflecting the different distributions of the process generating the data) and subject to nine different types of drift (three severities and three speeds). In addition, in the case of the planar dataset, labels were corrupted and irrelevant attributes were included. More recently, datasets have been proposed which incorporate loss of classes as well as drift, shift and cyclic behaviours [65, 153].

4.2.2 Real-world, time varying datasets

Various application domains are believed to possess non-stationary or time varying properties and potentially of interest as a benchmark. Naturally, little can be explicitly said regarding the specific non-stationary properties embedded in real-world datasets. However, benchmarking with such data validates both: 1) how appropriate proposed models are under real-world scenarios, as well as, 2) confirming how appropriate the properties explicitly embedded in artificial datasets are for validating model capabilities. This is particularly important, as for example, benchmarking of early ML approaches to batch (as opposed to streaming) classification tasks indicated that simple attribute thresholding was more robust than ML models as evaluated under real-world datasets, resulting in the development of more appropriate ML algorithms and evaluation practices [92].

Table 3: Example real-world classification datasets with non-stationary properties.

Task domain	Num. classes	Repository
Airline delays	2	[99][8]
Churn prediction	2	[183][9],[10],[11]
Credit card default	2	[84, 142][12]
Electricity market	2	[21, 89][13]
Poker hands	10	[10][14]
Spam filtering	2	[13, 69][15]
Weather prediction	2	[65, 147][16]

Table 3 represents a summary of the most frequently used real-world datasets for streaming data analysis. In general, we note that most datasets have a binary outcome, reflecting a fail / not failed characterization of the task outcome. Thus, in the case of the 'Airline delay' task the goal is to predict whether a flight will be delayed or not, whereas under the credit card dataset the goal is to predict a default on paying the minimum balance on the account. Both tasks are 'dynamic' as the likelihood of airline delays reflect weather conditions or dependencies between flights and credit defaulting is a function of the wider economic situation.

We also note that different tasks may or may not benefit from deriving conclusions from data sequencing properties (implicit in the stream) versus the prediction associated each data instance being a self contained property of the current input attributes alone. Thus, in the case of the poker hands dataset, each card is represented as a pair of integers denoting suite and numerical face value. However, dealt cards can appear in any permutation, implying that recognition of a specific hand for say, three of a kind or two pairs would need to be invariant to card location. Indeed, this also raises the issue of whether attributes representing a hand are presented as a vector or as a sequence. Although learning from attribute sequences is more difficult, it can also lead to more general results. For example, as in the case of evolving solutions for the generalized parity task [98].

4.2.3 Artificially modified real-world datasets

Various repositories already exist for benchmarking datasets (e.g., the UCI database [10]), but for the most part consist of datasets with stationary properties.[17] Evaluating streaming algorithms on such datasets has the potential to confirm to what degree performance is impacted by the assumptions used to develop the streaming algorithm (see for example [8, 107]). Con-

[8] `http://kt.ijs.si/elena_ikonomovska/data.html`

[9] `www.fuqua.duke.edu/centers/ccrm/datasets/download.html`

[10] `www.kddcup-orange.com`

[11] `www.sgi.com/tech/mlc/db`

[12] `http://sede.neurotech.com.br:443/PAKDD2009/`

[13] `http://moa.cms.waikato.ac.nz/datasets/`

[14] `http://archive.ics.uci.edu/ml/datasets/Poker+Hand`

[15] `http://www.esi.uem.es/~jmgomez/spam/index.html`

[16] `http://users.rowan.edu/~polikar/research/NSE/`

[17] The majority of datasets employed to date for benchmarking purposes on account of their temporal properties are distributed across multiple repositories (Section 4.2.2).

Table 4: Example of introducing drift into a static dataset. Example adopted from [138].

Overall label distribution	Partition		
	Original	Shift #1	Shift #2
Setosa (33.3%)	1	4	4
Versicolour (33.3%)	2	1	2
Virginica (33.3%)	3	3	3
Dummy class (0%)	4	2	1

versely, the original dataset can be modified to introduce properties associated with concept drift / shift. In so doing, users are again in a position to verify the types of adaptive behaviour that respond to particular instances of concept change.

In one such approach, the original dataset is divided into a series of partitions consisting of an equal number of exemplars, sampled with uniform probability [138]. The first partition reflects the original allocation of labels to exemplars. Thereafter, for each partition, pairs of labels are interchanged potentially resulting in multiple changes between partitions. Table 4 summarizes such a process in the case of the well known 'Iris' dataset [10]. Note that by introducing a 'dummy' class label it is possible to mimic different classes dropping out during a partition. Gao *et al.* take a similar approach to make small benchmark out of a larger dataset [81].

Yang *et al.* adopt a probabilistic framework for describing the probability of transitioning between different classes [199]. This results in a sequence of classes from which corresponding exemplars are then selected. The opportunity then exists to change the transition probabilities over the duration of a sequence.

One final approach is to take a suitably large dataset and order it relative to one specific attribute. Thus, in the case of the frequently employed 'forest cover type' dataset, the elevation attribute has been employed to 'order' the sequence [177]. Unlike the above forms of modification, this results in a gradual drift in concept as opposed to shift. Given that the dataset describes a multi-class task with significant amounts of class imbalance, from a streaming perspective, it is still capable of posing difficulty especially when evaluated against a naive base classifier as discussed in Section 4.1. The same authors also describe the modification of multi-class datasets to binary categorization in which the set(s) of classes from the original dataset associated with an in-class classification are varied over the course of the dataset, i.e., concept shift. Such a practice was applied to datasets that originally represented a database of movie genre[18] and document categorization [120].

5 Progress to date

In the following, developments to model building under streaming data are considered from six different perspectives: stream interfacing, temporal feature construction, label free change detection, semi-supervised learning, learning classifier systems, and class imbalance. The general goal of which is to identify the key themes and highlight remaining unresolved issues. Table 5 provides a 'snap shot' of the association between research themes and approaches pursued to

[18] `http://meka.sourceforge.net/`

date whereas Table 2 (Section 3.2) provides a similar characterization in the case of ensemble methods.

5.1 Stream interfaces

The sliding window represents an initial interface to the streaming data that delimits how much data a model can access at any point in time. Implicitly, constraints are enforced regarding how much historical information is present versus what sample rate to assume, i.e., window length versus sample interval or tap period (Figure 1). Two scenarios are recognized: windows of a fixed prior parameterization versus sliding windows with an evolved parameterization (Section 5.1.1). A second related issue takes the form of archiving specific data instances for use beyond the duration of the sliding window (Section 5.1.2). In essence, this addresses questions regarding what to learn from. For maximum reactiveness, models might only be evolved against the content of the current sliding window location. However, most of the content of a sliding window location might be highly correlated with the previous, potentially implying that there is not a lot of new material to learn from. Introducing a sampling algorithm between sliding window and a subset set of data against which evolutionary model building is performed addresses this issue and leads to mechanisms for label decoupling (discussed in Section 5.3.1).

Finally, we note that some researchers adopt a stream interface that is limited to the current exemplar, $\vec{x}(t)$, alone. Models applied under such a limitation assume that labels are available for all the stream [84, 107]. Such models are purely pattern recognizers as opposed to incorporating spatio-temporal information. Moreover, a greater emphasis is placed on reacting to label information, as opposed to say detecting change and then reacting. Exceptions to this would be a model that is capable of recording internal state (e.g., neural representations with recurrent connectivity or GP with indexable memory) or ensemble approaches in which each different model responds to different probability distributions. In this latter case an ensemble could potentially capture the transition between different probability distributions (i.e., a spatio-temporal relation).

5.1.1 Sliding windows

Sliding windows assume a first-in, first-out style of operation in which the oldest samples are shifted out as new samples are recorded. Parameters define the number of samples retained and interval between samples (also referred to as tap or skip length, Figure 1).[19]

Fixed window parameterization and attribute selection: The parameterization of a sliding window defines the length of the window and the interval between the consecutive samples (sampling interval). Any such characterization is task specific. Moreover, there is a tradeoff between longer and shorter windows (and corresponding decreased / increased sample resolution) [9, 45]. Thus, a short high resolution window and a longer low resolution window are often applied together. Indeed, GP as applied to multi-step prediction has been reported with multiple window resolutions [31] as do streaming approaches to decision tree induction, see Section 5.3.1.

Variable window lengths: if the underlying process is non-stationary, then assuming a window size optimized relative to some historical 'training partition' will at some point result in

[19] In addition non-overlapping windows have been used, in particular with ensemble methods, with different members of the ensemble being constructed with each new ensemble location (see Section 3.2).

Table 5: Characterization of research themes and approaches introduced in the course of this survey. Broad distinctions are made in terms of evolutionary versus non-evolutionary. Some theme specific or GA / GP / LCS distinctions are also employed where appropriate. Section referencing is provided in order to facilitate identification of the appropriate commentary. See Table 2 for a characterization of research themes in (non-evolutionary) ML ensembles.

Theme	Approach
Evolvability / Modularity (3.3.1)	GA: [173, 174, 187, 191, 198] GP: [14, 61, 63, 148, 192, 195] Environment: [30, 36, 67, 103, 151]
Selection / replacement (3.3.1, 3.3.3)	GA: [85, 87, 93, 121, 141] GP: [7, 8, 94, 95, 179, 182, 188]
Memory (3.3.2)	GA: [121, 174, 198] GP: [63, 188]
Distance metrics (3.3.2)	GA: [22, 140, 198] GP: [28, 64]
Sliding window (5.1.1, 5.2)	Evolve: [29, 49, 52, 126, 188] Non-evolve: [9, 18, 19, 45]
Feature construction (2.4, 5.2, 5.3.1)	Evolve: [3, 71, 126, 144, 161] Non-evolve: [171, 189, 196]
Class imbalance (3.3.3, 5.6)	Evolve: [7, 8, 157, 179] Non-evolve: [84, 143]
Active learning (5.1.2)	Evolve: [8, 7, 179] Non-evolve: [86, 97, 128, 176, 177, 203]
Change detection (5.3)	Input: [4, 19, 44, 53, 73, 105, 162, 171, 185] Model (GP): [126, 157] Model (non-GP): [12, 38, 78, 118, 119, 124, 125, 145, 175, 176, 177, 203]
Label budgets (5.3)	Evolve: [13, 179] Non-evolve: [122, 124, 125, 177]
Semi-supervised (5.4)	Evolve (non-streaming): [5, 41] Non-evolve: [54, 60]
Prototype based (5.5)	LCS: [1, 13, 32, 43] Non-evolve: [181, 197]

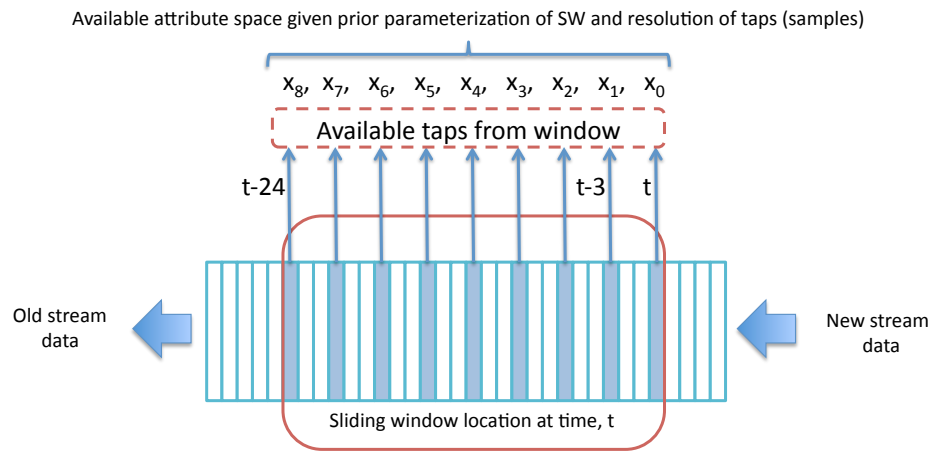

Figure 1: Generic sliding window (SW). In this example a SW parameterization is assumed in which a window length of 9 samples is employed, spanning a historical interval of 24 consecutive records, sampled at a resolution of every 3rd instance. The following evolutionary model builder is limited to the set of 9 samples $x_i; i \in \{0, ..., 8\}$. Each sample is potentially a vector of attributes, e.g., sensor values. The resulting model need not explicitly index all samples / sensor values.

a deterioration in performance. The continuous re-parameterization of such windows therefore becomes important. However, it might be useful to retain the capability to make use of previously evolved windows and / or models from the perspective of getting a 'head-start' on the construction of a new model. The DynFor algorithm [188] begins with a pair of candidate sliding window lengths n and $n+1$ and selects a solution window length relative to the current training partition. The process repeats with new sliding window lengths chosen relative to the 'direction' of improvement. Moreover, the authors note that for a stationary process the window size is likely to undergo incremental increases, whereas during a transition between different underlying processes, the window size is likely to decrease [188]. The process although capable of tuning the size of the sliding window assumes that the stream is appropriately labelled to facilitate continuous evolution. That said, similar assumptions are also central to the ADWIN algorithm employed with online decision tree induction (e.g., [19]). However, as will become apparent within the context of label free change detection (Section 5.3), various frameworks have recently been proposed to address online learning under limited label budgets that are compatible with broad classes of ML and EC.

5.1.2 Sampling

Following from the immediate interface to the stream (i.e., some form of sliding window) the machine learning algorithm might act directly on the sliding window content alone, or more generally, a decision is made regarding what to retain from the immediate content of the current sliding window location, $SW(t)$, for retention within a finite size sample (Figure 2). Such schemes have been widely deployed under classical stationary offline learning algorithms in order to decouple the original training partition cardinality from that of the training 'sample', i.e., an EC generation is only performed relative to the content of the data subset, $DS(i)$ (Figure 2). Active learning represents one of the most widespread examples of such a methodology in which performance of the model is fed back to bias the selection of exemplars as used for the

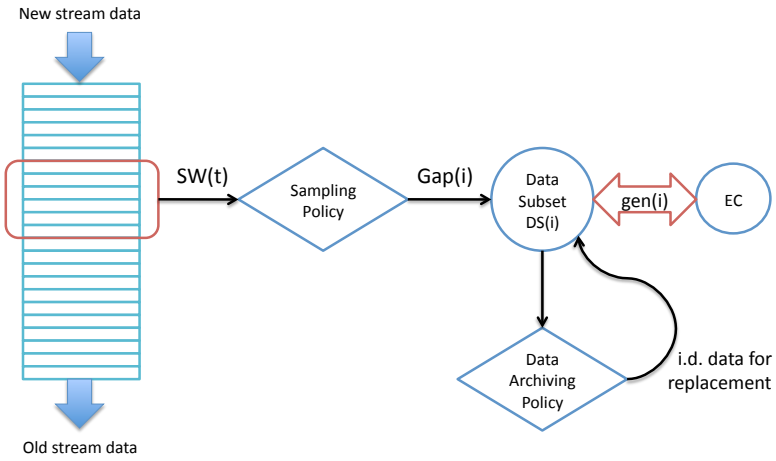

Figure 2: Relationship between sliding window (SW) and data subset (DS) identified through active learning. $SW(t)$ location of sliding window at shift location t. *Sampling policy* defines how a Gap sample of data from $SW(t)$ is identified, where $Gap \in \{SW(t)\}$ and $|Gap| \leq |SW|$ and $|Gap| \leq |DS|$. Update rates for the stream and Gap need not be the same or $freq(t) \neq freq(i)$. $DS(i)$ is the data subset against which generation, i, of EC is performed. *Data Archiving policy* defines how $|DS| - |Gap|$ samples are identified for replacement from the Data subset before the next EC generation is performed. Illustration adopted from [6, 179].

next round of model construction [39].

Under the context of active learning as applied to streaming data, we note that two sets of policies are necessary: a *Sampling policy* or how much to sample from the current sliding window location; and an *Archiving policy* or what data currently in the data subset is replaced between EC generations (Figure 2). With this in mind, there are two parameters that are significant to evolution under streaming data: 1) rate of data subset updates, $freq(i)$, versus rate of sliding window shift, $freq(t)$; and 2) the amount of data transferred between the sliding window and the data sample, Gap. The interaction between the proportion of the sliding window content transferred, Gap, and the frequency of sliding window shifts $freq(t)$ defines the labelling requirement. As long as $freq(i)$ and / or $|Gap|$ are less than the corresponding sliding window parameters ($freq(t)$ and $|SW|$) then there is a reduction in the proportion of the stream requiring labels. Naturally, credit assignment is only performed at the frequency of subset updates, $freq(i)$.

Probabilistic sampling: represents one of the earliest schemes for introducing biases into the selection of exemplars for decoupling fitness evaluation from the cardinality of the data. Gathercole and Ross compared pure random sampling to a scheme in which both exemplar difficulty and exemplar age biased the selection of exemplars to the subset of exemplars actually employed for fitness evaluation [83]. Extensions included hierarchical 'cache friendly' frameworks [165] and their use for supporting model building through stacked generalization [42]. Other biases frequently employed include those for representing each class equally – where this is correlated with maximizing the AUC, as opposed to sampling exemplars with equal probability which is correlated with maximizing the accuracy metric [193]. Naturally, streaming data implies that no revisiting can take place. Thus, probabilistic methods as discussed here are limited to constructing statistics from the current content of the sliding window. How many of the previously mentioned results, which are known to be sufficient for the case of static datasets, carry over to the case of streaming data, is for the most part unknown.

Pareto archiving: provides a formal scheme for characterizing which subset of exemplars to retain through the concepts of distinctions and Pareto dominance. In particular, the goal of (archived) exemplars is to distinguish between different models, and the non-dominated models in particular [47, 72, 146]. One potential drawback of the approach is that the exemplar archives might grow considerably. However, enforcing a finite archive through the introduction of diversity measures (such as fitness sharing) has been empirically shown to be effective for efficiently decoupling fitness evaluation from training partition cardinality under batch (offline) learning [133, 134]. The extension to sliding window interfaces and single pass constraints for online learning indicate that it is possible to match the performance of multi-pass batch algorithms when there is no labelling error [8]. Moreover, it may also be appropriate to tune the heuristic used to maintain a finite archive size depending on whether the stream is subject to sudden changes or continuous gradual changes [6, 7].

Scaffolding: is a concept from psychology used to express the role of tutoring during development which has been widely used in the context of incremental evolution in robotics (e.g., [204]). Elwell and Polikar cite the scaffolding concept as the motivation for the approach taken to selecting 'batches' of streaming data to re-evaluate and potentially modify their neural network based ensemble algorithm [65]. In particular, they relate scaffolding to drift detection and redundancy prevention.

5.2 Evolved temporal features

Sliding (or non-overlapping) windows represent the initial mechanism for delimiting how much data to present an evolutionary model building framework. The next obvious question is how to explicitly capture temporal properties from the window. We identify to generic themes:

1. Apply an a priori set of features – Candidates might include wavelet and Fourier coefficients[20] versus assuming an indexing pattern relative to the sliding window definition. In either case, the use of evolutionary methods capable of attribute identification (as in, say, GP) could result in the discovery of further application specific refinements. Thus, given a pre-specified window length and sampling rate, GP is capable of refining this further to determine which subset of taps to utilize (e.g., [165]). In addition, application domains might well have built up a set of primitives for capturing temporal properties deemed appropriate for a specific task, for example, as in the role of technical indicators in financial data analysis (price channel breakout, relative strength indexes etc.). An evolutionary model builder would then identify what combination of such primitives to utilize in the model (e.g., [52]).

2. Let the evolutionary method design temporal features directly – In this case a mathematical expression is designed as a function of time and attributes. Specific examples include differential, rational or integral transforms (e.g., [3]), primitives based on some form of moving average,[21] trigonometric functions, higher order statistics (e.g., [161]) or polynomials [144]. Implicit in this approach is the adoption of an axiomatic approach to (temporal) feature construction where prior knowledge regarding the task domain is used to identify a suitable kernel for constructing future temporal features.

[20] Denoting how much of a specific spatio-temporal basis function are present.

[21] For example, as in parameterizing specific technical indicators for feature construction in finance [71].

In both cases, the resulting temporal features are evolved relative to the prior parameterization of the sliding window (only data within the sliding window contributes to the outcome from temporal features). Naturally, adopting the first approach implies that model building can take place immediately, whereas temporal feature construction would then require a second independent process to identify the classification or regression model. Several authors have proposed schemes for feature construction under a batch training context (e.g., [11, 110]), whereas temporal feature construction is synonymous with the design of technical indicators (TI) under financial applications. As such, evolutionary methods have been proposed for the construction of TI with prior decision trees[22] [29], as well as the coevolution of both TI and partnering decision tree [126].

5.3 Label free change detection

Change or drift detection implies that shifts in the underlying process driving stream content should ideally be detected without recourse to labels. The 'change detection' and 'label budget' rows of Table 5 provides a summary of algorithms that attempt to address this issue. In the following we distinguish between two specific scenarios. Section 5.3.1 considers the case of universal statistical measures used to 're-trigger' model building. Section 5.3.2 introduces scenarios that make use of secondary 'behavioural' properties that, depending on the application, may be available to re-trigger model construction at appropriate points in the stream.

5.3.1 Universal measures of change detection

Klinkenberg and Renz previously proposed the following three properties by which a statistical characterization of change can trigger model reconstruction. In all cases, general statistics are collected and compared using statistical tests [106]:

Properties of the classification model: implies that measurements are made of the behaviour of the classifier as it operates on the stream, and a statistical comparison made against a reference behaviour. Such an approach has been widely used for characterizing the behaviour of leaf nodes in decision trees. Thus, frequency statistics or expected loss metrics have both been proposed [68, 97]. Under fuzzy / neural models of classification, metrics for conflict and ignorance with respect to a model's antecedent decision boundaries have been proposed as a mechanism for triggering requests for label information [128].

Properties in the input data or $p(x)$: implies that a statistical characterization is made of a reference (sliding) window and comparisons made with the current sliding window content using various statistics such as: entropy [4, 44, 185], Chernoff bounds [105], Kullback–Leibler divergence [162], Hoeffding bounds [19], Fractal correlation dimension [73] or Hellinger divergence metric [53]. A significant drawback of such methods however, can be a need for class labels. Finding schemes for decoupling from the labelling requirement represents an on going line of research (e.g., [73]). Moreover, input based statistics can be sensitive to the dimensionality of the input space, with various aspects of the curse of dimensionality reducing the effectiveness of distance metrics as the number of attributes increases [111, 196]. One approach for addressing this is to assume increasingly application specific solutions. Thus, thresholding word frequencies represented an early approach proposed for the specific case of change detection under text mining applications [171].

[22] The decision tree defines the condition under which an action is applied, say, as in sell, buy or hold.

Properties in the label space or $P(y|\vec{x})$: attempt to characterize variation in the classifier output. Assuming that labels are available at some fraction of the stream throughput, then requesting labels randomly (i.e., independently from any other source of information) has been shown to be surprisingly effective [203]. Attempts to make more explicit use of the classifier properties include using information regarding class boundaries to prioritize the request for labels [124, 176] or information about variation in classifiers from an ensemble as a proxy for change detection [203]. Specifically, statistical tests have been used to make use of classifier confidence. Thus, changes are detected when either classifier certainty drops below some prior threshold [125] or the number of confident predictions undergoes a significant change [118]. One of the additional favourable properties of this approach to change detection is that they are potentially applicable to a wide range of machine learning algorithms, including GP [157].

Naturally, if a previously encountered input instance, $\vec{x}$, is associated with more than one class label during the course of a stream, then additional measures are necessary. A recent work by Žliobaitė *et al.* considered this case from the perspective of a limited budget active learning framework [177]. Specifically, given a finite labelling budget, the authors show that a combination of stochastic sampling and initiating classifier 'boundary biased' sampling provide an effective means for detecting unique shifts in $P(y|\vec{x})$.

5.3.2 Task specific change detection

Certain task domains, such as decision making under (currency or stock) market trading environments, do not need to rely on labels to characterize performance. That is to say, although the decision maker being evolved provides one of a finite set of actions (sell, buy, hold), there is no capacity for providing an 'ideal' sell–buy–stay decision at each time step. Instead, performance is characterized in terms of alternative behavioural properties, such as the amount of acceptable loss (per time step) or number of consecutive losses. A change is detected when the policy of the decision maker steps outside of the predefined performance criteria [126, 127]. An open research question is as to whether other research domains can be approached in a similar way (e.g., see [183]) or for that matter through reinforcement learning style formulations as used in learning classifier systems (Section 5.5).

5.4 Semi-supervised learning

Semi-supervised learning algorithms attempt to make use of unlabelled data as well as labelled data during model construction (e.g., [33]) and as such have an intuitive relevance to online streaming data. Two basic formulations exist: inductive and transductive. The typical starting point is a *batch* of labelled data, $(x(i), y(i)); i \in \{0, ..., t\}$, and a *batch* of unlabelled data, $x(j); j \in \{t+1, ..., l\}$. Inductive approaches attempt to use both labelled and unlabelled data to build a model with better predictions for $x(k); k > l$, whereas transductive approaches use both batches of data to improve performance on the unlabelled interval $t < j \leq l$. From the perspective of streaming data analysis, Zhang *et al.* benchmark a semi-supervised SVM with data including concept drift [201], and later describe an approach for updating ensembles using semi-supervised learning [202]. Few works attempt an explicitly evolutionary model based approach to semi-supervised learning, and then do so under stationary tasks [5].

More recently, an extreme case is described where, following an initial batch of data with labels, the stream provides no further label information. Thus, feedback during the stream is limited to the input variables, $\vec{x}$, alone [60]. The approach is based on constructing classi-

fiers using the initial label information such that class conditional distributions are identified to define a tight envelope around data for each class, or the 'core' support. Such regions are then used as the basis for modelling (the stream) and thereafter updated without label information. In this regard, the approach is an example of classification through novelty detection as opposed to discrimination, where this is widely used in one-class classification [130, 131] or under GP [41, 134]. Moreover, [60] explicitly emphasize the significance of exemplars contributing to updating the core of class distributions as opposed to the boundary conditions; the latter representing the typical emphasis of both active learning and the 'passive–aggressive' classes of algorithms for online classification [40]. One limitation of the approach is that it is presently specific to drift as opposed to shift.

5.5 Learning Classifier Systems and Prototype methods

Learning classifier systems (LCS) in general use a GA to manage a population of rules (antecedent–consequent pairs) and reinforcement learning to guide credit assignment, with the latter typically associated with accuracy-based LCS or XCS [109]. Given a subset of rules with antecedents matching the current environmental condition, the 'winning' action can be selected deterministically (exploitation) or stochastically (exploration), with the latter used to promote non-greedy evaluation of state–action pairs.

Dam *et al*. use a real-valued variant of the multiplexer benchmark to introduce a time varying property into the task [43]. Specifically, the threshold determining when an input is considered a 'zero' or 'one' is controlled by a time varying process. Various experiments are performed with different step changes to the threshold, with and without noise. The standard formulation for XCS is found to have a recovery time proportional to the magnitude of the change in threshold. Moreover, a distinct 'sweet spot' exists with the amount of noise that XCS is able to operate effectively under. Dam *et al*. take this as an indication that XCS is unable to reuse individuals from a population to accelerate evolution during a change in the underlying concept. Instead evolution from scratch is more effective. Several recommendations are then made:

- Adapt the learning rate such that it decreases under 'noisy' environments and increases under concept shift.
- Concept shift is equated with increases in error over a window of consecutive exemplars (implying that the stream is labelled). Specifically, thresholds for error and a minimum number of covering rules are employed to recognize a concept shift.
- On detecting a concept shift various learning parameters are reinitialized versus reinitializing the entire population.

Under streams with concept shift alone, the case of reinitializing population content is the most effective. However, the ability to trigger population re-initialization is a function of prior knowledge regarding what constitutes a 'good' threshold.

Behdad and French note that under batch learning scenarios, LCS are first deployed under a purely exploratory setting and then under a purely exploitative setting [13]. Such a separation might not be appropriate in the case of online learning. Instead, the online setting it taken to require that XCS first classify (exploit) and then receive feedback (explore). Some balance needs maintaining in the ratio of explore to exploit cycles. Behdad and French assume that

more exploit (classification) cycles are performed over explore cycles (stochastic updates) in much the same way that classical algorithms for reinforcement learning mix greedy exploitation (say Sarsa or Q-learning) with ε-greedy exploration [170]. The 'delay' concept appears when a batch model of updating is reintroduced. Specifically, a delay of τ is taken to represent the number of exemplars per 'batch' of data. However, unlike the regular model for applying XCS to classification tasks, first XCS is deployed to label the data (exploitation) and then in an independent pass through the *same* data, the exploration phase is performed. Finally, the case of 'partial' feedback is considered. That is to say, in the case of some exemplars there might be no feedback, limiting exploration to only those cases with the feedback. The authors formulate probabilistic heuristics in an attempt to 'fill in' the gaps so that feedback under no label confirmation is possible [13].

Prototype approaches are frequently employed in clustering algorithms for characterizing cluster location (i.e., medoid as opposed to centroid style clustering) and have been considered for online learning under streaming data scenarios (e.g., [181]). Indeed, learning vector quantization (LVQ) represents a family of algorithms that add label based credit assignment to turn Kohonen's Self Organizing Feature Map into a classifier. Moreover, LVQ has itself been adapted to the case of online learning [197]. Cervantes *et al.* propose a framework for managing a set of prototypes incrementally using concepts from particle swarm optimization [32]. Thus, momentum is added to the clusters representing classes and therefore non-stationary properties in the stream can potentially be tracked. In addition, clusters with the same class label are grouped using mechanisms from online LVQ [197]. The approach taken to establishing groups is based on link maintenance and LVQ neighbourhoods. Links are subject to aging and therefore decay if a prototype no longer classifies data. A total of 8 parameters are necessary to characterize coefficients for aging, memory, inertia etc., with coefficient selection having an impact on the memory requirements and degree of adaptability of the algorithm.

A less widely investigated approach is to assume that models can be constructed a priori for a set of concept drifts, where this implies that it should be much easier to react to changes with a minimal labelling requirement during the course of the stream [164]. Naturally, various assumptions need to be made regarding the type of concept drift likely to appear, however, the payoff is that the cost of training *during* the stream can be significantly reduced relative to models that have no access to appropriate prior training. At some level this can also be interpreted in terms of models designed to react to repeating or cyclic properties.

5.6 Class imbalance

Building classification models under class imbalance or skewed data is a relatively mature – although by no means 'solved' – topic in the machine learning literature (e.g., [91]) and GP [11, 15, 134]. In part, this interest is driven by the observation that most real-world datasets are not 'balanced', a tendency that increases as multi-class classification is encountered.[23] In general, there are three approaches pursued for addressing the class imbalance problem, albeit with the assumption that the data is stationary:

- Perform model identification over some sample from the training partition. The scheme pursued for sampling might include feedback from the model during training (e.g., active

[23] Attempts to cast a multi-class classification problem into at least $C - 1$ binary classification problems merely emphasizes this effect. Thus, even if the C classes appear with equal frequency, each binary classification task represents an unequal partition of one class versus the rest.

learning [83, 165]) or enforce some prior heuristic for under / over sampling of specific classes. From the perspective of evolutionary computation this represents a test case for learning what to learn from, or competitive coevolution (e.g., [47]). Under the specific context of regression tasks, schemes have been proposed in which data uniqueness is first used to recursively eliminate all but the most 'meaningful' exemplars from the training partition after which model building is pursued [184].

- Introduce penalties into the cost function. The basic assumption here is that exemplars are weighted differently depending on their class. Specific examples include reformulations of the fitness function to reflect prior knowledge of class frequency or cost (e.g., [16, 115]), performance on exemplars (e.g., [62]) or Pareto multi-objective formulations (e.g., [15]).

- Combined approaches in which both schemes are pursued together. Such hybrid schemes have the added advantage of decoupling the cost of fitness evaluation from the cardinality of the original training partition [58, 123, 134].

In contrast, comparatively little appears with respect to class imbalance under streaming data. Processing data in batches (e.g., non-overlapping sliding windows) provides one avenue for addressing class imbalance under streaming data scenarios. That is to say, the distribution of classes represented in the batch can be artificially balanced with less frequently occurring classes relying on historical samples, whereas the most frequently occurring classes assume the most recent samples (e.g., [82]). Various schemes have been proposed for prioritizing retention of minor class exemplars within the batch used to construct classifiers, with k-NN algorithms frequently appearing for this purpose [34, 86]. Conversely, Ditzler and Polikar propose to employ oversampling or data rebalancing with ensemble methods, but not without incurring computational overheads that might limit the applicability to streaming data [55]. However, one implication of holding on to minority class data for longer periods of time (relative to the major class(es)) is that minor class(es) will increasingly be characterized as stationary [55]. This is an example of a sample bias issue to which any form of resampling can potentially lead to models with unforeseen classification properties.[24]

Attempting to address class imbalance under strictly 'online' conditions is potentially more challenging. One approach proposed assumes that labels are freely available. Thus, if all the streaming data can be labelled at no cost, then approaches might be assumed in which the model is made as reactive as possible. Class imbalance is addressed by applying different costs for each class, either under an a priori fixed cost or by adapting the costs online [84]. An alternative approach is to assume that the stream is sampled. The default being a fixed sample rate [143].

Various schemes have been proposed that attempt to maintain estimates for the frequency of different classes while conforming to a strictly online model of deployment [190]. To date, the principle drawback is that label information is still necessary, implying that only through subsampling can there be any reduction in the rate at which labels are produced. Possible schemes for avoiding this require on some form of label free change detection (see Section 5.3), i.e., a sampling bias associated with data mapped to the decision boundary in the less frequent class(es).

[24]For a general discussion of this topic (albeit under a non-streaming scenario) see for example Chapter 9 from [59].

6 Conclusion

The field of streaming data analysis has been reviewed from a broad perspective to identify several current key themes which are equally applicable to EC model building:

- Ensemble methods have been widely utilized to address issues of incremental model refinement versus generating entirely new models. From the perspective of evolutionary computation there are potentially several lessons that can be learnt from ensemble methods regarding diversity maintenance (synonymous with population diversity) and ensemble composition (synonymous with teaming in GP).
- A wide range of benchmarking datasets and evaluation metrics are already available providing a large repository of previous results and established methodologies for evaluation.
- Several generic schemes have been proposed for decoupling the labelling requirement and / or change detection under streaming data. Such algorithms are equally applicable to evolutionary methods as they are to more classical forms of ML.

Properties of model based evolutionary paradigms that potentially make them appropriate to streaming data tasks have been reviewed from the perspective of evolvability, diversity maintenance and memory mechanisms. Differences between approaches originally promoted under genetic algorithms versus their utility under GP were highlighted. With respect to challenges we anticipate in the near future we highlight the following:

- *Model building under an open versus closed world assumption:* Supervised learning as classically deployed makes a closed world assumption, i.e., the training partition is sufficient to provide a complete description of the underlying task. Conversely, learning under a non-stationary stream implies that what can be inferred from any part of the stream is incomplete. Distinctions need to be made between the known (what the model is explicitly capable of) and the unknown (what explicitly lies outside the capability of a current model). This represents a general requirement for novelty detection as opposed to discrimination [130, 131]. To date, there have been a few GP frameworks proposed that operate under a closed world assumption – for example, by making use of a Gaussian rather than a Sigmoid style membership function [41, 132, 134, 200]. However, little is known about their operation under streaming data conditions.
- *Local versus global search:* Concept drift is characterized by gradual variation in the underlying structure pertaining to a model. Many classical ML approaches naturally assume a greedy credit assignment scheme, thus can readily lend themselves to tracking gradual changes associated with concept drift. Equivalent formulations for model based evolutionary methods might take the form of specific types of variation operator, applied relative to a previously identified genome. Deciding when to employ such operators might be motivated by frameworks proposed by ensemble methods or change detection.
- *Forecasting / regression tasks under streaming data:* To date, research in symbolic regression has tended to concentrate on improving model accuracy and trustworthiness. The latter particularly with respect to regression models as applied to system identification, in which case the problem might be too little data as opposed to too much (e.g., [184]). Moreover, there is little emphasis placed on online operation with a labelling budget. That

said, interesting datasets certainly exist that might provide the basis for the development of appropriate regression tasks for benchmarking. For example, forecasting benchmarks are often based on chaotic attractors (e.g., [158]) thus uncertainty regarding concept shift / drift. Likewise, some of the most challenging regression tasks include a 'switch' between different model generating processes and dummy attributes [108]. Indeed, similar challenges exist in related fields, such as regime switching in econometrics (e.g., [117]). To date, however, there has not been much emphasis placed on performance under reduced labelling budgets, where this remains a key requirement for streaming data scenarios.

- *Evolvability versus diversity:* Historically a significant effort has been placed on the role of diversity maintenance (Section 3.3.3). However, more recently the type of diversity has been questioned from both the perspective of EC (e.g., [191]) and ensemble learning (e.g., [26]). As a consequence, diversity maintenance is increasingly being seen as a byproduct of other underlying properties associated with evolvability (Section 3.3.1). We anticipate that mechanisms for supporting evolvability and, possibly more importantly, the identification of appropriate feedback loops between environment and evolvability to continue to be a significant challenge for future research.

Finally, we characterize the differences in approach between model building using EC versus ML in general as follows:

- *Level of prescription:* Ensemble based ML currently requires specific a priori decisions to be made regarding the types of ensembles to be maintained, or whether to drop ML learners after change detection. Conversely, EC frameworks have the potential to evolve either scenario from the same population by adopting suitable schemes for diversity / evolvability. On the one hand this leaves the ability to discover the relevant transition between models. On the other, there is potentially less certainty that such a discovery will be made.

- *Computational:* EC model building requires multiple passes through the training partition; a requirement that is potentially in conflict with the open-ended / non-stationary component of the stream itself. ML more often than not assumes some form of greedy credit assignment, making them more 'naturally' online / reactive (although achieving this under label budgets is still an open question). One approach for addressing this issue is by assuming some form of active learning from the outset (e.g., the data subset of Figure 2). In addition, the cost of model construction through EC itself can often be reduced by assuming representations that are less costly to evolve. For example, dropping support for double precision arithmetic. One framework adopting such an approach conducts 1,000 generations on the last 1,000 exemplars from the stream with a population size of 100 individuals in as little as 5 seconds on a regular desktop computer [127]. Depending on the type of stream, this might in itself be sufficient for real-time operation.

- *Type of design decision:* We maintain that many design decisions are not specific to adopting EC versus ML approaches to model building. Thus, decisions regarding the stream interface, detecting change or operating under label budgets are potentially appropriate for model building under either EC or ML. However, an ML practitioner might consider having to make decisions pertaining to population diversity, evolvability and memory as not providing sufficient control over the properties of the resulting model. This naturally relates to the 'level of prescription' in framework design as discussed above.

7 Future Research

In the following we motivate a longer term research agenda[25] through two themes: evolutionary hyper heuristics and local versus global search:

- *Evolutionary hyper heuristics:* GP as heuristic / algorithm generator implies that GP operates on a search space of algorithm components [150]. From a streaming data perspective the opportunity then exists for continuous adaptation of the functional components (e.g., elements of and / or relation between the modules in Figure 2). Potentially, GP as an algorithm generator would be able to tune between a spectrum of streaming algorithms, say semi-supervised to / from active learning – Dyer *et al*. note that active learning and semi-supervised learning are quite close algorithmically [60] – or compose ensembles of learners. Indeed, the wide range of approaches to ensemble learning point to the "design space" of algorithms for streaming data being both rich and most likely being open to meta-heuristic search processes. Moreover, navigating the redundancies that appear to exist in the multiple combinations of ensemble solutions potentially points to automated algorithm design providing a more effective scheme for rationalizing what properties of an ensemble are most effective for addressing specific features of the streaming data task. The principle challenge, of course, is managing to do so in a computationally feasible manner.

- *Local versus global search:* Frameworks in which GP is used to find 'projections' from the original attribute space to a new 'feature space' are beginning to appear [11, 104]. The benefit of adopting such an approach from a streaming data perspective is that the behaviour of the stream can be analyzed in the new feature space and we are free to design properties useful for stream analysis into the feature space. Moreover, whereas construction of the mapping between attribute and feature spaces might be non-linear, the mapping from feature space to label space could be pursued using linear models, e.g., Naive Bayes or Linear Discriminant Functions. Within the context of streaming data, it has recently been proposed that the task of 'adaptive preprocessing' the stream data is significantly more difficult than that of constructing a streaming classifier [178]. We see the adaptive preprocessing step as being synonymous with mapping from the original attribute space to that of a new more convenient feature space and a natural role for model based EC. Conversely, any number of (greedy) ML algorithms for streaming might be applicable to mapping from feature to label space. Naturally, the update rates for each stage need not be the same.

Acknowledgments

The author would like to thank the reviewers for their constructive feedback on this article resulting in significant improvements on earlier drafts. Support through the NSERC CRD grant program and RUAG Schweiz AG is readily acknowledged.

[25] Potential short term research goals having been noted in the conclusion (Section 6).

References

[1] H. A. Abbass, J. Bacardit, M. V. Butz, and X. Llora. Online adaptation in learning classifier systems: Stream data mining. Technical Report IlliGAL Report No. 2004031, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, 2004.

[2] H. Abdulsalam, D. B. Skillicorn, and P. Martin. Classification using streaming random forests. *IEEE Transactions on Knowledge and Data Engineering*, 23(1):22–36, 2012.

[3] A. Agapitos, M. Dyson, J. Kovalchuk, and S. M. Lucus. On the genetic programming of time-series predictors for supply chain management. In *ACM Genetic and Evolutionary Computation Conference*, pages 1163–1160, 2008.

[4] C. Alippi, G. Boracchi, and M. Roveri. Just-in-time classifiers for recurrent concepts. *IEEE Transactions on Neural Networks and Learning Systems*, 24(4):620–634, 2013.

[5] F. L. Arcanjo, G. L. Pappa, P. V. Bicalho, W. Meira, and A. S. de Silva. Semi-supervised genetic programming for classification. In *ACM Genetic and Evolutionary Computation Conference*, pages 1259–1266, 2011.

[6] A. Atwater. Towards coevolutionary genetic programming with Pareto archiving under streaming data. Master's thesis, Faculty of Computer Science, 2013.

[7] A. Atwater and M. I. Heywood. Benchmarking Pareto archiving heuristics in the presence of concept drift: Diversity versus age. In *ACM Genetic and Evolutionary Computation Conference*, pages 885–892, 2013.

[8] A. Atwater, M. I. Heywood, and A. N. Zincir-Heywood. GP under streaming data constraints: A case for Pareto archiving? In *ACM Genetic and Evolutionary Computation Conference*, pages 703–710, 2012.

[9] B. Babcock, M. Datar, and R. Motwani. Sampling from a moving window over streaming data. In *ACM–SIAM Symposium on Discrete Algorithms*, pages 633–634, 2002.

[10] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[11] K. Badran and P. Rockett. Multi-class pattern classification using single, multi-dimensional feature-space feature extraction evolved by multi-objective genetic programming and its application to network intrusion detection. *Genetic Programming and Evolvable Machines*, 13(1):33–63, 2012.

[12] M. Baena-Garcìa, J. Del Campo-Àvila, R. Fidalgo, and A. Bifet. Early drift detection method. In *ECML PKDD International Workshop on Knowledge Discovery from Data Streams*, pages 77–86, 2006.

[13] M. Behdad and T. French. Online learning classifiers in dynamic environments with incomplete feedback. In *IEEE Congress on Evolutionary Computation*, pages 1786–1793, 2013.

[14] T. V. Belle and D. H. Ackley. Code factoring and the evolution of evolvability. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1383–1390. Morgan Kaufmann, 2002.

[15] U. Bhowan, M. Johnson, M. Zhang, and X. Yao. Evolving diverse ensembles using genetic programming for classification with unbalanced data. *IEEE Transactions on Evolutionary Computation*, 17(3):368–386, 2013.

[16] U. Bhowan, M. Zhang, and M. Johnson. Genetic programming for classification with unbalanced data. In *European Conference on Genetic Programming*, volume 6021 of *LNCS*, pages 1–12, 2010.

[17] 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.

[18] A. Bifet, E. Frank, G. Holmes, and B. Pfahringer. Accurate ensembles for data streams: Combining restricted hoeffding trees using stacking. In *Proceedings of the Asian Conference on Machine Learning*, pages 1–16, 2010.

[19] A. Bifet and R. Gavalda. Learning from time-changing data with adaptive windowing. In *SIAM International Conference on Data Mining*, pages 443–448, 2007.

[20] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà. New ensemble methods for evolving data streams. In *ACM International Conference on Knowledge Discovery and Data Engineering*, pages 139–148, 2009.

[21] 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.

[22] T. Blackwell and J. Branke. Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE Transactions on Evolutionary Computation*, 10(4):459–472, 2006.

[23] D. Brain and G. I. Webb. The need for low bias algorithms in classification learning from large data sets. In *Principles of Knowledge Discovery and Datamining*, volume 2431 of *LNCS*, pages 62–73, 2002.

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

[25] J. Branke, E. Salihoğlu, and Ş. Uyar. Towards an analysis of dynamic environments. In *Proceedings of the ACM Genetic and Evolutionary Computation Conference*, pages 1433–1440, 2005.

[26] G. Brown and L. I. Kuncheva. "Good" and "bad" diversity in majority vote ensembles. In *Multiple Classifier Systems*, volume 5997 of *LNCS*, pages 124–133, 2010.

[27] D. Brzezinski and J. Stefanowski. Reacting to different types of concept drift: The accuracy updated ensemble algorithm. *IEEE Transactions on Neural Networks and Learning Systems*, 25(1):81–94, 2014.

[28] E. K. Burke, S. Gustafson, and G. Kendall. Diversity in genetic programming: An analysis of measures and correlation with fitness. *IEEE Transactions on Evolutionary Computation*, 8(1):47–62, 2004.

[29] M. Butler and D. Kazakov. A learning adaptive Bollinger band system. In *IEEE Conference on Computational Intelligence on Financial Engineering & Economics*, pages 1–8, 2012.

[30] R. Calabretta, S. Nolfi, D. Parisi, and G. P. Wagner. Duplication of modules facilitates the evolution of functional specialization. *Artificial Life*, 6(1):69–84, 2000.

[31] E. Carreño Jara. Long memory time series forecasting by using genetic programming. *Genetic Programming and Evolvable Machines*, 12(3):429–456, 2011.

[32] A. Cervantes, P. Isasi, C. Gagné, and M. Parizeau. Learning from non-stationary data using a growing network of prototypes. In *IEEE Congress on Evolutionary Computation*, pages 2634–2641, 2013.

[33] O. Chapelle, B. Scholkopf, and A. Zien. *Semi-Supervised Learning*. MIT Press, 2006.

[34] S. Chen and H. He. Towards incremental learning of non-stationary imbalanced data stream: A multiple selectively recursive approach. *Evolving Systems*, 2(1):35–50, 2011.

[35] M. Chorev and L. Carmel. The function of introns. *Frontiers in Genetics*, 3(55), 2012. DOI: 10.3389/fgene.2012.00055.

[36] J. Clune, J.-B. Mouret, and H. Lipson. The evolutionary origins of modularity. *Proceedings of the Royal Society: B – Biological Sciences*, 280:20122863:1–9, 2013.

[37] H. G. Cobb. An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent non-stationary environments. Technical Report AIC-90-001, Naval Research Laboratory, Washington, USA, 1990.

[38] L. Cohen, G. Avrahami-Bakish, M. Last, A. Kandel, and O. Kipersztok. Real-time data mining of non-stationary data streams from sensor networks. *Information Fusion*, 9(3):344–353, 2008.

[39] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.

[40] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

[41] R. Curry and M. I. Heywood. One-class genetic programming. In *European Conference on Genetic Programming*, volume 5481 of *LNCS*, pages 1–12, 2009.

[42] R. Curry, P. Lichodzijewski, and M. I. Heywood. Scaling genetic programming to large datasets using hierarchical dynamic subset selection. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 37(4):1065–1073, 2007.

[43] H. H. Dam, C. Lokan, and H. A. Abbass. Evolutionary online data mining: An investigation in a dynamic environment. In *Studies in Computational Intelligence*, volume 51, chapter 7, pages 153–178. Springer, 2007.

[44] T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi. An information-theoretic approach to detecting changes in multi-dimensional data streams. In *Proceedings of the Symposium on the Interface of Statistics*, 2006.

[45] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. In *ACM–SIAM Symposium on Discrete Algorithms*, pages 635–644, 2002.

[46] A. P. Dawid. Statistical theory: The prequential approach. *Journal of the Royal Statistical Society-A*, 147:278–292, 1984.

[47] E. D. de Jong. A monotonic archive for pareto-coevolution. *Evolutionary Computation*, 15(1):61–94, 2007.

[48] K. A. de Jong. Evolving in a changing world. In *Proceedings of the International Symposium on Foundations of Intelligent Systems*, pages 512–519. Springer, 1999.

[49] I. Dempsey, M. O'Neill, and A. Brabazon. Adaptive trading with grammatical evolution. In *IEEE Congress on Evolutionary Computation*, pages 2587–2592, 2006.

[50] I. Dempsey, M. O'Neill, and A. Brabazon. *Foundations in Grammatical Evolution for Dynamic Environments*, volume 194 of *Studies in Computational Intelligence*. Springer, 2009.

[51] I. Dempsey, M. O'Neill, and A. Brabazon. *Survey of EC in dynamic environments*, chapter 3, pages 25–54. 2009. In ([50]).

[52] M. A. H. Dempster and C. M. Jones. A real-time adaptive trading system using genetic programming. *Quantitative Finance*, 1:397–413, 2001.

[53] G. Ditzler and R. Polikar. Hellinger distance based drift detection for non-stationary environments. In *IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments*, pages 41–48, 2011.

[54] G. Ditzler and R. Polikar. Semi-supervised learning in non-stationary environments. In *IEEE–INNS International Joint Conference on Neural Networks*, pages 1–8, 2011.

[55] G. Ditzler and R. Polikar. Incremental learning of concept drift from streaming imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 25(10):2283–2301, 2013.

[56] G. Ditzler, G. Rosen, and R. Polikar. Discounted expert weighting for concept drift. In *IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments*, pages 61–66, 2013.

[57] P. Domingos and G. Hulten. Catching up with the data: Research issues in mining data streams. In *Workshop on Research Issues in Data Mining and Knowledge Discovery*, 2001.

[58] J. Doucette and M. I. Heywood. GP classification under imbalanced data sets: Active subsampling AUC approximation. In *European Conference on Genetic Programming*, volume 4971 of *LNCS*, 2008.

[59] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, 2nd edition, 2001.

[60] K. Dyer, R. Capo, and R. Polikar. COMPOSE: A semi-supervised learning framework for initially labeled non-stationary streaming data. *IEEE Transactions on Neural Networks and Learning Systems*, 25(1):12–26, 2014.

[61] M. Ebner, M. Shackleton, and R. Shipman. How neutral networks influence evolvability. *Complexity*, 7(2):19–33, 2002.

[62] J. Eggermont, A. E. Eiben, and J. I. van Hemert. Adapting the fitness function in GP for data mining. In *European Conference on Genetic Programming*, volume 1598 of *LNCS*, pages 195–204, 1999.

[63] J. Eggermont, T. Lenaerts, S. Poyhonen, and A. Termier. Raising the dead: Extending evolutionary algorithms with a case-based memory. In *European Conference on Genetic Programming*, volume 2038 of *LNCS*, pages 280–290, 2001.

[64] A. Ekárt and S. Németh. Maintaining the diversity of genetic programming. In *European Conference on Genetic Programming*, volume 2278 of *LNCS*, pages 162–171, 2002.

[65] R. Elwell and R. Polikar. Incremental learning of concept drift in non-stationary environments. *IEEE Transactions on Neural Networks*, 22(10):1517–1531, 2011.

[66] S. Esmeir and S. Markovitch. Anytime learning of any cost classifiers. *Machine Learning*, 82:445–473, 2011.

[67] C. Espinosa-Soto and A. Wagner. Specialization can drive the evolution of modularity. *PLoS: Computational Biology*, 6:e1000719:1–10, 2010.

[68] W. Fan, Y. Huang, H. Wang, and P. S. Yu. Active mining of data streams. In *Proceedings of SIAM International Conference on Data Mining*, pages 457–461, 2004.

[69] T. Fawcett. "in vivo" spam filtering: A challenge problem for KDD. *ACM SIGKDD Explortations*, 5(2):140–198, 2003.

[70] A. Fern and R. Givan. Online ensemble learning: An empirical study. *Machine Learning*, 53:71–109, 2003.

[71] P. Fernandez-Blanco, D. Bosdas-Sego, F. Soltero, and J. I. Hidalgo. Technical market indicators optimization using evolutionary algorithms. In *ACM Genetic and Evolutionary Computation Conference – ARC-FEC Workshop*, pages 1851–1858, 2008.

[72] S. G. Ficici and J.B. Pollack. Pareto optimality in coevolutionary learning. In *European Conference on Artificial Life*, pages 286–297, 2001.

[73] G. Folino and G. Papuzzo. Handling different categories of concept drift in data streams using distributed GP. In *European Conference on Genetic Programming*, volume 6021 of *LNCS*, pages 74–85, 2010.

[74] G. Folino, C. Pizzuti, and G. Spezzano. Training distributed GP ensemble with a selection algorithm based on clustering and pruning for pattern classification. *IEEE Transactions on Evolutionary Computation*, 12(4):458–468, 2008.

[75] Y. Freund and R. Shapire. A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139, 1997.

[76] J. Gama. *Knowledge discovery from data streams*. CRC Press, 2010.

[77] J. Gama. A survey on learning from data streams: Current and future trends. *Progress in Artificial Intelligence*, 1(1):45–55, 2012.

[78] J. Gama, P. Medas, G. Castillo, and P. P. Rodrigues. Learning with drift detection. In *Advances in Artificial Intelligence*, volume 3171 of *LNCS*, pages 66–112, 2004.

[79] J. Gama, R. Sebastião, and P. Rodrigues. On evaluating stream learning algorithms. *Machine Learning*, 90(3):317–346, 2013.

[80] J. Gama, R. Sebastiao, and P. P. Rodrigues. Issues in evaluation of stream learning algorithms. In *ACM Conference on Knowledge Discovery and Data Mining*, pages 329–338, 2009.

[81] J. Gao, W. Fan, and J. Han. On appropriate assumptions to mine data streams: Analysis and practice. In *IEEE International Conference on Data Mining*, pages 143–152, 2007.

[82] J. W. Gao, W. Fan, J. Han, and P. S. Yu. A general framework for mining concept-drifting data streams with skewed distributions. In *Proceedings of SIAM International Conference on Data Mining*, pages 3–14, 2007.

[83] C. Gathercole and P. Ross. Dynamic training subset selection for supervised learning in genetic programming. In *Parallel Problem Solving Nature*, volume 866 of *LNCS*, pages 312–321, 1994.

[84] A. Ghazikhani, R. Monsefi, and H. S. Yazdi. Online cost-sensitive neural network classifiers for non-stationary and imbalanced data streams. *Neural Computation and Applications*, 23:1283–1295, 2013.

[85] A. Ghosh, S. Tstutsui, and H.Tanaka. Function optimization in non-stationary environment using steady state genetic algorithms with aging of individuals. In *IEEE Conference on Evolutionary Computation*, pages 666–671, 1998.

[86] A. Godase and V. Attar. Classification of data streams with skewed distributions. In *IEEE Workshop on Evolving and Adaptive Intelligent Systems*, pages 151–156, 2013.

[87] J. J. Greffenstette. Genetic algorithms for changing environments. In *Proceedings of Parallel Problem Solving from Nature*, volume 2, pages 137–144. Elsevier, 1992.

[88] S. Grossberg. Nonlinear neural networks: Principles, mechanisms, and architectures. *Neural Networks*, 1(2):17–61, 1988.

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

[90] H. He and S. Chen. IMORL: Incremental multiple-object recognition and localization. *IEEE Transactions on Neural Networks*, 19(10):1727–1738, 2008.

[91] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

[92] R. C. Holt. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–91, 1993.

[93] G. S. Hornby. ALPS: The age layered population structure for reducing the problem of premature convergence. In *ACM Genetic and Evolutionary Computation Conference*, pages 815–822, 2006.

[94] T. Hu and W. Banzhaf. Neutrality and variability: Two sides of evolvability in linear genetic programming. In *ACM Genetic and Evolutionary Computation Conference*, pages 963–970, 2009.

[95] T. Hu and W. Banzhaf. The role of population size in rate of evolution in genetic programming. In *European Conference on Genetic Programming*, volume 5481 of *LNCS*, pages 85–96, 2009.

[96] T. Hu and W. Banzhaf. Evolvability and speed of evolutionary algorithms in light of recent developments in biology. *Journal of Artificial Evolution and Applications*, 2010:568375–1 – 28, 2010.

[97] S. Huang and Y. Dong. An active learning system for mining time changing data streams. *Intelligent Data Analysis*, 11(4):401–419, 2007.

[98] L. Huelsbergen. Finding general solutions to the parity problem by evolving machine-language representations. In *Proceedings of Annual Conference on Genetic Programming*, pages 158–166. Morgan Kaufmann, 1998.

[99] E. Ikonomovska. DataExpo: Airline dataset, 2009.

[100] K. Imamura, T. Soule, R. B. Heckendorn, and J. A. Foster. Behavioral diversity and a probabilistically optimal GP ensemble. *Genetic Programming and Evolvable Machines*, 4(3):235–254, 2003.

[101] N. Japkowicz and M. Shah. *Evaluating Learning Algorithms: A classification perspective*. Cambridge University Press, 2012.

[102] M. Karnick, M. D. Muhlbaier, and R. Polikar. Incremental learning in non-stationary environments with concept drift using a multiple classifier based approach. In *Proceedings of the International Conference on Pattern Recognition*, pages 1–4, 2008.

[103] N. Kashtan, E. Noor, and U. Alon. Varying environments can speed up evolution. *Proceedings of the National Academy of Sciences*, 104(34):13713–13716, 2007.

[104] A. Kattan, A. Agapitos, and R. Poli. Unsupervised problem decomposition using genetic programming. In *Proceedings of the European Conference on Genetic Programming*, volume 6021 of *LNCS*, pages 122–133, 2010.

[105] D. Kifer, S. Ben-David, and J. Gehrke. Detecting change in data streams. In *Proceedings of the International Conference on Very Large Data Bases*, pages 180–191. Morgan Kaufmann, 2004.

[106] R. Klinkenberg and I. Renz. Adaptive information filtering: Learning in the presence of concept drifts. In *ICML / AAAI Workshop on Learning for Text Categorization*, pages 33–40. AAAI, 1998.

[107] J. Z. Kolter and M. A. Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *Journal of Machine Learning*, 8:2755–2790, 2007.

[108] M. F. Korns. Symbolic regression of conditional target expressions. In R. Riolo, U.-M. O'Reilly, and T. McConaghy, editors, *Genetic Programming Theory and Practice VII*, chapter 13, pages 211–228. Springer, 2010.

[109] T. Kovacs. *Strength or accuracy: Credit assignment in learning classifier systems*. Springer, 2004.

[110] K. Krawiec. Genetic programming-based construction of features for machine learning and knowledge discovery tasks. *Genetic Programming and Evolvable Machines*, 3(4):329–343, 2002.

[111] H.-P. Kriegel, P. Kröger, and A. Zimek. Subspace clustering. *WIREs Data Mining and Knowledge Discovery*, 2:351–364, 2012.

[112] K.Trojanowski and Z. Michalewicz. Evolutionary optimization in non-stationary environments. *Journal of Computer Science and Technology*, 1(2):93–124, 2000.

[113] L. I. Kuncheva. Classifier ensembles for changing environments. In *Multiple classifier systems*, volume 3077 of *LNCS*, pages 1–15, 2004.

[114] T. N. Lal, O. Chapelle, J. Weston, and A. Elisseeff. Embedded methods. In *Feature extraction: Foundations and Applications*, volume 207 of *Studies in Fuzziness and Soft Computing*, chapter 5, pages 137–165. Springer, 2006.

[115] W. B. Langdon and B. F. Buxton. Evolving receiver operating characteristics for data fusion. In *Proceedings of the European Conference on Genetic Programming*, volume 2038 of *LNCS*, pages 87–96, 2001.

[116] W. B. Langdon and R. Poli. *Foundations of Genetic Programming*. Springer, 2001.

[117] T. Lange and A. Rahbek. An introduction to regime switching time series models. In T. G. Anderson, R. A. Davis, J. P. Kreiß, and T. V. Mikosch, editors, *Handbook of Financial Time Series*, pages 871–887. Spriner, 2009.

[118] C. Lanquillon. Information filtering in changing domains. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 41–48, 1999.

[119] D. Lewis. Evaluating and optimizing autonomous text classification systems. In *ACM International Conference on Research and Development in Information Retrieval*, pages 246–254, 1995.

[120] D. Lewis, Y. Yang, T. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[121] J. Lewis, E. Hart, and G. Ritchie. A comparison of dominance mechanisms and simple mutation on non-stationary problems. In *Parallel Problem Solving from Nature*, volume 1498 of *LNCS*, pages 139–148, 1998.

[122] P. Li, X. Wu, and X. Hu. Mining recurring concept drifts with limited labeled streaming data. *ACM Transactions on Intelligent Systems and Technology*, 3(2):29:1–29:32, 2012.

[123] P. Lichodzijewski and M. I. Heywood. Managing team-based problem solving with Symbiotic Bid-based Genetic Programming. In *ACM Genetic and Evolutionary Computation Conference*, pages 363–370, 2008.

[124] P. Lindstrom, B. MacNamee, and S. J. Delany. Handling concept drift in a text data stream constrained by high labelling cost. In *Proceedings of the International Florida Artificial Intelligence Research Society Conference*. AAAI, 2010.

[125] P. Lindstrom, B. MacNamee, and S. J. Delany. Drift detection using uncertainty distribution divergence. *Evolutionary Intelligence*, 4(1):13–25, 2013.

[126] A. Loginov and M. I. Heywood. On the impact of streaming interface heuristics on GP trading agents: An FX benchmarking study. In *Proceedings of the ACM Genetic and Evolutionary Computation Conference*, pages 1341–1348, 2013.

[127] A. Loginov and M. I. Heywood. On evolving multi-agent FX traders. In *EvoApplications*, volume 8602 of *LNCS*, 2014.

[128] E. Lughofer. On-line active learning based on enhanced reliability concepts. In *IEEE Workshop on Evolving and Adaptive Intelligent Systems*, pages 1–6, 2013.

[129] S. Ma and C. Ji. Performance and efficiency: Recent advances in supervised learning. *Proceedings of the IEEE*, 87(9):1519–1536, 1999.

[130] M. Markou and S. Singh. Novelty Detection: A review–part 1: Statistical approaches. *Signal Processing*, 83:2481–2497, 2003.

[131] M. Markou and S. Singh. Novelty Detection: A review–part 2: Neural network based approaches. *Signal Processing*, 83:2499–2521, 2003.

[132] A. R. McIntyre and M. I. Heywood. Cooperative problem decomposition in Pareto competitive classifier models of coevolution. In *European Conference on Genetic Programming*, volume 4971 of *LNCS*, pages 289–300, 2008.

[133] A. R. McIntyre and M. I. Heywood. Pareto cooperative-competitive genetic programming: A classification benchmarking study. In R. Riolo, T. Soule, and B. Worzel, editors, *Genetic Programming Theory and Practice*, volume IV, chapter 4, pages 43–60. Springer, 2008.

[134] A. R. McIntyre and M. I. Heywood. Classification as clustering: A pareto cooperative-competitive GP approach. *Evolutionary Computation*, 19(1):137–166, 2011.

[135] Jan Hendrik Metzen, Mark Edgington, Yohannes Kassahun, and Frank Kirchner. Analysis of an evolutionary reinforcement learning method in a multiagent domain. In *Proceedings of the ACM International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 291–298, 2008.

[136] L. L. Minku. Concept drift datasets and generators, 2010. `http://www.cs.bham.ac.uk/~minkull/opensource.html`.

[137] L. L. Minku, H. Inoue, and X. Yao. Negative correlation in incremental learning. *Natural Computing Journal*, 8:289–320, 2009.

[138] L. L. Minku, A. P. White, and X. Yao. The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on Knowledge and Data Engineering*, 22(5):730–742, 2010.

[139] L. L. Minku and X. Yao. DDD: A new ensemble approach for dealing with concept drift. *IEEE Transactions on Knowledge and Data Engineering*, 24(4):619–633, 2012.

[140] N. Mori, H. Kita, and Y. Nishikawa. Adaptation to a changing environment by means of the feedback thermodynamical genetic algorithm. In *Parallel Problem Solving from Nature*, volume 1498 of *LNCS*, pages 149–157, 1998.

[141] R. W. Morrison. *Designing evolutionary algorithms for dynamic environments*. Natural Computing. Springer, 2004.

[142] Neurotech. Pakdd 2009 data mining competition, 2009.

[143] H. M. Nguyen, E. W. Cooper, and K. Kamei. Online learning from imbalanced data streams. In *International Conference on Soft Computing and Pattern Recognition*, pages 347–352, 2011.

[144] N. Nikolaev and H. Iba. Accelerated genetic programming of polynomials. *Genetic Programming and Evolvable Machines*, 2(3):231–257, 2000.

[145] K. Nishida and K. Yamauchi. Learning, detecting, understanding, and predicting concept changes. In *IEEE-INNS International Joint Conference on Neural Networks*, pages 2280–2287, 2009.

[146] J. Noble and R. Watson. Pareto coevolution: Using performance against coevolved opponents in a game as dimensions for pareto selection. In *Genetic and Evolutionary Computation Conference*, pages 493–500. Morgan Kaufmann, 2001.

[147] U.S. National Oceanic and Atmospheric Administration. Federal climate complex global surface summary of day data, 2010. `ftp://ftp.ncdc.noaa.gov/pub/data/gsod`.

[148] M. O'Neill and C. Ryan. Grammatical evolution by grammatical evolution: The evolution of grammar and genetic code. In *European Conference on Genetic Programming*, volume 3003 of *LNCS*, pages 138–149, 2004.

[149] N. C. Oza and S. Russell. Experimental comparison of online and batch versions of bagging and boosting. In *ACM International Conference on Knowledge Discovery and Data Mining*, pages 359–364, 2001.

[150] G. L. Pappa, G. Ochoa, M. R. Hyde, A. A. Freitas, J. Woodward, and J. Swan. Contrasting meta-learning and hyper-heuristic research: The role of evolutionary algorithms. *Genetic Programming and Evolvable Machines*, 15(1):3–35, 2014.

[151] M. Parter, N. Kashtan, and U. Alon. Facilitated variation: How evolution learns from past environments to generalize to new environments. *PLoS Computational Biology*, 4(11):e1000206, 2008.

[152] A. Pocock, P. Yiapanis, J. Singer, M. Luján, and G. Brown. Online non-stationary boosting. In *Multiple Classifier Systems*, volume 5997 of *LNCS*, pages 205–214, 2010.

[153] R. Polikar and R. Elwell. Benchmark datasets for evaluating concept drift / nse algorithms, 2011. `http://users.rowan.edu/?polikar/research/NSE`.

[154] R. Polikar, L. Udpa, S. S. Udpa, and V. Honavar. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man, and Cybernetics – Part C*, 31(4):497–508, 2001.

[155] A. Prugel-Bennett. Benefits of a population: Five mechanisms that advantage population-based algorithms. *IEEE Transactions on Evolutionary Computation*, 14(4):500–517, 2010.

[156] J. Quinonero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, editors. *Dataset shift in machine learning*. MIT Press, 2009.

[157] S. Rahimi, A. R. McIntyre, M. I. Heywood, and N. Zincir-Heywood. Label free change detection on streaming data with cooperative multi-objective genetic programming. In *ACM Genetic and Evolutionary Computation Conference*, pages 159–160, 2013.

[158] K. Rodríguez-Vázquez and P. J. Fleming. Evolution of mathematical models of chaotic systems based on multi objective genetic programming. *Knowledge and Information Systems*, 8(2):235–256, 2005.

[159] R. Schapire and Y. Freund. Decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and Systems Sciences*, 55(1):119–139, 1997.

[160] M. Scholz and R. Klinkenberg. Boosting classifiers for drifting concepts. *Intelligent Data Analysis*, 11(1):3–28, 2007.

[161] R. Schwaerzel and T. Bylander. Predicting currency exchange rates by genetic programming with trigonometric functions and high-order statistics. In *ACM Genetic and Evolutionary Computation Conference*, pages 955–956, 2006.

[162] R. Sebastio and J. Gama. Change detection in learning histograms from data streams. In *Proceedings of the Portuguese Conference on Artificial Intelligence*, volume 4874 of *LNCS*, pages 112–123. Springer, 2007.

[163] H. A. Simon. The architecture of complexity. *Proceedings of the American Philosophical Society*, 106:467–482, 1962.

[164] P. Sobolewski and M. Woźniak. LDCnet: Minimizing the cost of supervision for various types of concept drift. In *IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments*, pages 68–75, 2013.

[165] D. Song, M. I. Heywood, and A. N. Zincir-Heywood. Training genetic programming on half a million patterns: An example from anomaly detection. *IEEE Transactions Evolutionary Computation*, 9(3):225–239, 2005.

[166] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

[167] R. Stapenhurst and G. Brown. Theoretical and empirical analysis of diversity in non-stationary learning. In *IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments*, pages 25–32, 2011.

[168] A. Storkey. *When training and test sets are different: Characterizing learning transfer*, chapter 1, pages 3–28. 2009. In ([156]).

[169] W. Street and Y. Kim. A streaming ensemble algorithm (SEA) for large-scale classification. In *ACM Conference on Knowledge Discovery and Data Mining*, pages 377–382, 2001.

[170] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT Press, 1998.

[171] R. Swan and J. Allan. Extracting significant time varying features from text. In *ACM International Conference on Information and Knowledge Management*, pages 38–45, 1999.

[172] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen. Dynamic integration of classifiers for handling concept drift. *Information Fusion*, 9(1):56–68, 2008.

[173] P. D. Turney. Increasing evolvability considered as a large-scale trend in evolution. In *Genetic and Evolutionary Computation Conference: Workshop on Evolvability*, pages 43–46. Morgan Kaufmann, 1999.

[174] A.Ş. Uyar and A. E. Harmanci. Performance comparisons of genotype-to-phenotype mapping schemes for diploid representations in changing environments. In *International Conference on Recent Advances in Soft Computing*, pages 128–134, 2002.

[175] I. Žliobaitė. Change with delayed labelling: When is it detectable? In *IEEE International Conference on Data Mining Workshops*, pages 843–850, 2010.

[176] I. Žliobaitė, A. Bifet, B. Pfahringer, and G. Holmes. Active learning with evolving streaming data. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 597–612. Springer, 2011.

[177] I. Žliobaitė, A. Bifet, B. Pfahringer, and G. Holmes. Active learning with drifting streaming data. *IEEE Transactions on Neural Networks and Learning Systems*, 25(1):27–54, 2014.

[178] I. Žliobaitė and B. Gabrys. Adaptive preprocessing for streaming data. *IEEE Transactions on Knowledge and Data Engineering*, 26(2):309–321, 2014.

[179] A. Vahdat, A. Atwater, A. R. McIntyre, and M. I. Heywood. On the application of GP to streaming data classification tasks with label budgets. In *ACM Genetic and Evolutionary Computation Conference: ECBDL Workshop*, pages 1287–1294, 2014.

[180] 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. Springer, 2015.

[181] H. Valizadegan and P.-N. Tan. A prototype-driven framework for change detection in data stream classification. In *IEEE Symposium on Computational Intelligence and Data Mining*, pages 88–95, 2007.

[182] L. Vanneschi and G. Cuccu. Variable size population for dynamic optimization with genetic programming. In *ACM Genetic and Evolutionary Computation Conference*, pages 1895–1896, 2009.

[183] W. Verbeke, K. Dejager, D. Martens, J. Nur, and B. Basens. New insights into churn prediction in the telecommunication sector: A profit driven data mining approach. *European Journal of Operational Research*, 218:211–229, 2012.

[184] E. Vladislavleva, G. Smits, and D. den Hertog. On the importance of data balancing for symbolic regression. *IEEE Transactions on Evolutionary Computation*, 14(2):252–227, 2010.

[185] P. Vorburger and A. Bernstein. Entropy-based concept shift detection. In *Proceedings of the Sixth International Conference on Data Mining*, pages 1113–1118, 2006.

[186] A. Wagner. Environmental change in adaptation and innovation. In *The origins of evolutionary innovations*, chapter 11. Oxford University Press, 2011.

[187] G. P. Wagner and L. Altenberg. Complex adaptations and the evolution of evolvability. *Complexity*, 50(3):433–452, 1996.

[188] N. Wagner, Z. Michalewicz, M. Khouja, and R. R. McGregor. Time series forecasting for dynamic environments: The DyFor genetic program model. *IEEE Transactions on Evolutionary Computation*, 11(4):433–452, 2007.

[189] J. Wang, P. Zhao, S. C. H. Hoi, and R. Jin. Online feature selection and its applications. *IEEE Transactions on Knowledge and Data Engineering*, 26(3):698–710, 2014.

[190] S. Wang, L. L. Minku, and X. Yao. A learning framework for online class imbalance learning. In *IEEE Symposium on Computational Intelligence and Ensemble Learning*, pages 36–45, 2013.

[191] Y. Wang and M. Wineberg. Estimation of evolvability genetic algorithm and dynamic environments. *Genetic Programming and Evolvable Machines*, 7(3):355–382, 2006.

[192] R. A. Watson and J. B. Pollack. Modular interdependency in complex dynamic systems. *Artificial Life*, 11(4):445–457, 2005.

[193] G. M. Weiss and R. Provost. Learning when training data are costly: The effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, 19:315–354, 2003.

[194] G. Widmer and M. Kubat. Effective learning in dynamic environments by explicit context tracking. In *Proceedings of the European Conference on Machine Learning*, volume 667 of *LNCS*, pages 227–243, 1993.

[195] G. Wilson and W. Banzhaf. Interday and intraday stock trading using PAM developmental GP and linear GP. In A. Brabazon, M. OÕNeill, and D. G. Maringer, editors, *Natural Computing in Computational Finance 3*, volume 293 of *SCI*, chapter 11, pages 191–212. Springer, 2010.

[196] X. Wu, K. Yu, W. Ding, H. Wang, and X. Zhu. Online feature selection with streaming features. *IEEE Transactions on Pattern Analysis and Machine Learning*, 35(5):1178–1182, 2013.

[197] Y. Xu, S. Furao, O. Hasegawa, and J. Zhao. An online incremental learning vector quantization. In *Advances in Knowledge Discovery and Data Mining*, volume 5476 of *LNAI*, pages 1046–1053, 2009.

[198] S. Yang. Dominance learning in diploid genetic algorithms for dynamic optimization problems. In *ACM Genetic and Evolutionary Computation Conference*, pages 1435–1448, 2006.

[199] Y. Yang, X. Wu, and X. Zhu. Mining in anticipation for concept change: Proactive–reactive prediction in data streams. *Data Mining and Knowledge Discovery*, 13(3):261–289, 2006.

[200] M. Zhang and W. Smart. Using Gaussian distribution to construct fitness functions in genetic programming for multiclass object classification. *Pattern Recognition Letters*, 27(11):1266–1274, 2006.

[201] P. Zhang, X. Zhu, and L. Guo. Mining data streams with labeled and unlabeled training examples. In *IEEE International Conference on Data Mining*, pages 627–636, 2009.

[202] P. Zhang, X. Zhu, J. Tan, and L. Guo. Classifier and cluster ensembles for mining concept drifting data streams. In *IEEE International Conference on Data Mining*, pages 1175–1180, 2010.

[203] X. Zhu, P. Zhang, X. Lin, and Y. Shi. Active learning from stream data using optimal weight classifier ensemble. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 40(6):1607–1621, 2010.

[204] T. Ziemke, N. Bergfeldt, G. Buason, T. Susi, and H. Svensson. Evolving cognitive scaffolding and environment adaptation: A new research direction for evolutionary robotics. *Connection Science*, 16(4):339–350, 2004.