A Concept-Drift Perspective on Prototype Selection and Generation

Ludmila I. Kuncheva
School of Computer Science
Bangor University
Bangor, Gwynedd, UNITED KINGDOM
Email: l.i.kuncheva@bangor.ac.uk

Iain A. D. Gunn
School of Computer Science
Bangor University
Bangor, Gwynedd, UNITED KINGDOM
Email: i.gunn@bangor.ac.uk

***Abstract*—This study brings together systematised views of two related areas: data editing for the nearest neighbour classifier and adaptive learning in the presence of concept drift. The growing number of studies in the intersection of these areas warrants a closer look. We revise and update the taxonomies of the two areas proposed in the literature and argue that they are not sufficiently discriminative with respect to methods for prototype selection and prototype generation in the presence of concept drift. We proceed to create a bespoke taxonomy of these methods and illustrate it with ten examples from the literature. The new taxonomy can serve as a road-map for researching the intersection area and inform the development of new methods.**

I. Introduction

In their book [1], Devroye et al. open the chapter on nearest neighbour rules with one of the most important messages in pattern recognition: "Simple rules survive." Indeed, the nearest neighbour family of classification methods proposed in 1951 [2], [3] rival today's state-of-the-art classifiers in accuracy and elegance. According to the k-nearest neighbour classifier, an object $\mathbf{x} \in \mathbb{R}^d$ is assigned to the class most represented among its k nearest neighbours from a given reference set S with N objects.[1] A notable example is 1-nn, the nearest neighbour classifier.

Reduction of the computational demands of k-nn without sacrificing much accuracy is often sought through reducing the size of the reference set S. Starting with the classical condensing [4] and editing [5] algorithms, a wealth of data editing approaches and methods have been proposed and periodically summarised in insightful surveys [6]–[10].

Recently, there has been development of "online" or "streaming" versions of the nearest neighbour methods. As data streams by their nature cannot be stored in full, data editing is implied. Also among the challenges of modern data is its non-stationary behaviour, termed also "changing environments", "concept drift" or "concept shift" [11]. The large body of literature addressing this challenge has invited a collection of timely and comprehensive surveys [12]–[14].

The aim of this study is to explore the types of edited nearest-neighbour classifiers used in the context of concept drift. We discuss possible classifications of such algorithms in terms of existing taxonomies which have been developed, separately, for prototype methods and for adaptive learning. While the taxonomic diagrams we give for these two areas have new layouts, their content and level of detail are largely drawn from the existing literature. We are interested in marrying the two areas with a view to systematise the methods for data editing in the presence of concept drift. This will help explore the possibilities for creating new, more efficient and accurate methods in this group. To this end, we give a novel taxonomy showing the lines on which a fusion of the taxonomic concepts of the two areas might proceed.

The rest of the paper is organised as follows. Section II builds upon the recent taxonomies of prototype / instance selection and generation. Learning in the presence of concept drift is discussed briefly in Section III, highlighting the nearest neighbour involvement. Section IV presents our new taxonomy of nearest neighbour methods for data with concept drift, illustrated with a sample of such methods from past and recent literature. Our Conclusion section contains remarks about future developments of the proposed taxonomy.

II. Prototype/Instance Selection and Generation for the Nearest Neighbour Classifier

Due to its simplicity and accuracy, the k-nearest neighbour classifier is scored among the top ten algorithms in data mining [15]. The purpose of editing is to replace the reference set S with a *smaller* set of what will be called "prototypes". The meaning of "prototype" depends on the approach we choose for the data editing: prototype *selection* (instance selection) [9] or prototype *generation*[2] [10]. In prototype selection, the reduced set of prototypes, S' is a subset of S along with the labels of the objects. In prototype generation, the prototypes are allowed to be different points in the same space or to be extended as other structures such as hyper-rectangles or hyper-ellipses. Prototypes in the original space can be created by relabelling, merging or re-positioning members of an initial subset of S or can be obtained as cluster centres.

Although prototype selection and prototype generation are often treated as separate research areas [9], [10], for the

[1]The terms *object*, *instance*, and *example* will be used interchangeably throughout this paper.

[2]Synonyms of prototype generation in the literature are prototype *construction*, *extraction*, *reduction* and *replacement*

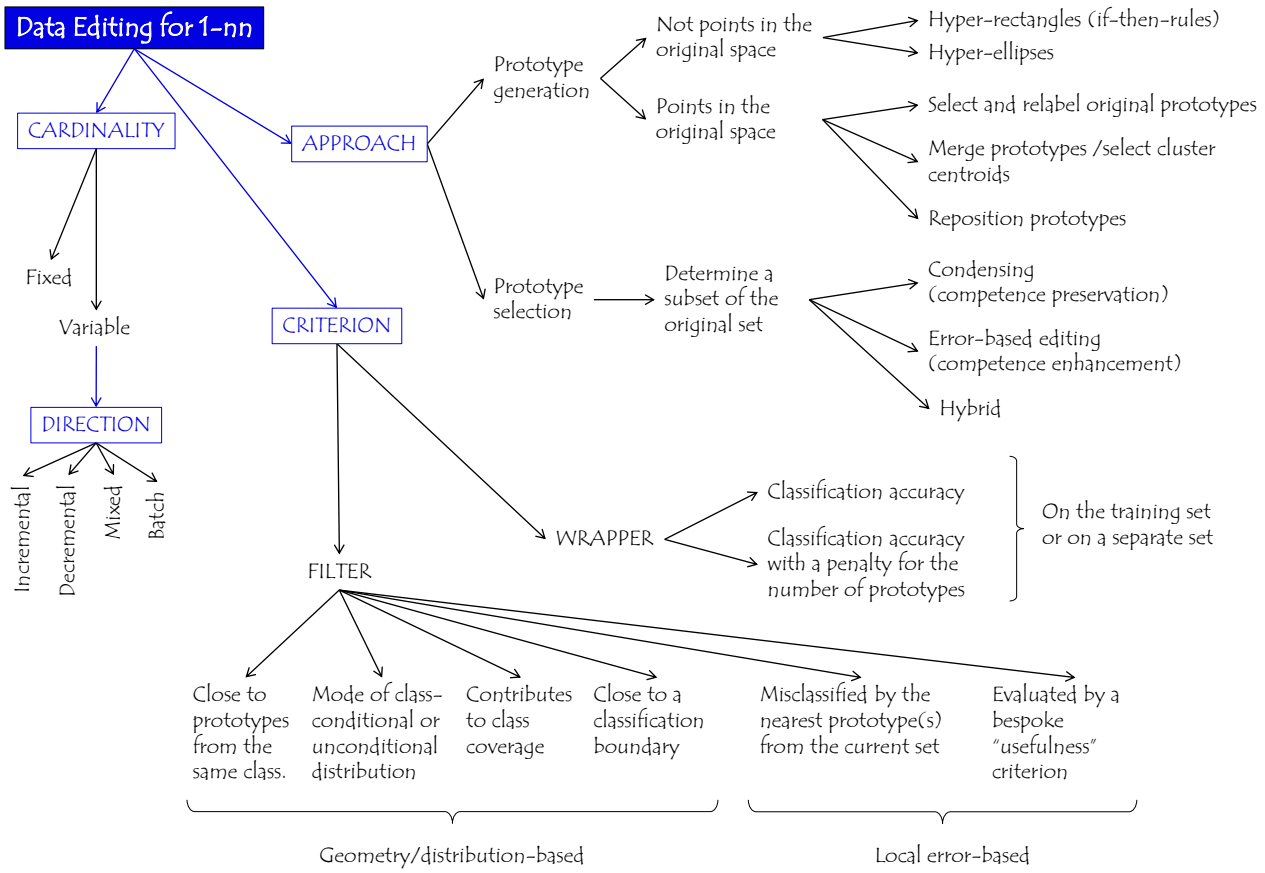


Fig. 1. A taxonomy of methods for prototype selection and prototype generation. The nodes in boxes show properties that should be specified, and their values are chosen among the leaves of the respective sub-tree.

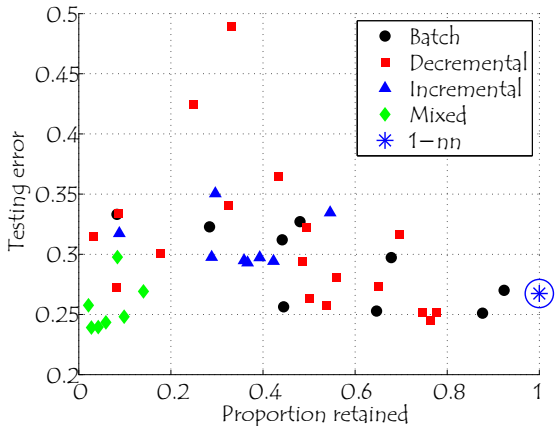
purposes of our study, we will join them and look at their common properties. One overlap between these areas is that instance selection algorithms have been found to be beneficial in initialising prototype generation algorithms such as LVQ [16]. Both approaches are suitable for dealing with streaming data and concept drift.

Taking inspiration from the published surveys, Figure 1 offers our view of a possible taxonomy for prototype selection and generation methods. The nodes in boxes show properties that should be specified, and their values are chosen among the leaves of the respective sub-tree. These properties may or may not prove to be transferable to classification of prototype selection / generation methods for concept drift.

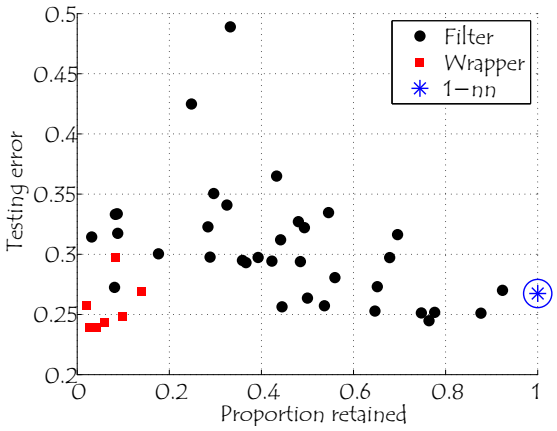
Our "Direction" and "Criterion" categories correspond to the "Direction" and "Evaluation" categories of García et al. [9]. "Direction" addresses the question of whether the prototype set is formed by successive additions to the empty set, by successive deletions from the whole training data, or by some mixed process in which the number of prototypes may vary up or down. A *batch* method is a deletion-based method in which each instance is assessed for removal before any of them is removed. "Criterion" refers to the method by which the algorithm evaluates a potential prototype set. Our category "Cardinality" concerns whether the number of prototypes may take different values, or is an input parameter of the algorithm [17].[3]

The "Approach" distinction between generation and selection has already been discussed. The different prototype generation approaches listed are self-explanatory. The different prototype selection approaches listed reflect a traditional distinction in prototype selection methods between *condensing* and *editing* methods, where "editing" was used in a more specific sense than is now common. Historically, the two streams of methods grew respectively out of Hart's condensing nearest neighbour and Wilson's edited nearest neighbour. The condensing stream aims at producing the smallest possible reference set whose resubstitution error is zero (called a consistent reference set). This group tends to preserve boundary objects. The editing group evolved on the philosophy that the boundary objects may contain noise and keeping them may do more harm than good; broadly speaking, these methods work by removing points which are misclassified according to their neighbours. Soon it was established that neither group

[3]In [9] a fixed number of prototypes was considered to be a possibility in the "Direction" category, "Fixed" methods being assumed to be a subset of "Mixed" methods.



(a) Methods labelled according to the "Direction" category.



(b) Methods labelled according to the "Criterion" category.

Fig. 2. Scatterplots of 42 prototype selection methods, showing experimental results from [9]. Each point is the average across 39 data sets. The mixed strategy, associated with the wrapper criterion, is better than the incremental, decremental, or batch strategies, associated with filter methods.

alone was successful enough, and the *hybrid* methods came into focus. We have retained this traditional distinction among prototype selection methods in Figure 1, although we believe the distinction is less useful for modern algorithms, especially for the case of streaming data subject to concept drift.

As an example of how this taxonomy works, consider Wilson's method (sometimes called *the* Edited Nearest Neighbour algorithm [7]). First, a leave-one-out k-nn (usually $k = 3$) is run on the whole set S and the misclassified objects are marked for deletion. The marked objects are removed, and the remaining ones are returned as the reduced reference set S'.

The representation of Wilson's method within the proposed taxonomy is as follows:

- APPROACH: Prototype selection; Editing
- CARDINALITY: Variable (The number of prototypes is an output parameter)
- DIRECTION: Batch (Objects removed simultaneously)
- CRITERION: Filter; Remove misclassified

Of course, there are aspects of prototype selection / generation which are left out of this taxonomy. Examples of these aspects are whether the methods have been biologically inspired (genetic algorithms [18]–[20], immune-based algorithms [21], ant colonies [22]), combined with feature selection [23]–[26] addressing the scalability of the problem [27]–[29] or striving for interpretability [30].

We will now evaluate the relative merit of some of the categories for the prototype *selection* approach using the comparative experiments on a large collection of data sets reported by García et al. [9]. Figure 2a demonstrates the advantage of the mixed strategy in the "Direction" category over incremental, decremental and batch strategies. The figure shows a scatterplot of 42 prototype selection methods. Each point is the average over 39 small data sets (number of points less than 2000), which were partitioned using the ten-fold cross-validation procedure [9]. The testing classification error is plotted against the prototype retention rate. The closer to the bottom left corner of the plot, the better the method. Similarly, the plot in Figure 2b shows the advantage of wrapper over filter methods. Of course, this advantage comes at a computational price.

We plot only the small-data-set results from [9] because selecting from a small set will be a typical task when prototype selection is used for streaming data in the presence of concept drift: the prototype set will have to be selected from a small window of data after a distribution change has been detected. Indeed, the very purpose of writing a streaming algorithm is to avoid applying the offline algorithm to a large data set.

III. LEARNING IN THE PRESENCE OF CONCEPT DRIFT

Consider streaming data such that:

- One data point $\mathbf{x} \in \mathbb{R}^d$ is received at time t.
- The class label of the point is not available at time t. The point is labelled by the classifier.
- The true label is revealed.

Receiving a batch of N points $X \subset \mathbb{R}^d$ at time t is an extension of this model where the time interval is set in such a way that the labels of the incoming points are only available at time $t+N$, after the N points in the batch have been labelled.

"Concept drift" is the generally accepted term for change in the probability distributions related to the problem. Sometimes this change includes appearance of new classes and collapsing of old ones. We are interested in concept drift to the extent that it affects adversely the performance of the classifier and requires action. Therefore, the occasional outlier or a short abnormal event should be treated as noise and ignored.

Concept drift is sometimes described as either "real" or "virtual". *Real* concept drift is a change in the posterior probability distributions $p(y|\mathbf{x})$, where y is the class label variable. Such change comes about when the labelling of the data changes while the underlying unconditional distribution $p(\mathbf{x})$ remains the same. An example is a change in user's preference. Suppose that the classifier is filtering spam from non-spam e-mail for a user called Bob. At some time moment Bob decides

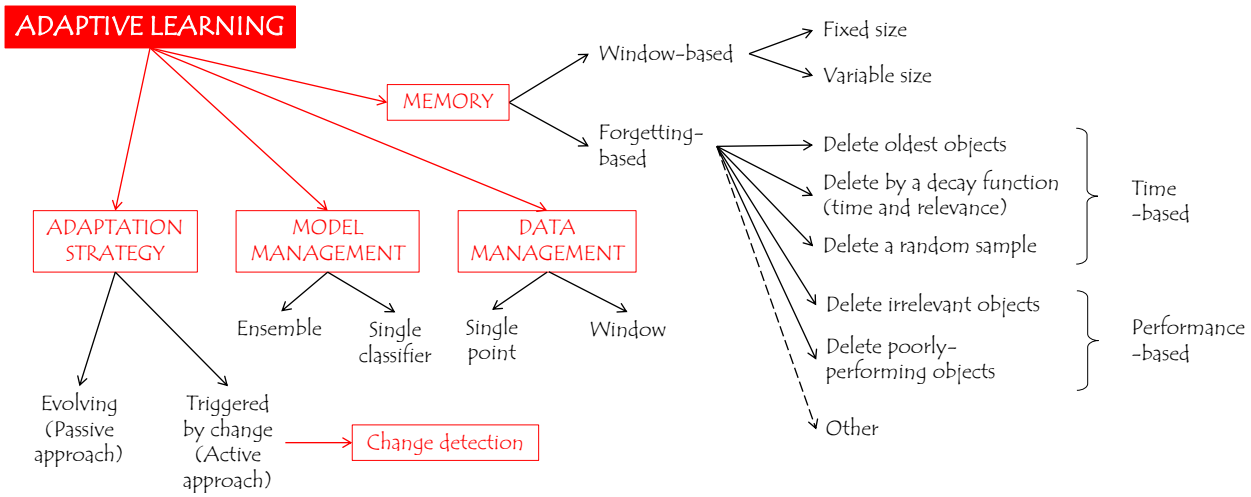


Fig. 3. A taxonomy of methods for adaptive learning. The nodes in boxes show properties that should be specified, and their values are chosen among the leaves of the respective sub-tree. The Change Detection box is included to represent a potentially large taxonomy of its own.

to buy a house, and with this, his e-mail preferences change towards receiving more news and ads about the house market. The e-mail flow distribution ($p(\mathbf{x})$) does not change but the posterior probabilities do. A message, say $\mathbf{z}$, about a new house in Bob's preferred location and price region would have had a large value of $P(y = \text{spam}|\mathbf{z})$ before but should have a small value now. *Virtual* concept drift, on the other hand, is a change in $p(\mathbf{x})$, which will likely lead to change in $p(y|\mathbf{x})$ anyway. The type of concept drift informs the choice of a change detection method.

Figure 3 shows a taxonomy of the adaptive learning methods drawing upon the taxonomies and mind maps in older and more recent surveys [12]–[14], [31].

The two adaptation strategies are "Evolving" (passive) and "Triggered by change" (active). According to the evolving strategy, the classifier is constantly updated regardless of whether or not concept drift is present. This strategy will eventually adapt the classifier to the new distribution but the process may not be efficient. In the active approach, the classifier is kept unchanged until a change has been detected, upon which a retraining procedure is carried out. Combinations of the two approaches have been explored as well [32].

The "Model management" category distinguishes between single and ensemble classifiers and "Data Management" divides the approaches into *Single Examples* and *Windows*, according to whether only the most recent point or a sliding window of the most recent points is available for learning [12]. Single-example Data Management is "online learning", in which only the most recent data point is available for learning.

In the data-editing taxonomy, Figure 1, most of the real diversity between algorithmic approaches fell under the single category of "Filter", and there is a similar problem with this adaptive-learning taxonomy: the variety of the adaptive learning methods is mostly owed to the "Memory" category. We should note that the forgetting-based alternatives are not mutually exclusive.

As an example, consider the simple (impractical as it may be) instance-based learning algorithm IB2 [33]. An incoming object at time t is classified first, and then added to the reference set if the reference set misclassifies it, before time $t + 1$. According to our taxonomy, the description of this method is as follows:

- ADAPTATION STRATEGY: Evolving (Passive approach)
- MODEL MANAGEMENT: Single classifier (1-nn)
- DATA MANAGEMENT: Window
- MEMORY.Window-based: Variable size.

The evident redundancy in describing both the Data management and Memory techniques as a "Window" indicates an area where the existing taxonomy might be improved for use with nearest-neighbour classifiers: for lazy learners, there is no distinction between "Memory" and "Data management" in the sense which the current taxonomy uses.

IV. PROTOTYPE SELECTION AND GENERATION FOR STREAMING DATA WITH CONCEPT DRIFT

Most of the established data editing methods do not address explicitly the problem of streaming data or streaming data *and* concept drift. "Streaming" implies that the data being received cannot be stored indefinitely. This requires that nearest neighbour methods for streaming data are necessarily data editing methods. Time-consuming iterative algorithms for prototype selection may be too computationally expensive for the streaming case.

This section reviews some existing editing k-nn methods for streaming data. We consider this to be a collection of example approaches and methods, and do not claim a comprehensive coverage of the area. We will argue that the two taxonomies in Figures 1 and 3 do not offer sufficient fidelity to discriminate

TABLE I
PROTOTYPE SELECTION METHODS FOR STREAMING DATA WITH CONCEPT DRIFT

Method name and reference	**Data editing**	**Adaptive Learning**
Instance-Based Learning	Approach: Prototype selection; Hybrid	Adaptation strategy: Evolving
IB3 [33]	Cardinality: Variable	Model management: Single classifier
	Direction: Mixed	Data management: Single point
	Criterion: Filter; Usefulness	Memory.Forgetting: Delete poorly-predicting objects
Instance-Based Learning	Approach: Prototype selection; Editing	Adaptation strategy: Mixed*
on Data Streams	Cardinality: Variable	Model management: Single classifier
INL-DS [34]	Direction: Mixed	Data management: Single point
	Criterion: Filter; Usefulness	Memory.Forgetting: Delete old/poorly-predicting/sample
IBLStreams [35]	Approach: Prototype selection; Editing	Adaptation strategy: Mixed*
	Cardinality: Variable	Model management: Single classifier
	Direction: Mixed	Data management: Single point
	Criterion: Filter: Usefulness	Memory.Forgetting: Delete old/poorly-predicting/sample
COMPOSE [36]	Approach: Prototype selection; Editing	Adaptation strategy: Evolving
	Cardinality: Variable	Model management: Single classifier
	Direction: Mixed	Data management: Single point
	Criterion: Filter; Close to same class	Memory.Forgetting: Delete objects far from class centres
Prediction Error	Approach: Prototype selection; Hybrid	Adaptation strategy: Evolving
Context Switching	Cardinality: Variable	Model management: Single classifier
PECS [37]	Direction: Mixed	Data management: Single point
	Criterion: Filter; Usefulness	Memory.Forgetting: Delete poorly-predicting objects
SyncStream [38]	Approach: Mixed selection and generation	Adaptation strategy: Mixed*
	Cardinality: Variable	Model management: Single classifier
	Direction: Mixed	Data management: Window
	Criterion: Filter; Usefulness	Memory: old *concepts* deleted; poorly-predicting objects deleted
Adaptive NN Classification	Approach: Prototype generation	Adaptation strategy: Evolving
Algorithm for Data-streams	Cardinality: Fixed	Model management: Ensemble
ANNCAD [39]	Criterion: N/A (prototype locations fixed)	Data management: Single point
		Memory.Forgetting: Decay function
Learning Vector Quantization	Approach: Prototype generation	Adaptation strategy: Evolving
LVQ [40]	Cardinality: Fixed	Model management: Single classifier
	Criterion: N/A	Data management: Single point
		Memory: N/A
Email Classification	Approach: Prototype selection; Hybrid	Adaptation strategy: Evolving
Using Examples	Cardinality: Variable	Model management: Single classifier
ECUE [41]	Direction: Mixed	Data management: Single point
	Criterion: Filter; Usefulness	Memory.Forgetting: Delete poorly-performing and irrelevant
SpamHunting [42]	Approach: Prototype selection; Hybrid	Adaptation strategy: Evolving
	Cardinality: Fixed or Variable	Model management: Single classifier
	Direction: Mixed	Data management: Single point
	Criterion: Filter; Oldest removed	Memory.Window: Fixed or Variable
Algorithm of Lu et al.	Approach: Prototype selection; Hybrid	Adaptation strategy: Mixed*
[32]	Cardinality: Variable	Model management: Single classifier
	Direction: Mixed	Data management: Single point
	Criterion: Filter; Usefulness	Memory.Forgetting: Other

Notes

* Evolving *and* change-detection based

between the prototype methods for adaptive learning, and hence propose a new taxonomy.

Historically, Aha et al.'s Instance-Based Learning Algorithm IB3 was the first prototype selection technique capable of handling concept drift [33]. At nearly the same time, Kohonen developed the Learning Vector Quantization technique [40], a seminal example of prototype generation.

ANNCAD [39] is not presented as a prototype method, but the approach of discretising the feature space is equivalent to using the centre of each hyper-rectangular cell of the discretisation as a generated prototype. Note, though, that ANNCAD does not perform a true nearest-neighbour classification: to reduce computational demands, only those neighbours are queried which are also nearby branches in the tree structure which describes the various resolutions of the discretisation.

PECS [37] takes a similar approach to IB3. The main differences, apart from the statistical methods, are that PECS immediately includes all new examples in the prototype set, and that it never truly deletes examples, only inactivates them (PECS is therefore not strictly a streaming algorithm, although it can handle concept drift).

More recent approaches, especially those from the "Case-Based Reasoning" community [41], [42], [32], but also others such as COMPOSE [36], have been presented in the form of "systems" or "frameworks" rather than algorithms, posing a further challenge to taxonomic studies.

Table I shows our collection of data editing methods suitable for streaming data with concept drift along with their classification according to the two proposed taxonomies. This table shows a great deal of repetition from the description of one algorithm to another. We argue this is indicative that the current taxonomies – for general prototype methods on the one hand, and general adaptive learning methods on the other hand – are inadequate to categorise the emerging family of prototype methods for adaptive learning.

There are some *a priori* reasons why this might be so. For example, an algorithm for incremental data editing for static data, which takes a strictly decremental or incremental direction when forming its prototype set, cannot be applied without modification to streaming data; the set would tend either towards emptiness or to unbounded growth. So we see that all entries in table I either have "mixed" direction, or a prototype set of fixed cardinality. Similarly, the "wrapper" criterion for evaluating a prototype set is impossible for the streaming case, since it requires the evaluation of the entire training set [9]. So all algorithms in the table for which prototypes are evaluated use some method which must, in the existing taxonomy, be unhelpfully described as a "filter" in every case.

In a future taxonomy, it would be most desirable for methods with variable-cardinality prototype sets to have a more nuanced description of how this set is evolved. In particular, for the streaming case, there are likely to be different rules for adding new prototypes to the set and removing old ones from it. Some algorithms might add all new examples to the set to start with, while others might only add those misclassified by the existing set. Prototypes are often removed on the grounds of poor predictive performance. This is a very different perspective from the non-streaming case, where the issue was thought of as an evaluation of a potential prototype set, not as how best to incrementally evolve such a set.

This issue of describing the evolution of the prototype set gives rise to an equivalent problem in the "Adaptive Learning" taxonomy. Few algorithms are so naïve as to use a simple window for their adaptation; table I therefore lists many algorithms as having a "forgetting mechanism"; but it is the distinctions between these mechanisms which are needed to describe the prototype methods.

Based on these lines of thinking, our suggestion towards the development of a taxonomy for prototype methods for adaptive learning is shown in Figure 4.

In this taxonomy, we have removed the explicit "Cardinality" attribute, because we find that algorithms based on prototype selection strongly tend to have variable cardinality. From the sub-categories of the "Variable" cardinality in Figure 1, only "Mixed" is suitable for streaming data editing because the prototype set cardinality must be kept bounded while the set is updated indefinitely.

The "Data Management" category in the adaptive learning taxonomy made a distinction between the availability of a single data point and a sliding window of such points. As this is not useful for lazy learners, this category has been repurposed in the new taxonomy to make a distinction between algorithms which operate in a truly online manner on a single new data point at a time, and those for which the data stream is regarded as arriving in batches. This latter distinction is separate from the issue of which prototypes are "remembered" and for how long, unlike the former distinction.

The "Adaptation strategy" question from the Adaptive Learning taxonomy has been rephrased as a question of "Change detection". This is because a prototype method which continues to evolve its prototype set in response to an incoming data stream must thereby necessarily adapt passively to any concept drift in the stream. This is the reason for all the "mixed" adaptation strategies in table I: the distinction between passive and active change adaptation is a false dichotomy in the case of prototype methods.

The Condensing/Error-Editing distinction, which is a historical feature of the prototype selection methods, is also less important for the case of streaming data. Hybrid methods overshadow both individual categories. For the offline case, such hybrid methods were often the result of applying a wrapper approach where the overall accuracy of the classifier guides the construction of the prototype set. In the context of streaming data, the wrapper approach would be time-consuming and applicable only to batches of data. This has invited two-stage approaches which are explicitly composed of an error-editing stage first (called also "competence enhancement") followed by a condensing stage (called also "competence preservation") [32], [41]. Both stages use filter criteria. In our taxonomy, we do not distinguish explicitly between condensing/editing and hybrid methods. Instead, we consider

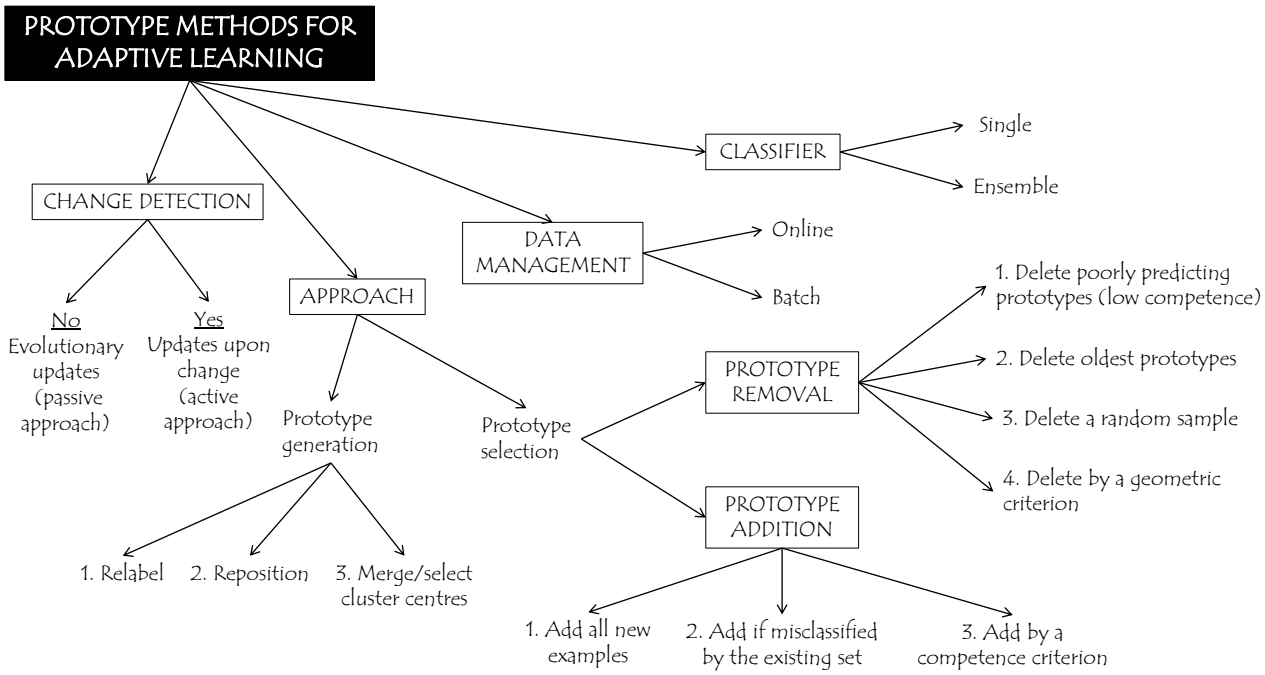


Fig. 4. A taxonomy of prototype methods for adaptive learning. The nodes in boxes show properties that should be specified, and their values are chosen among the leaves of the respective sub-tree.

specific mechanisms by which individual prototypes might be added to, removed from, or evolved within the prototype set. We believe this is a more natural way of considering the composition of the prototype set, given that in the streaming case the prototype set must continue to be evolved indefinitely. We believe also that the "Prototype addition" and "Prototype removal" categories offer great adaptability for new methods: a wealth of future algorithms might be described by appropriate new labels in these categories.

Table II shows proposed classification under the new taxonomy of the methods presented in Table I. Compared with table I, much less repetition between descriptions of algorithms is seen in table II. The only instance of strong repetition is between the descriptions of INL-DS and IBLStreams, which are genuinely very closely-related algorithms. This may indicate that our proposed taxonomy discriminates better among these algorithms than the existing alternatives.

V. Conclusion

Key aspects of the existing taxonomies are inadequate to describe nearest-neighbour algorithms for streaming data subject to concept drift. In particular, the old distinctions between editing/condensing/hybrid prototype-selection methods, and between wrapper or filter evaluation mechanisms, are clearly unsuitable for streaming algorithms and must be replaced. The same holds for the incremental/decremental/mixed/batch categorisation of data editing methods. Only the mixed strategy (containing both incremental and decremental parts) is suitable for streaming data. On the other hand, the adaptive learning field is not specifically focused on prototype methods. There-

TABLE II
Prototype selection methods for streaming data with concept drift represented within the taxonomy in Figure 4.

Method	Change detection	Approach			Data management	Classifier
		PG	PS			
			+	−		
IB3	N		2	1	O	S
INL-DS	Y		1	1, 3	O	S
IBLStreams	Y		1	1, 3	O	S
COMPOSE	N		1	4	B	S
PECS	N		1	1	O	S
SyncStream	Y	3	1	1	O	S
ANNCAD	N	1			O	E
LVQ	N	2			O	S
ECUE	N		3	1	B	S
SpamHunting	N		1	2	O	S
Lu et al.	Y		3	1	B	S

Notes:

PG : Prototype generation: (1) Relabel; (2) Reposition; (3) Merge or select cluster centres.

PS $(+)$: Prototype selection (Prototype Addition): (1) Add all; (2) Add misclassified; (3) Add by a competence criterion.

PS $(-)$: Prototype selection (Prototype Removal): (1) Delete poorly predicting; (2) Delete oldest; (3) Delete a random sample; (4) Delete by a geometric criterion.

O/B : Online / Batch

S/E : Single classifier / Ensemble

fore, taking inspiration from both fields, we have proposed a taxonomy which can describe the mechanisms by which prototype sets are evolved in streaming algorithms.

Future work includes carrying out a comprehensive review of the existing data editing methods for streaming data, and refining the taxonomy accordingly. An experimental evaluation on simulated and real data with different types of concept drift may shed light on the relative merits of the different categories and approaches.

ACKNOWLEDGMENT

This work was done under project PR-2015-188 funded by The Leverhulme Trust, UK.

REFERENCES

[1] L. Devroye, L. Györfi, and G. Lugosi, *A probabilistic theory of pattern recognition*. New York: Springer-Verlag, 1996.

[2] E. Fix and J. L. Hodges Jr, "Discriminatory analysis-nonparametric discrimination: consistency properties," DTIC Document, Tech. Rep., 1951.

[3] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *Information Theory, IEEE Transactions on*, vol. 13, no. 1, pp. 21–27, 1967.

[4] P. Hart, "The condensed nearest neighbor rule," *IEEE Transactions on Information Theory*, vol. 16, pp. 515–516, 1968.

[5] D. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-2, pp. 408–421, 1972.

[6] B. V. Dasarathy, *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. Los Alamitos, California: IEEE Computer Society Press, 1990.

[7] D. R. Wilson and T. R. Martinez, "Reduction techniques for instance-based learning algorithms," *Machine learning*, vol. 38, no. 3, pp. 257–286, 2000.

[8] H. Brighton and C. Mellish, "Advances in instance selection for instance-based learning algorithms," *Data mining and knowledge discovery*, vol. 6, no. 2, pp. 153–172, 2002.

[9] S. Garcia, J. Derrac, J. R. Cano, and F. Herrera, "Prototype selection for nearest neighbor classification: Taxonomy and empirical study," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 3, pp. 417–435, 2012.

[10] I. Triguero, J. Derrac, S. Garcia, and F. Herrera, "A taxonomy and experimental study on prototype generation for nearest neighbor classification," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 42, no. 1, pp. 86–100, 2012.

[11] J. G. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N. V. Chawla, and F. Herrera, "A unifying view on dataset shift in classification," *Pattern Recognition*, vol. 45, no. 1, pp. 521–530, 2012.

[12] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, p. 44, 2014.

[13] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in non-stationary environments: A survey," *IEEE Computational Intelligence Magazine*, vol. 10, no. 4, pp. 12–25, 2015.

[14] I. Žliobaitė, "Learning under concept drift: an overview," *arXiv preprint arXiv:1010.4784*, 2010.

[15] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg, "Top 10 algorithms in data mining," *Knowledge and Information Systems*, vol. 14, no. 1, pp. 1–37, 2008.

[16] F. J. Ferri, "Combining adaptive vector quantization and prototype selection techniques to improve nearest neighbour classifiers," *Kybernetika*, vol. 34, no. 4, pp. 405–410, 1998.

[17] S. Ferrandiz and M. Boullé, "Bayesian instance selection for the nearest neighbor rule," *Machine learning*, vol. 81, no. 3, pp. 229–256, 2010.

[18] J. R. Cano, F. Herrera, and M. Lozano, "Using evolutionary algorithms as instance selection for data reduction in kdd: an experimental study," *Evolutionary Computation, IEEE Transactions on*, vol. 7, no. 6, pp. 561–575, 2003.

[19] Z.-Y. Chen, C.-F. Tsai, W. Eberle, W.-C. Lin, and S.-W. Ke, "Instance selection by genetic-based biological algorithm," *Soft Computing*, pp. 1–14, 2015.

[20] T. Nakashima and H. Ishibuchi, "Ga-based approaches for finding the minimum reference set for nearest neighbor classification," in *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*. IEEE, 1998, pp. 709–714.

[21] G. P. Figueredo, N. F. F. Ebecken, D. A. Augusto, and H. J. Barbosa, "An immune-inspired instance selection mechanism for supervised classification," *Memetic Computing*, vol. 4, no. 2, pp. 135–147, 2012.

[22] A. Miloud-Aouidate and A. R. Baba-Ali, "Ant colony prototype reduction algorithm for knn classification," in *Computational Science and Engineering (CSE), 2012 IEEE 15th International Conference on*. IEEE, 2012, pp. 289–294.

[23] L. Kuncheva, "Editing for the k-nearest neighbors rule by a genetic algorithm," vol. 16, pp. 809–814, 1995.

[24] J. Derrac, I. Triguero, S. García, and F. Herrera, "Integrating instance selection, instance weighting, and feature weighting for nearest neighbor classifiers by coevolutionary algorithms," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 42, no. 5, pp. 1383–1397, 2012.

[25] C.-F. Tsai, W. Eberle, and C.-Y. Chu, "Genetic algorithms in feature and instance selection," *Knowledge-Based Systems*, vol. 39, pp. 240–247, 2013.

[26] N. García-Pedrajas, A. de Haro-García, and J. Pérez-Rodríguez, "A scalable approach to simultaneous evolutionary instance and feature selection," *Information Sciences*, vol. 228, pp. 150–174, 2013.

[27] I. Triguero, D. Peralta, J. Bacardit, S. García, and F. Herrera, "Mrpr: A mapreduce solution for prototype reduction in big data classification," *Neurocomputing*, vol. 150, pp. 331–345, 2015.

[28] J. Calvo-Zaragoza, J. J. Valero-Mas, and J. R. Rico-Juan, "Improving knn multi-label classification in prototype selection scenarios using class proposals," *Pattern Recognition*, vol. 48, no. 5, pp. 1608–1622, 2015.

[29] A. de Haro-García and N. García-Pedrajas, "A divide-and-conquer recursive approach for scaling up instance selection algorithms," *Data Mining and Knowledge Discovery*, vol. 18, no. 3, pp. 392–418, 2009.

[30] J. Bien and R. Tibshirani, "Prototype selection for interpretable classification," *The Annals of Applied Statistics*, pp. 2403–2424, 2011.

[31] M. Maloof and R. Michalski, "Incremental learning with partial instance memory," *Artificial intelligence*, vol. 154, no. 1, pp. 95–126, 2004.

[32] N. Lu, J. Lu, G. Zhang, and R. L. de Mantaras, "A concept drift-tolerant case-base editing technique," *Artificial Intelligence*, vol. 230, pp. 108–133, 2016.

[33] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Machine learning*, vol. 6, no. 1, pp. 37–66, 1991.

[34] J. Beringer and E. Hüllermeier, "Efficient instance-based learning on data streams," *Intelligent Data Analysis*, vol. 11, no. 6, pp. 627–650, 2007.

[35] A. Shaker and E. Hüllermeier, "IBLStreams: a system for instance-based classification and regression on data streams," *Evolving Systems*, vol. 3, no. 4, pp. 235–249, 2012.

[36] K. B. Dyer, R. Capo, and R. Polikar, "COMPOSE: A semisupervised learning framework for initially labeled nonstationary streaming data," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 25, no. 1, pp. 12–26, 2014.

[37] M. Salganicoff, "Tolerating concept and sampling shift in lazy learning using prediction error context switching," *Artificial Intelligence Review*, vol. 11, no. 1-5, pp. 133–155, 1997.

[38] J. Shao, Z. Ahmadi, and S. Kramer, "Prototype-based learning on concept-drifting data streams," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 412–421.

[39] Y.-N. Law and C. Zaniolo, "An adaptive nearest neighbor classification algorithm for data streams," in *Knowledge Discovery in Databases: PKDD 2005*. Springer, 2005, pp. 108–120.

[40] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[41] S. J. Delany, P. Cunningham, A. Tsymbal, and L. Coyle, "A case-based technique for tracking concept drift in spam filtering," vol. 18, pp. 187–195, 2005.

[42] F. Fdez-Riverola, E. Iglesias, F. Díaz, J. Méndez, and J. Corchado, "Applying lazy learning algorithms to tackle concept drift in spam filtering," *Expert Systems with Applications*, vol. 33, pp. 36–48, 2007.