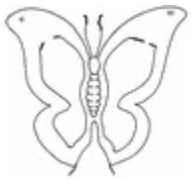
Workshop AIL
Active and Incremental Learning

August 27, 2012, Montpellier, France

ECAI 2012 – 20TH European Conference on Artificial Intelligence



Vincent Lemaire[1] , Jean-Charles Lamirel[2] , Pascal Cuxac[3]

[1] Orange Labs, Lannion, vincent.lemaire@orange.com

[2] TALARIS-LORIA, Vandoeuvre les Nancy, jean-charles.lamirel@loria.fr

[3] INIST-CNRS R&D, Vandoeuvre les Nancy, pascal.cuxac@inist.fr

Workshop AIL :
Active and Incremental Learning

Chairs:

Vincent Lemaire[1] and Jean-Charles Lamirel[2] and Pascal Cuxac[3]

[1] Orange Labs, Lannion, vincent.lemaire@orange.com

[2] TALARIS-LORIA, Vandoeuvre les Nancy, jean-charles.lamirel@loria.fr

[3] INIST-CNRS R&D, Vandoeuvre les Nancy, pascal.cuxac@inist.fr

This workshop aims to offer a meeting opportunity for academics and industry-related researchers, belonging to the various communities of Computational Intelligence, Machine Learning, Experimental Design and Data Mining to discuss new areas of active and incremental learning, and to bridge the gap between data acquisition or experimentation and model building. How active sampling, incremental learning and data acquisition, can contribute towards the design and modelling of highly intelligent autonomous learning systems?

Example 1: Imagine a simple problem which is to supervise a node in a telecommunication network using a binary classifier. The first step is to train incrementally (and perhaps using active learning) the classifier using a data stream and then to maintain this classifier without any "human" interaction. The classifier has to be completely autonomous. Papers which will propose new elements to go to this direction will be in the scope of the workshop.

Example 2: The diachronic analysis of a large and evolving information source for detecting stable, emerging or vanishing topics could be done with the help of clustering methods. However, to provide satisfying detection accuracy, the exploited methods must altogether deal with the plasticity stability dilemma and be able to manage an evolving data description space. Paper that will propose efficient solutions that respect these constraints will be also in the scope of the workshop.

Topics of interest to the workshop include (but are not limited to):

- Experimental Design
- Active Learning
- Autonomous Learning
- Incremental Learning
- Autonomous intelligent systems
- On-line learning
- Novelty and Drift Detection
- Adaptive clustering methods
- Case Studies of AIL Learning (railway stations, airport, bank branches, etc)
- Autonomous Robots

Organizing committee

- Mahmoud Abou-Nasr (Ford Motor Company,USA),
- Shadi Al Shehabi (Allepo University,Aleppo,Syria),
- Cesare Alippi (Politecnico di Milano,Milano,Italia),
- Tomas Arredondo (U.T.F.S.M. Valparaíso,Chile),
- Vassilis Athitsos (University of Texas at Arlington,Texas,USA),
- Albert Bifet,(University of Waikato,Hamilton,New Zealand),
- Alexis Bondu (EDF R&D, France),
- Fazli Can (Bilkent University,Ankara,Turkey),
- Laurent Candillier (Wikio Group / Nomao, France),
- Chaomei Chen (Drexel University,Philadelphia,USA),
- Jung-Chen Chiang (NCKU,Tainan,Taiwan),
- Fabrice Clérot (Orange Labs, France),
- Pascal Cuxac (INIST-CNRS,Vandoeuvre les Nancy, France),
- Abdoulaye B. Diallo (UQAM,Montreal,Canada),
- Gideon Dror (Academic college of Tel-Aviv Yaffo,Israel),
- Hugo Jair Escalante, (National Institute of Astrophysics, Optics and Electronics, Mexico),
- Françoise Fessant (Orange Labs, France),
- Dominic Forest (EBSI Univ. Montreal,Montreal,Canada),
- José García-Rodríguez (University of Alicante,Spain),
- Wolfgang Glanzel (KU Leuven, Leuven, Belgia),
- Jing-Hao He (University of Rhode Island,Kingston,USA),
- Pascale Kuntz-Cosperec (Polytech'Nantes, France),
- Jean-Charles Lamirel (Talaris Loria, France),
- Vincent Lemaire (Orange Labs, France),
- Bin Li (University of Technology (UTS),Sydney,Australia),
- Gaelle Losli (Polytech Clermont-Ferrand, France),
- Florin Popescu (Fraunhofer Institute,Berlin,Germany),
- Vikram Pudi (IIIT Hydeabad,Hyderabad,India),
- Manuel Roveri (Politecnico di Milano,Milano,Italia),
- Christophe Salperwyck (Orange Labs, France),
- Danny Silver (University of Acadia,Wolfville,Canada),
- Tony C. Smith (University of Waikato,Hamilton,New Zealand)
- Alexander Statnikov (New York University,New-York,USA),
- Dan Tamir (Texas State University,San Marcos,USA),
- Fabien Torre (Université Lille 3, France),
- Ioannis Tsamardinos (University of Crete,Greece),
- Vincent Tseng (NCKU,Tainan,Taiwan),
- Zhi-Hua Zhou (Nanjing University,Nanjing,China),
- Xingquan Zhu (University of Technology (UTS),Sydney,Australia),
- Djamel Zighed (Louis Lumiere University,Lyon, France).

Sponsor

www.orange.com

Invited Talk :

Learning models in nonstationary environments: the Just-In-Time approach

Manuel Roveri

roveri@elet.polimi.it

Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milano, Italy

Abstract:

Most machine learning techniques assume, either explicitly or implicitly, that the data-generating process is stationary. This assumption guarantees that the model learnt during the initial training phase remains valid over time and that its performance is in line with our expectations. Unfortunately, this assumption does not truly hold in the real world representing, in many cases, a simplistic approximation of the reality.

The talk will describe the Just-In-Time (JIT) approach that is a flexible tool implementing the detection/adaptation paradigm to cope with evolving processes. Solutions following this approach improve the knowledge about the model in stationary conditions by exploiting additional information coming from the field during the operational life. Differently, in nonstationary conditions, as soon as a change in the data-generating process is detected, the learnt model is discarded and a suitable one activated to keep the performance. As a valuable and challenging application of the proposed approach, JIT classifiers for concept drift will be detailed and discussed.

Incremental Decision Tree based on order statistics

Christophe Salperwyck[1] **and Vincent Lemaire**[2]

Abstract. New application domains generate data which are not persistent anymore but volatile: network management, web profile modeling... These data arrive quickly, massively and are visible just once. Thus they necessarily have to be learnt according to their arrival orders. For classification problems online decision trees are known to perform well and are widely used on streaming data. In this paper, we propose a new decision tree method based on order statistics. The construction of an online tree usually needs summaries in the leaves. Our solution uses bounded error quantiles summaries. A robust and performing discretization or grouping method uses these summaries to provide, at the same time, a criterion to find the best split and better density estimations. This estimation is then used to build a naïve Bayes classifier in the leaves to improve the prediction in the early learning stage.

1 Introduction

Learning machines have shown their ability to deal with huge volumetry on real problems [15, 9]. Nevertheless most of the works were realized for data analysis on homogeneous and stationary data. Learning machines usually use data sets with fixed sizes and produce static models.

New application fields for data mining emerge in which data are not anymore persistent data table but rather "temporary" data. These data are called streaming data. Among these domains one finds: management of telecommunication networks, user modeling in a social network, web mining. One of the technical challenges is to design algorithms able to handle these new application constraints. As data arrive quickly and are visible only once, it is necessary to learn them as they arrive. Incremental learning appears as a natural solution to streaming problems.

Among the methods in incremental learning, models based on decision trees inspired by the algorithm "Very Fast Decision Tree" (VFDT) [8] are widely used. The tree construction is incremental and leaves are transformed into nodes as examples arrive. The new examples go down into the tree and are inserted variable by variable in a summary. A criterion (Gini or Entropy) uses this summary to find the cut points to transform a leaf into a node. The tree prediction can be improved by the addition of a local model in each leaf as in VFDTc [12].

The error rate of this kind of algorithm is more important in the early learning stage than a batch algorithm as C4.5. But having learnt several hundreds of thousand examples, this error rate becomes lower than C4.5 since C4.5 is not able to deal with millions of examples and thus has to use only a part of the available information.

[1] Orange Labs, Lannion, France and LIFL, Université de Lille 3, Villeneuve d'Ascq, France, email: christophe.salperwyck@orange.com
[2] Orange Labs, Lannion, France, email: vincent.lemaire@orange.com

In this article we focused on the construction of online decision trees. Section 2 presents existing approaches in terms of: split criterion, summaries in leaves, and local models. Our approach, based on order statistics, is detailed in the 3rd section. The experimental part which compares our approach with existing ones is in section 4. The last section concludes this article.

2 Previous works

Constructing an online decision tree is based on three main choices. In the first place, it is impossible in the data stream context, potentially of infinite size, to keep all the examples. The use of data summaries of limited size is necessary to be able to control the tree memory consumption. The fact that decisions are local to the leaf justifies storing summaries in each leaf. Secondly, cut points are chosen by the evaluation in every leaf of a criterion (generally the Gini or the entropy criterion). This choice being a definitive action has to be robust and made with a certain confidence. Finally before a split occurs, the available information in leaves is not used. Using a local model in each leaf allows exploiting this information to improve the global prediction of the tree. Figure 1 illustrates a decision tree and its three main components.

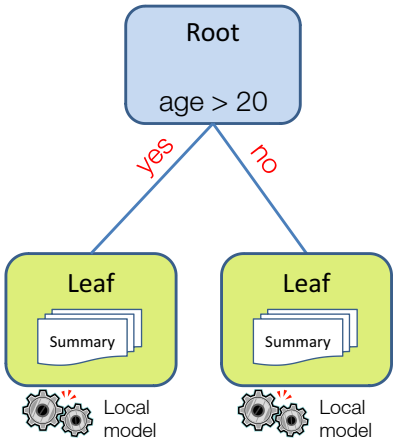


Figure 1. Main components of an online decision tree.

This section presents the different approaches used in the literature to answer the three main key points mentioned above. The quality of a decision tree depends on: (i) the summaries in the leaves, (ii) the split criterion, (iii) the local model.

2.1 Summaries in the leaves

The purpose of data summaries in the leaves is to memorize stream characteristics. This summary is used to find the "best" cut point to transform a leaf into a node and also to build a local model in the leaf. In certain application domains, as for example the management of a network of telecommunication or energy, the stream is potentially of infinite size. The generated tree de facto possesses a large number of decision nodes but the available memory is often of limited size. Therefore these summaries need to have a low memory footprint with low errors and address the precision / memory tradeoff.

The paragraphs below present several summaries. Numerical and categorical attributes are generally handled by means of different summary techniques.

2.1.1 Numerical attributes

Gama and Pinto propose a summary for numerical attributes called Partition Incremental Discretization - PiD [11] which is partially incremental. This solution is based on two levels. Level 1 realizes a first discretization where the counts are stored by interval. This first level implements an incremental algorithm which combines the methods "Equal Frequency" and "Equal Width" and has to contain more intervals than the level 2. Second level uses level 1 discretization to make the final discretization, which can be of several types: "Equal Frequency", "Equal Width", K-means, Recursive entropy discretization, Proportional discretization. The memory consumption of this method depends mainly on the number of intervals in the first level. The second level is not incremental.

Pfahringer et al. [21] carried out a study on summaries for numerical attributes; their study is dedicated to trees using the Hoeffding bound. They tested the following summaries methods:

- Very Fast Machine Learning (VFML): this very simple method takes the k first values of the stream and uses them to build $k+1$ intervals. The next values are aggregated in the closest interval defined by the first values. The memory consumption depends on the parameter k. This method comes from the source code [18] provided by the authors of VFDT: Domingos and Hulten.
- Gaussian approximation (GA): the data distribution is supposed to be a normal law. The purpose is to estimate the three parameters of the law which define this Gaussian: the average, the standard deviation (or the variance) and the number of elements. These three values can be stored incrementally making this method perfectly adapted to an online use. This approximation is chosen by Kirkby [19] in all his experiments. The memory consumption is constant independently of the nature of the observed stream.
- Exhaustive binary trees (EBT): Gama et al. use this method for their VFDTc tree [12]. A binary search tree, for each numerical variable, is built incrementally. This tree also keeps in each node the counts of values smaller and bigger than the cut point. This structure allows an immediate access to the counts on both sides of a cut point. The tree memory consumption depends on the number of different values arriving in the leaf.
- GK: this method, proposed by Greenwald and Khanna [14], is a quantiles based summary. It maintains a sorted list of intervals and controls the error on the quantile position (detailed in section 3.1.1). Its memory consumption depends either on the maximal error tolerated on quantiles or on the number of intervals that can be kept.

2.1.2 Categorical attributes

To our knowledge, in most of the publications related to online trees there is no dedicated summary for categorical attributes but just exhaustive counting. It means that for each categorical variable and for each value the number of occurrences is stored. It can be sufficient if the number of values is limited and thus the memory consumed is low. This method linearly depends on the number of different values in the data stream.

2.2 Split criterion

During the construction of the decision tree a split criterion is used to transform a leaf into a node. The objective is to produce the most homogeneous possible groups of individuals regarding the target variable. To transform a leaf into a node it is necessary to determine at the same time on which attribute to cut and on which value (cut point).

In the literature on offline and online decision trees, two main cut criteria are used: Gini in the CART algorithm and the "gain ratio" based on the entropy in C4.5. Those two criteria find the best cut point for an attribute and each of them provides a score. The node is split on the attribute having the best score. This process to transform a leaf into a node is repeated to produce the final decision tree.

The difference in building an online tree and an offline tree comes from the fact that data arrive continuously for the first one. The choice of the attribute to cut is made according to the summary and not on all data. The choice of transforming a leaf into a node, is a definitive action. To make sure that this choice is realized with a certain confidence, Domingos and Hulten suggest in VFDT the use of the Hoeffding bound [16]. This bound brings a guarantee on the choice of the good attribute. The Hoeffding bound was afterwards often used to build online decision tree: VFDTc [12], CVFDT [17], IADEM [22], "ensemble of Hoeffding trees" [19]... The Hoeffding bound is also used in this article to construct the proposed online tree. Detailed description of this bound is presented below.

Hoeffding bound provides an upper bound on the observed mean of a random variable and its true mean. Consider a real-valued random variable r whose range is R. Suppose we have made n independent observations of this variable, and computed its mean $\bar{r}$. The Hoeffding bound states that, with probability $1-\delta$, the true mean of the variable is at least $\bar{r}-\epsilon$, where $\epsilon = \sqrt{\frac{R^2}{2n} ln(\frac{1}{\delta})}$. The interest of this bound is that it depends only on: i) the range R, ii) the number of observations n, iii) the desired confidence δ.

This bound guarantees the choice (with a probability $1-\delta$) of the attribute to cut. The bound is applied on the average of an evaluation split criterion G. The best attribute a is definitively considered better than the best second attribute b if $\bar{G}_a - \bar{G}_b > \epsilon$.

2.3 Local model

Gama et al. [12] observed empirically that 100 to 1000 examples are needed before transforming a leaf into a node. These examples in leaves are not used to improve the global model as long as the leaf is not transformed into a node. They suggest using these examples by adding in every leaf a local model (called "functional tree leaves" by Gama et al.). The reader can note that such kind of technique, which uses local models positioned in leaves, exists for a long time. For example the algorithm NBTree [20] uses a decision tree with a naïve Bayes classifier in leaves.

A good local model for online decision trees has to consume a small amount of memory, be fast to build and be fast to return a prediction. It has to be in adequacy with the summaries built in leaves. A good ability to predict with a small number of examples is required because summaries in leaves can be based on few examples. A study [24] on the speed (in number of examples) of different classifiers shows that the forests of tree and the naïve Bayes classifier are classifiers which require few examples to learn. Among those two classifiers only the naïve Bayes classifier respects at best the conditions required by the online construction. Indeed, it does not require additional memory if the summary returns a density estimation by interval (this is the case for all the summaries presented previously). Furthermore it is fast to elaborate and has a low algorithmic complexity to predict. This classifier was also used in VFDTc and improved the prediction on several benchmark datasets [12].

3 Our approach

This paper introduces current works on the construction of online decision trees based on order statistics. For summaries we choose methods based on order statistics and addressing the precision / memory tradeoff. For the criterion, the choice turns to the MODL [5] method which finds Bayes optimal cut points with order statistics. The MODL approach also provides robust density estimation that can be used by a local model. In our case the naïve Bayes classifier is chosen.

3.1 Summaries in the leaves

The summaries used in the proposed approach have at the same time a fixed memory consumption and strong guarantees on the error over the counts. These summaries are described below.

3.1.1 Numerical attributes: quantiles summaries (GK)

Quantiles provides order statistics on the data. The ϕ-quantile, with $\phi \in [0, 1]$ is defined as the element in the position $\lceil \phi N \rceil$ on a sorted list of N values. ϵ is the maximum error on the position of the element: an element is an ϵ approximation of a $\phi - quantile$ if its rank is between $\lceil (\phi - \epsilon)\, N \rceil$ and $\lceil (\phi + \epsilon)\, N \rceil$. It corresponds to an "Equal Frequency" discretization; the number of quantiles being in that case the number of intervals.

The GK quantiles summary [14] is an algorithm to compute quantiles using a memory of $O(\frac{1}{\epsilon} log(\epsilon N))$ in the worst case. This method does not need to know the size of the data in advance and is insensitive to the arrival order of the examples. The algorithm can be configured either with the number of quantiles or with a bound on the error. Its internal structure is based on a list of tuples $< v_i, g_i, \Delta_i >$ where :

- v_i is a value of an explanatory variable of the data stream
- g_i corresponds to the number of values between v_{i-1} and v_i
- Δ_i is the maximal error on g_i

Some insights about the choice of this summary are given in [23]. This method is studied in [21] and implemented with a summary per class and per attribute. It is evaluated as being less successful than the GA or VFML methods. However we adapted the GK summary to store directly the class counts in tuples. Therefore just a summary per attribute is needed and not a summary per class and per attribute. Experimentally this modification improves the quality of prediction (due to the lack of place the experiments related to this point are not presented in this article).

3.1.2 Categorical attributes: Count-min Sketch (CMS)

The purpose of the Count-min Sketch [7] is to find the top-k most frequent values in a data stream with a maximal error ϵ on their counts. A counting matrix of size $t \times b$ is used for the storage. This method uses t hash functions h_i in $\{1, \ldots, b\}$ which select the cell in the matrix to increment: $\forall i = 1, \ldots, t \quad h_i(x) \leftarrow h_i(x) + 1$.

The values for t and b is computed by means of two parameters δ and ϵ. To estimate the frequency $\hat{f}$ of a value with an error inferior to ϵn and a probability of at least $1 - \delta$ then it is necessary to take $t = log\frac{1}{\delta}$ and $b = O(\frac{1}{\epsilon})$. The frequency of a value v is estimated by the minimum of $h_i(x)$: $\hat{f} = \underset{i}{argmin}(h_i(x))$.

The Count-min Sketch is adapted to store class counts. Figure 2 presents this adaptation.

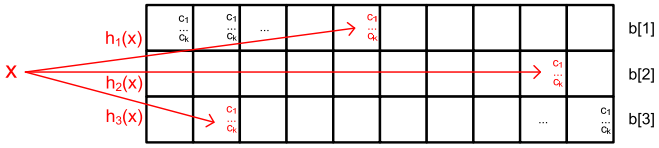


Figure 2. Count-min Sketch adapted for classification.

Using the Count-min Sketch is relevant when the number of different values is large. In the opposite case a simple counting is better as it neither introduces errors nor consumes a large amount of memory. Some other methods with guarantee could be used as well [6].

3.2 Criterion

To be coherent with our summaries (GK and CMS), the MODL criterion based on order statistics is chosen to find cuts and groups respectively for a numerical and a categorical variable. The MODL approach, designed for discretization and value groupings, also returns the quality of a cut or a grouping. This indication l, named 'level' ($l \in [0, 1]$), corresponds to a compression ratio. It indicates the information contained in a numerical or categorical variable when considering a target variable.

The MODL discretization [5] and grouping [4] are supervised and do not need any parameters. They are based on class counts and evaluate all possible intervals for the numerical variables and groups for the categorical variables. The quality evaluation of the model is based on a Bayesian approach. For numerical variables, its purpose is to find the best discretization parameters: number of intervals, frontiers of the intervals and class distribution in the intervals in a Bayesian sense. For categorical variables the method performs grouping in a similar way.

The tree, proposed in this article, is built online in the same manner as VFDT. It uses the Hoeffding bound but the Entropy Gain is replaced by the MODL criterion. The MODL criterion has an additional advantage because it returns a value of criterion $l > 0$ if and only if the discretization model / grouping model is better than the model which returns the majority class. This property allows an automatic "pre-pruning" of the tree while in VFDT this pruning must be separately implemented. What is more this criterion estimates not only binary cut points but can also estimate many cuts for each attribute, which allows to build trees having nodes with more than two sons.

3.3 Local model: density estimation with a two levels approach based on order statistics

We choose the naïve Bayes classifier as the local model in the leaves for our tree. Naïve Bayes has good performances when it is built with few data and its prediction has low algorithmic complexity. This classifier requires an estimation of the class conditional density. This density is estimated for all intervals or groups of values calculated by the MODL method applied respectively to GK or CMS summaries contained in the leaves. This discretization, applied on the summary, corresponds to the two levels discretization method proposed by Gama with its PiD method (see 2.1.1). However our two levels approach is completely based on order statistics and can treat indifferently numerical or categorical attributes. For numerical variables, we choose as the first level the GK quantiles summary and as the second level the MODL discretization method. For categorical variables, we choose as the first level the CMS summary (or a simple counting) and as the second level the MODL grouping method.

The main advantages of the MODL method are its robustness and the absence of parameters. The MODL approach discretizes a numerical variable in several intervals if and only if this discretization is better (in a Bayesian sense) than the model with a single interval (the approach is similar for grouping). If the most probable model is with an interval it means that the attribute is not informative given the observed data. In that case the majority class is predicted. Kirkby in [19] studies exactly this behavior by comparing Hoeffding trees predicting the majority class in leaves and those having a naïve Bayes predictor in leaves. He observes that sometimes the majority class prediction is better than naïve Bayes. Out of this fact he proposes a method choosing automatically either one model or the other by estimating their error rate on the examples which pass in leaves. As the MODL discretization returns just one interval if a variable is not informative, our two levels approach based on MODL method should predict the majority class as well in that case.

4 Experimentations

This section aims to compare our approach to the existing methods described in section 2. We first present the data streams used, then the compared methods and finally the obtained results.

4.1 Data streams used

In order to have enough data for testing online algorithms, artificial generators have been developed to generate stream containing millions of examples. An overview of these generators is presented in [19] and [10]. For the experiments in this paper the following generators were used:

- Random RBF data: is generated by first creating a random set of centers for each class. Each center is randomly assigned a weight, a central point per attribute, and a standard deviation. To generate new instances, a center is chosen at random taking the weights of each center into consideration. Attribute values are randomly generated and offset from the center, where the overall vector has been scaled so that its length equals a value sampled randomly from the Gaussian distribution of the center. The particular center chosen determines the class of the instance. Random RBF data contains only numeric attributes as it is non-trivial to include nominal values. We used 1000 centers and 50 attributes in our experiments.
- Random Tree: data generated by a randomly constructed decision tree consisting of $rt1$ nominal attributes with $rt2$ values each, $rt3$ numeric attributes, $rt4$ classes, a tree depth of $rt5$, with leaves starting at level $rt6$ and a $rt7$ chance of leaves. The final tree has a certain number of nodes and leaves. Different settings for parameters $rt1, ..., rt6$ generate different data streams. This generator gives preferential treatment to decision tree classifiers. We used the following parameters: $rt1 = 10, rt2 = 5, rt3 = 10, rt4 = 3, rt5 = 5, rt6 = 0.15$.
- Waveform: it produces 21 attributes, all of which include noise. It differentiates between 3 different classes of waves, each of which is generated from a combination of two or three base waves. This generator is based on a normal law and gives a preferential treatment to classifiers which assume that data follow a normal law.
- Function (F6): data generation based on the Function F6 described in the appendix of [1]. The stream contains 6 numerical attributes and 3 categorical attributes and a level noise of 5%.

4.2 Algorithms compared

For our experiments we used the MOA toolbox: Massive Online Analysis [2] developed by the university of Waikato which takes up the VFML library supplied by Hulten and Domingos [18]. This toolbox contains stream generators and many online algorithms with which we wish to compare. We made an extension package[3] for the MOA toolbox with our new summaries, new split criterion and new local model.

All the tested trees come from the same algorithm based on the Hoeffding trees. This algorithm contains three parameters: i) the number of examples to be considered before trying to find a cut point; ii) the confidence in the split regarding the Hoeffding bound and iii) the parameter τ which is used to force the choice between two attributes when the criterion difference is too small so that the Hoeffding bound can not decide between them. The configuration of all the trees versions tested here is the same, namely: $n = 200$, $\delta = 10^{-6}$, $\tau = 0.05$. The same settings were used and described by Kirkby in [19].

The summaries in leaves for categorical attributes do not use, in these experiments, the count-min sketch and the MODL grouping because the datasets used contain few different values. The summary, for the categorical attributes, is thus based on a simple counting. The numerical attributes are summarized either by a Gaussian approximation or by quantiles using the GK method.

The attribute and the cut point selected to transform a leaf into a node are the ones maximizing the MODL criterion and which are non zero ($l > 0$). Only binary splits for numerical attributes are considered. For categorical attributes the criterion is evaluated on the model with a leaf per value.

The tested approaches were either without local model or with a naïve Bayes classifier in every leaf. Table 1 presents a synthetic view of the algorithms presented below:

- **HT: Hoeffding Tree -** This algorithm corresponds to the Hoeffding Tree named VFDT in [8]. The summaries are based on density estimation using a Gaussian by class (estimated as being the most successful in [21]). For numerical attributes 10 binary cuts by class, taken between the minimum and the maximum observed, are estimated. This tree does not possess a classifier in leaves.

[3] Extension available here: `http://chercheurs.lille.inria.fr/salperwy/index.php?id=moa-extension`

- **HTNB: Hoeffding Tree using a naïve Bayes -** This version is identical to the previous one but possesses a naïve Bayes classifier in the leaves. The conditional densities are estimated: i) for numerical variables by means of the Gaussian estimation parametrized by 3 values stored in the summary (μ, σ, n); and ii) for categorical variables by means of the counts per class and value.
- **HTmGKc: Hoeffding Tree using MODL split criterion -** This version corresponds to the version which we described in the section 3. The summary for the numerical variables is based on the GK quantiles summary with tuples containing directly the counts per class. Each variable is summarized by 10 tuples. 10 possible cut points are taken from the summary and are evaluated with the MODL criterion described in section 3.2. This version does not possess local model in leaves.
- **HTmGKcNBm: Hoeffding Tree using MODL split criterion and naïve Bayes classifier -** This classifier is identical to the previous version "HTmGKc" but the leaves contain a naïve Bayes classifier. Conditional estimation densities needed by the classifier come from counts per class per value for categorical variable. To have more robust estimation on numerical variables density estimations are computed on the intervals found by the MODL discretization [5].

Method	Summary	Criterion	Discretization	Local model
HT	Gaussian	Entropy Gain	-	-
HTNB	Gaussian	Entropy Gain	-	NB
HTmGKc	GK 10 tuples	MODL Level	-	-
HTmGKcNBm	GK 10 tuples	MODL Level	MODL on GK	NB

Table 1. Specification of the tested algorithm (NB: naïve Bayes)

4.3 Results

Our experimental results are presented in Figure 3 where a curve is plotted for each data set. The evaluation method is the "hold-out test set" and has been chosen among methods described in [13]. For each generator a stream containing 11 million examples has been created. The 1st million of examples is used as a test set. The remaining 10 millions examples represent the training dataset. Trees accuracies are evaluated every 300,000 examples. In a general way our method behaves well on the tested streams and is competitive compared to the other methods. The MODL split criterion applied on the GK summaries, for numerical variables, and on the counts per class, for categorical variables, is globally better than the Entropy Gain criterion calculated on Gaussian summaries, for numerical variables, and the counts per class, for the categorical variables.

The contribution of the naïve Bayes classifier in leaves is debatable with Gaussian summaries because sometimes the accuracy of the global classifier (the entire tree) is either significantly improved (WaveForm) or significantly degraded (F6). With our two levels summaries, the naïve Bayes classifier improves the accuracy of the entire tree especially in the beginning of training. There is no degradation thanks to the robustness of the MODL approach. It creates intervals only if they contain information. If the variable is not informative no discretization model is proposed. The estimation based on these intervals is then provided to the naïve Bayes classifier.

The density estimation using a Gaussian approximation gives better results on "WaveForm" data streams. This result is not surprising because it seems normal that the naïve Bayes classifier having for density estimation a Gaussian approximation works well on data coming from "WaveForm" generator which uses a Gaussian to generate data.

The behavior of the algorithms in low memory environments was not studied in this article. However there are several techniques that we could apply. One of the simplest methods consists in deleting summaries for the less interesting variables. The MODL level is calculated for all variables and gives a precise indication of the information contained in a variable. Therefore it could be used to rank variables and choose which ones to remove from summaries. A second technique consists in deactivating the less promising leaves and thus limits the development of a part of the tree.

The concept drift (see `http://www.cs.waikato.ac.nz/~abifet/PAKDD2011/`) was not studied in this paper but several techniques for concept drift detection will be developed in future works using the summaries and the MODL approach. One of them will use two summaries: one for the past distribution and one for the current. These two summaries will be used to detect drift using the MODL approach (in a similar way to [3]).

5 Conclusion

This paper constitutes a current work on an incremental method used to build an online decision tree with order statistics. Our approach was presented and compared to the previous works on various points needed to elaborate an online decision tree. The GK quantiles summaries and the CMS summary are used in leaves. The MODL method uses these summaries to provide at the same time a cut criterion / value grouping and a robust density estimation per interval / group. This estimation allows afterwards to build a naïve Bayes classifier in the leaves. The experiments showed that our tree performs well compared to existing methods thanks to the robust density estimation provided by the MODL discretization/grouping. Particularly the local classifier always improves the prediction in the early learning stage. The next steps to confirm the interest of our approach are: (i) experiment more data streams and real dataset, (ii) control memory consumption and (iii) extend the algorithm to handle drifts.

REFERENCES

[1] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami, 'Database mining: A performance perspective', *IEEE Transactions on Knowledge and Data Engineering*, **5**, 914–925, (1993).

[2] Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Richard Kirkby, and Ricard Gavaldà, 'New ensemble methods for evolving data streams', *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09*, 139, (2009).

[3] Alexis Bondu and Marc Boullé, 'A supervised approach for change detection in data streams', in *IJCNN*, pp. 519–526, (2011).

[4] M. Boullé, 'A grouping method for categorical attributes having very large number of values', *Machine Learning and Data Mining in Pattern Recognition*, 228–242, (2005).

[5] Marc Boullé, 'MODL: A Bayes optimal discretization method for continuous attributes', *Machine Learning*, **65**(1), 131–165, (May 2006).

[6] G. Cormode and M. Hadjieleftheriou, 'Finding frequent items in data streams', *Proceedings of the VLDB Endowment*, **1**(2), 1530—-1541, (January 2008).

[7] Graham Cormode and S. Muthukrishnan, 'An improved data stream summary: the count-min sketch and its applications', *Journal of Algorithms*, **55**(1), 58–75, (April 2005).

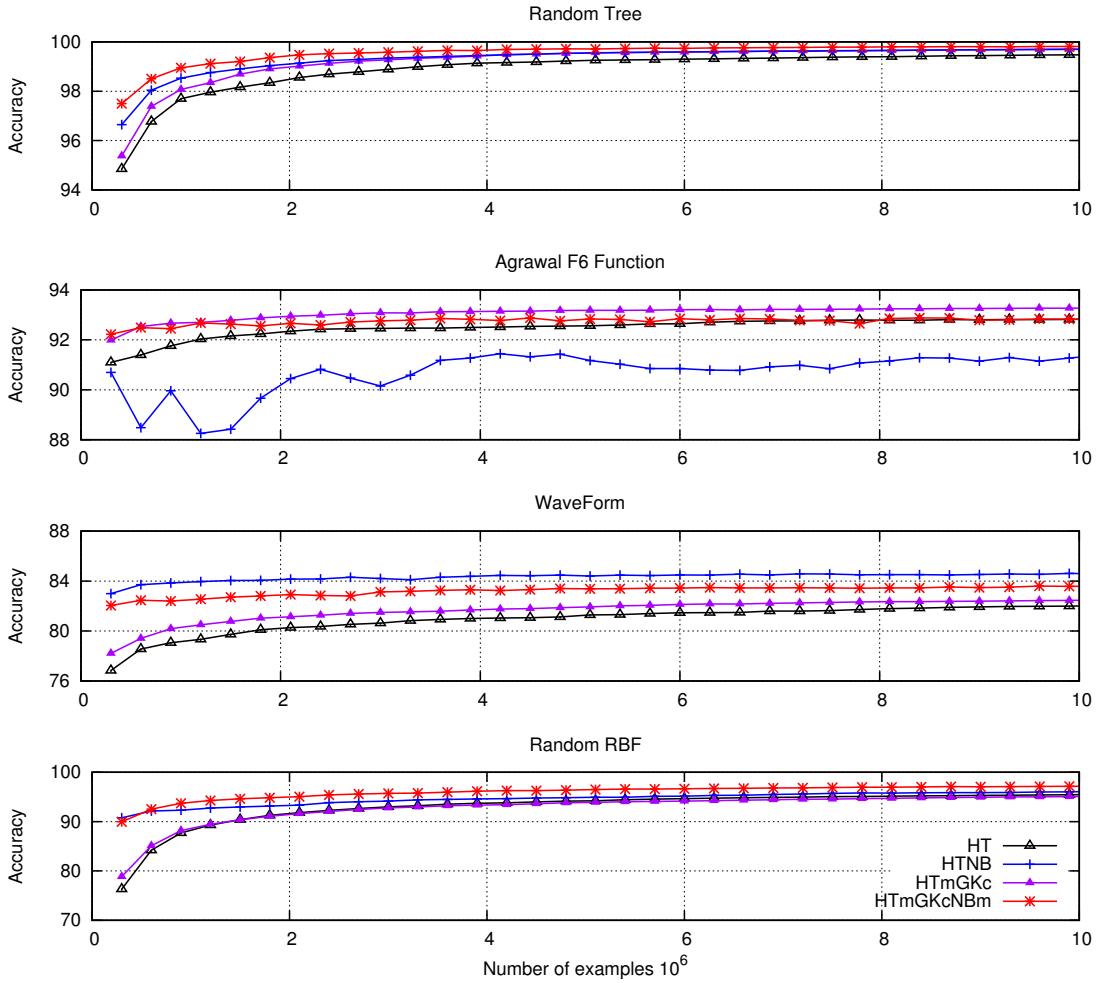


Figure 3. Accuracy versus the number of examples used to train the entire tree.

[8] P. Domingos and G. Hulten, 'Mining high-speed data streams', in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 71–80. ACM New York, NY, USA, (2000).

[9] R. Féraud, M. Boullé, F. Clérot, F. Fessant, and V. Lemaire, 'The orange customer analysis platform', in *Industrial Conference on Data Mining (ICDM)*, pp. 584–594, (2010).

[10] J. Gama, *Knowledge Discovery from Data Streams*, Chapman and Hall/CRC Press, 2010.

[11] J. Gama and Carlos Pinto, 'Discretization from data streams: applications to histograms and data mining', in *Proceedings of the 2006 ACM symposium on Applied computing*, pp. 662–667, (2006).

[12] J. Gama, R. Rocha, and P. Medas, 'Accurate decision trees for mining high-speed data streams', in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 523–528. ACM New York, NY, USA, (2003).

[13] J. Gama, P.P. Rodrigues, and R. Sebastião, 'Evaluating algorithms that learn from data streams', in *Proceedings of the 2009 ACM symposium on Applied Computing*, pp. 1496–1500. ACM New York, (2009).

[14] M. Greenwald and S. Khanna, 'Space-efficient online computation of quantile summaries', *ACM SIGMOD Record*, **30**(2), 58–66, (2001).

[15] I Guyon, V. Lemaire, G. Dror, and D. Vogel, 'Analysis of the kdd cup 2009: Fast scoring on a large orange customer database', *JMLR: Workshop and Conference Proceedings*, **7**, 1–22, (2009).

[16] W Hoeffding, 'Probability inequalities for sums of bounded random variables', *Journal of the American Statistical Association*, (1963).

[17] G. Hulten, L. Spencer, and P. Domingos, 'Mining time-changing data streams', in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 97–106. ACM New York, NY, USA, (2001).

[18] Geoff Hulten and Pedro Domingos, 'VFML – a toolkit for mining high-speed time-changing data streams'. 2003.

[19] Richard Kirkby, *Improving Hoeffding Trees*, Ph.D. dissertation, University of Waikato, 2008.

[20] Ron Kohavi, 'Scaling up the accuracy of naive-Bayes classifiers: A decision-tree hybrid', in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, volume 7. Menlo Park, USA: AAAI Press, (1996).

[21] Bernhard Pfahringer, Geoffrey Holmes, and Richard Kirkby, 'Handling numeric attributes in hoeffding trees', *Advances in Knowledge Discovery and Data Mining*, 296–307, (2008).

[22] G. Ramos-Jimenez, J. del Campo-Avila, and R. Morales-Bueno, 'Incremental algorithm driven by error margins', *Lecture Notes in Computer Science*, **4265**, 358, (2006).

[23] C. Salperwyck and V. Lemaire, 'Incremental discretization for supervised learning', *CLADAG: CLAssification and Data Analysis Group - 8th International Meeting of the Italian Statistical Society*, (2011).

[24] C. Salperwyck and V. Lemaire, 'Learning with few examples: an empirical study on leading classifiers', in *The 2011 International Joint Conference on Neural Networks (IJCNN)*, pp. 1010–1019. IEEE, (2011).

Online Active Constraint Selection For Semi-Supervised Clustering

Caiming Xiong and David Johnson and Jason J. Corso [1]

Abstract.

Due to strong demand for the ability to enforce top-down structure on clustering results, semi-supervised clustering methods using pairwise constraints as *side information* have received increasing attention in recent years. However, most current methods are *passive* in the sense that the side information is provided beforehand and selected randomly. This may lead to the use of constraints that are redundant, unnecessary, or even harmful to the clustering results. To overcome this, we present an active clustering framework which selects pairwise constraints online as clustering proceeds, and propose an online constraint selection method that actively selects pairwise constraints by identifying uncertain nodes in the data. We also propose two novel methods for computing node uncertainty: one global and parametric and the other one local and nonparametric. We evaluate our active constraint selection method with two different semi-supervised clustering algorithms on UCI, digits, gene and image datasets, and achieve results superior to current state of the art active techniques.

1 Introduction

Many semi-supervised clustering methods have been proposed to enforce top-down structure while clustering [3, 5, 11, 15, 16, 23]. These methods allow the user to incorporate pairwise constraints, which may be either must-link (the two points/nodes belong in the same cluster) or cannot-link (the two points/nodes belong in different clusters), on the data as *side information*. These papers have shown that the use of pairwise constraints can significantly improve the correspondence between clusters and semantic labels *when the constraints are selected well*. [7] demonstrates that poorly chosen constraints can lead to worse performance than no constraints at all. Moreover, in real world problems each added constraint represents an additional real world cost, so maximizing the effectiveness of each constraint in order to minimize the total number of constraints needed is an important goal.

Currently, most work in semi-supervised clustering ignores this problem and simply selects a random constraint set (see above cited), but some work has now been done on *active* constraint selection methods [2, 10, 17, 21, 24], which allow semi-supervised clustering algorithms to intelligently select constraints based on the structure of the data and/or intermediate clustering results.

Active selection methods can be stratified according to whether nodes or node-pairs are the primary element on which the process is based. Node-based methods first select nodes of interest, and then query constraints based on those nodes [2,9,17], while those methods that directly seek pair constraints [10, 21, 24], define an uncertainty measure on pairs and iteratively seek the most uncertain pairs during constraint selection.

Both of these current approaches have drawbacks, however. Current node-based methods function by selecting all of their constraints in one long selection phase before clustering. Because of this, they cannot incorporate information from actual clustering results into their decisions, and may thus choose many unnecessary constraints (for instance, constraints regarding points that the algorithm is able to cluster correctly even without side information). In contrast, the pair-based methods choose constraints online based on intermediate clustering results, but due to the nature of the pair selection problem (n^2 possible constraints to rank and select from) have thus far been limited to either binary or small-scale clustering problems.

In this paper, we overcome these limitations and present a node-based constraint selection framework (illustrated in Figure 1) that combines a preprocessing stage with an online, iterative process that selects uncertain nodes based on intermediate clustering results. Our framework is general to any pair-based semi-supervised clustering algorithm.

In the preprocessing stage, we adopt the farthest-first strategy to identify representative nodes in each cluster (similar to [2] and [17]). Subsequently, we repeat an online cluster-select-query loop to allow data and clustering-dependent constraints to be actively added to the constraint set. In each iteration, our algorithm seeks the most uncertain node in the dataset. The algorithm then queries constraints between this node and previously identified nodes representing each cluster. An online oracle then provides the must-link or cannot-link value for each constraint.

We define two notions of node uncertainty, from a local nonparametric and one from a global parametric view. The local measure is based on cluster contradiction between a point and its nearest neighbors, while the global measure is computed from cluster confusion in a global mixture model of the data.

Why uncertainty? If enough information has been extracted to unambiguously identify the correct relationship between each true cluster and each node, then the clustering algorithm should achieve perfect accuracy [2, 17]. Intuitively, then, uncertainty in the cluster identity of nodes leads to clustering errors. Furthermore, in this paper we show that, given some relaxation, the most uncertain node is the same as the node that will maximally reduce the uncertainty of the whole dataset if queried. Thus, by issuing queries that will unambiguously identify the cluster of the most uncertain node at each iteration we are able to make optimal progress toward a completely certain, and thus trivially solvable, clustering problem.

Why *node* uncertainty? We adopt an approach based on node rather than pair uncertainty for two reasons. First, an uncertain pair may be uncertain either because it contains one uncertain node or

[1] Department of Computer Science and Engineering, SUNY at Buffalo

because it contains *two* uncertain nodes. In the latter case, a constraint between these nodes yields limited information because the constraint will not extrapolate well beyond these two nodes. Second, pair selection has an inherently higher complexity due to the presence of n^2 constraints for every n nodes, limiting the scalability of a pair-based approach.

Main contributions. Our paper makes four contributions:

- We present a new active selection framework, **O**nline **C**onstraint **S**election via **N**ode **U**ncertainty (**OCSNU**), with general applicability to pair-based semi-supervised clustering methods.
- We propose and justify two novel definitions of node uncertainty.
- We show theoretically that selecting the most uncertain node at each iteration yields the maximum reduction in total uncertainty for the dataset.
- We test our framework and constraint selection methods on two different semi-supervised clustering algorithms and conclusively obtain strong results.

We compare OCSNU with baseline and state of the art active clustering and active learning techniques on UCI machine learning datasets, two digits datasets [1], one gene dataset [6] and part of the Caltech 101 image dataset [8]. The results show that given the same number of pairs queried, OCSNU using our node uncertainty measures can obtain better accuracy than existing methods.

Relation to active learning. Active query selection has previously seen extensive use in the field of active learning. [20] and [4], for example, both offer methods similar to ours in that they select and query uncertain nodes. However, in active learning algorithms, the oracle needs to know the class label of the queried data point. This approach is not applicable to many clustering problems, where the oracle can only give feedback about the relation between pairs of nodes. Though we implicitly label queried nodes by comparing them to a set of exemplar nodes representing each cluster, we do so strictly via pairwise queries.

Additionally, though for the sake of comparison we begin our experiments with the cluster structure fully explored (i.e. at least one example of each class identified), in real data this may not be the case—the total number of clusters may not be known during the initial exploration phase. Our method can dynamically add new clusters as needed to adapt to such situations, while these (and most other) active learning methods cannot.

2 Active constraint selection for semi-supervised clustering

In this section, we first present the framework for our active semi-supervised clustering, then describe details of each framework-component and two novel node uncertainty models/definitions for use in active constraint selection. Throughout the rest of the paper, let X be a data set with n nodes, $X = \{x_1, x_2, \cdots, x_n\}$ and $x_i \in R^d$ with k total clusters.

2.1 Active semi-supervised clustering framework

We now present our framework for active clustering—recall the basic flow of the algorithm depicted earlier in Figure 1. We divide the whole framework into two parts: the exploration preprocess and the online active selection/clustering process. We begin by running the exploration preprocess once, before computing any clustering results. The goal is to obtain a set of exemplar nodes Q^0, with (hopefully but not necessarily) at least one representing each true cluster.

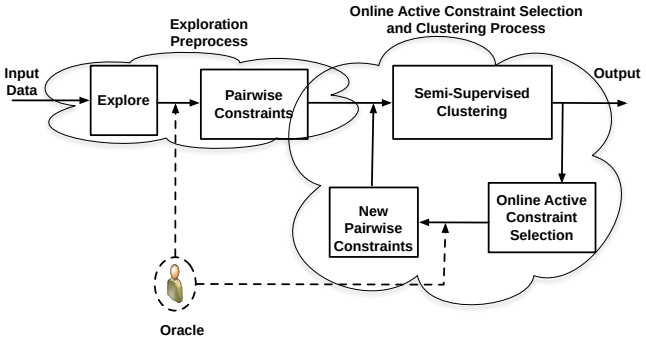


Figure 1. Flowchart of OCSNU for active clustering. First run the exploration preprocess to obtain a set of exemplar nodes and associated pair constraints, then input constraints to the online active constraint selection and clustering process. Second, enter into the online cluster-select-query loop, in which we repeatedly do semi-supervised clustering based on the current constraints, then use the results of the clustering to actively select new constraints and query the oracle. This process continues until either the oracle is satisfied or a fixed number of queries have been made.

After this, the data and the initial Q^0 serve as inputs to the online active clustering process. In each iteration of this process, our approach will automatically select and query new pair constraints based on the current clustering results and add the newly obtained constraints into the current constraint set, then recluster using the new constraint set. We iterate the online active clustering process until the oracle is satisfied or a fixed number of queries have been made.

2.2 Exploration preprocess

In the exploration preprocess phase, we adopt the exploration process from [2], which uses a farthest-first traversal strategy to query and identify at least one node from each of the k clusters in a reasonably small number of attempts. The identified nodes from the same cluster are collected into a single node set, so there are at most k non-empty node sets. [2] prove that this strategy yields a significant $\ln k$ factor improvement in the number of queries required to identify the needed exemplars, as compared to random selection.

We denote the preprocessing as the 0^{th} iteration in the online process, and when it is finished we will have obtained k_0 node sets for use, such that the set of node sets $NS^0 = \{X_1, X_2, \cdots, X_{k_0}\}$, where each $X_i \in NS^0$ is a node set , and $k_0 \leq k$ since preprocessing could be stopped before k non-empty node sets are found. The initial identified nodes set $Q^0 = \bigcup_{X_i \in NS^0} X_i$. [2]

Using these node sets, we then begin the active clustering process, using the set of constraints represented in NS^0 (i.e. must-link constraints among all nodes in the same set, and cannot-link constraints among nodes in different sets) as input to an initial semi-supervised clustering operation.

2.3 Online active constraint selection

After obtaining a set of output clusters from a semi-supervised clustering algorithm, our method will then perform an active online

[2] Note the set of node sets NS^0 will be updated after each iteration in the online active clustering process. After the t^{th} iteration, the set of node sets NS^{t-1} will be updated to NS^t and $Q^t = \bigcup_{X_i \in NS^t} X_i$.

search for new constraints to further improve the clustering. As we described earlier, at each iteration we identify the most uncertain node in the dataset and query the relationship between this node and some of the previously identified exemplars. We show here that the most uncertain node is (given some relaxation), also the node that will yield the maximum reduction in total node uncertainty for the dataset when queried.

In the t^{th} iteration, Q^{t-1} is the set of queried nodes, which we consider to be "certain" (i.e. if we were clustering only these nodes we could do so trivially and unambiguously) and $X = Q^{t-1} \bigcup \neg Q^{t-1}$, where $\neg Q^{t-1}$ is the set of unqueried nodes. We define the uncertainty $\mathbf{U}$ of the dataset in the t^{th} iteration to be conditioned on the original data distribution and previous semi-supervised clustering results, which are in turn based on the queried node set Q^{t-1}. Thus the uncertainty can be expressed as $\mathbf{U}(\cdot|M(X, C^{t-1}))$, where C^{t-1} is clustering result at iteration $t-1$ and $M(\cdot,\cdot)$ is a model for calculating uncertainty (we provide two such models here).

For a set of nodes X', denote $\mathbf{U}(\{X'\}; M(X, C^{t-1})) = \sum_{x_i \in X'} \mathbf{U}(\{x_i\}; M(X, C^{t-1}))$ if all nodes in X' are conditionally independent under $M(X, C^{t-1})$. Since queried nodes are considered to be "certain" , we assume $\mathbf{U}(X; M(X, C^{t-1})) = \mathbf{U}(\neg Q^{t-1}; M(X, C^{t-1}))$, therefore our objective function for node selection is as follows:

$$\mathbf{x}_j^* = \underset{\mathbf{x}_j \in \neg Q^{t-1}}{\operatorname{argmax}} \mathbf{U}(\neg Q^{t-1}; M(X, C^{t-1})) - \mathbf{U}(\neg (Q')^{t-1}; M(X, (C')^{t-1})) \ , \quad (1)$$

where $(Q')^{t-1} = Q^{t-1} \bigcup \{\mathbf{x}_j\}$ and $(C')^{t-1}$ is the clustering result based on $(Q')^{t-1}$.

In order to solve Equation 1 and obtain node x_j, we make two assumptions: first, $(C')^{t-1}$ and C^{t-1} are sufficiently similar that,

$$\mathbf{U}(\neg (Q')^{t-1}; M(X(C')^{t-1})) \approx \mathbf{U}(\neg (Q')^{t-1}; M(X, C^{t-1})) \quad (2)$$

This assumption is reasonable because each iteration yields only a small number of localized constraints, so we expect that generally only a small subset of the clustering results is altered. The objective function thus becomes:

$$\begin{aligned} & \mathbf{U}(\neg Q^{t-1}; M(X, C^{t-1})) - \mathbf{U}(\neg (Q')^{t-1}; M(X, (C')^{t-1})) \\ \approx \quad & \mathbf{U}(\neg Q^{t-1}; M(X, C^{t-1})) - \mathbf{U}(\neg (Q')^{t-1}; M(X, C^{t-1})) \quad (3) \end{aligned}$$

Second, similar to the strong naive Bayes assumption (which is oversimplified but works quite well in many complex real-world situations) we assume node x_i is conditionally independent of every other node x_k for $i \neq k$. Thus we can infer that:

$$\begin{aligned} & \mathbf{U}(\neg Q^{t-1}; M(X, C^{t-1})) - \mathbf{U}(\neg (Q')^{t-1}; M(X, C^{t-1})) \\ = & \sum_{x_i \in \neg Q^{t-1}} \mathbf{U}(\{x_i\}; M(X, C^{t-1})) - \sum_{x_k \in \neg Q'^{t-1}} \mathbf{U}(\{x_k\}; M(X, C^{t-1})) \\ = \ & \mathbf{U}(\{x_j\}; M(X, C^{t-1})) \quad (4) \end{aligned}$$

Therefore, the objective function can be approximated as:

$$\mathbf{x}_j^* = \underset{\mathbf{x}_j \in \neg Q^{t-1}}{\operatorname{argmax}} U(\{x_j\}; M(X, C^{t-1})) \quad (5)$$

So the problem is transferred into finding the most uncertain node in the unqueried node set based on the current uncertainty model $M(X, C^{t-1})$. Next, we propose two definitions/models for node uncertainty, which we can then use to actively select new nodes and associated pair constraints. These two models represent, respectively, a local nonparametric and a global parametric view of uncertainty.

Local nonparametric structure model for node uncertainty. Sparse graphs are a widely used and effective way to describe the structure of data in many machine learning algorithms. Here we adopt a $\mathcal{K}$NN graph to represent the entire dataset, with the local structure of each node described by $\mathcal{K}$ edges.

Now define a "good" edge in the $\mathcal{K}$NN graph as an edge between two points in the same cluster. If the $\mathcal{K}$NN graph consists of all "good" edges then the output of a clustering algorithm run on the graph should match the $\mathcal{K}$NN graph connectivity. Alternately, in a graph with many "bad" edges (such as most graphs generated from real data), the clustering result usually does *not* obey the $\mathcal{K}$NN graph connectivity, and the graph will thus contain nodes whose neighbors are assigned to different clusters. Thus, uncertain nodes should be identifiable by the presence of "bad" edges linked to them.

Based on this observation, we define a local uncertainty model based on the degree of cluster assignment disagreement between a node and its neighbors. Thus, $M_L(X, C^{t-1})$ is based on the local structure of the data in the feature space and the clustering result in $t-1^{th}$ iteration. We calculate the uncertainty of a node $\mathbf{x}_j$ using:

$$\mathbf{U}(\{\mathbf{x}_j\}; M_L(X, C^{t-1})) = 1 - \frac{\#\{c_z = c_j, z \in N_j\}}{\#\{N_j\}} \ , \quad (6)$$

where $\frac{\#\{c_z = c_j, z \in N_j\}}{\#\{N_j\}}$ is the ratio of neighbors of node j in the graph that are assigned to same cluster as node $\mathbf{x}_j$ during clustering (notation c_z denotes the cluster index of neighbor z). If $\mathbf{U}(\{\mathbf{x}_j\}; M_L(X, C^{t-1}))$ is high, the cluster relationship between this node and its neighbors disagrees strongly with the local structure, so this node is highly uncertain.

We note that the distance measure and $\mathcal{K}$ value used to compute the $\mathcal{K}$NN graph do effect this uncertainty formulation, and serve as inputs to the algorithm. In our experiments, we select the commonly used Mahalanobis distance whose metric matrix is equal to the inverse covariance matrix of the data points, and $\mathcal{K} = 10$.

Global parametric model for node uncertainty. Most semi-supervised clustering methods learn a modified similarity kernel matrix as part of the clustering algorithm. Here, we define a new representation for the data by computing the k largest eigenvectors of the modified kernel matrix and using them as the features in a new data space. This functions similarly to the spectral eigenmap from input dimensionality d to dimensionality $k \leq d$:

$$\phi : \mathbf{x}_i \in \mathbb{R}^d \rightarrow \mathbb{R}^\infty \rightarrow \mathbb{R}^k \ . \quad (7)$$

$\mathbf{x}_i \in \mathbb{R}^d \rightarrow \mathbb{R}^\infty$ is accomplished using the kernel trick (with the kernel learned via semi-supervised clustering) to project the data from the original feature space to an infinitely high dimensional space. In the high dimensional space, we assume data nodes lie on a low dimensional manifold and each cluster is represented by a Gaussian.

Eigendecomposition then gives us $\mathbb{R}^\infty \rightarrow \mathbb{R}^k$, projecting the data from the high dimensional feature space back to a low dimensional representation. We can consider this projection process a marginalization over most of the infinite number of dimensions. In the ideal case, this projection should retain the distribution of data nodes, each cluster can thus be accurately represented in the low dimensional space by a Gaussian, and the whole dataset can be represented by a global Gaussian mixture model (GMM):

$$p(\mathbf{x}_i | \{\alpha_z\}, \{\mu_z\}, \{\Sigma_z\}) = \sum_{z=1}^{k} \alpha_z \mathcal{N}(\mathbf{x}_i; \mu_z, \Sigma_z) \ , \quad (8)$$

where $\{\alpha_z\}$ are the mixing weights and $(\{\mu_z\}, \{\Sigma_z\})$ are the component Gaussian parameters. Using EM, it is easy to estimate the parameters for the GMM and obtain the probability of each data point given each cluster z: $p(z|\mathbf{x}_i) = \frac{\alpha_z \mathcal{N}(\mathbf{x}_i;\mu_z,\Sigma_z)}{\sum_j^k \alpha_j \mathcal{N}(\mathbf{x}_i;\mu_j,\Sigma_j)}$. We can then define an uncertainty model $M_G(X, C^{t-1})$ based on the entropy of the $p(z|\mathbf{x}_i)$ distribution for a point, yielding an uncertainty measure:

$$\mathbf{U}(\{\mathbf{x}_j\}; M_G(X, C^{t-1})) = \sum_{z=1}^{k} p(z|\mathbf{x}_j) \log p(z|\mathbf{x}_j) \ , \quad (9)$$

where C^{t-1} implicitly determines this node uncertainty function via the learned kernel matrix.

Querying pairs based on the selected node. After finding the most uncertain node, we obtain pairs to query using the node sets NS^{t-1} as follows.

First, for each node set X_i, choose the single node within the set which is closest to the selected node $\mathbf{x}_j$: $\mathbf{x}_l = \text{argmin}_{\mathbf{x}_l \in X_i} Dist(\mathbf{x}_j, \mathbf{x}_l)$ and record this node and distance value.

Second, since there are k_{t-1} node sets, we will have recorded k_{t-1} nodes and distance values, so sort the nodes based on their corresponding distances. Now, in order of ascending distance, query the oracle on the relation between the selected node $\mathbf{x}_j$ and $\mathbf{x}_l$ until we find a must-link connection, then add $\mathbf{x}_j$ into the node set. If all of the relations are cannot-link, we create a new node set $X_{k_{t-1}+1}$ and add $\mathbf{x}_j$ to it. This new node set $X_{k_{t-1}+1}$ is then added to NS^{t-1} to obtain NS^t, and Q^{t-1} is correspondingly updated to Q^t. Since the relation between the new node and all node sets in NS^t is known, we can generate new pairwise constraints between the selected node $\mathbf{x}_j$ and all nodes in Q^t.

2.4 Semi-supervised clustering

When new pairwise constraints are obtained, we add them to the original constraint set. We then run the semi-supervised spectral clustering method on the new constraint set. There are a number of possible candidate methods for semi-supervised clustering, here we choose the two spectral clustering-based algorithms described below, due to the power and generality of spectral methods.

Spectral Learning This method [14] is a simple, easily implemented spectral learning algorithm, which applies the constraints directly. Given the set of pairwise constraints, the algorithm directly modifies the affinity(similarity) matrix W, then performs spectral clustering on the modified affinity matrix. Specifically, the new affinity matrix is now defined as:

- For each pair of must-linked points (i, j) assign the values $W_{ij} = W_{ji} = 1$.
- For each pair of cannot-linked points (i, j) assign the value $W_{ij} = W_{ji} = 0$.
- Normalize $N = \frac{1}{d_{max}}(W + d_{max}I - D)$, where d_{max} is the maximum of summed rows of W, and I is the identity matrix.

The spectral clustering algorithm then proceeds normally.

Flexible Semi-supervised spectral clustering This method was proposed by [22]. Rather than applying the constraints directly, this method optimizes the following objective function:

$$\underset{u \in R^N}{\text{argmin}}\, u^T L u \text{ s.t. } u^T D u = \sum_i D_{ii}, u^T Q u \geq \alpha \ , \quad (10)$$

where L is the normalized graph Laplacian of the affinity matrix and D is the diagonal degree matrix. It is easy to see that the difference from spectral clustering is the new term $u^T Q u \geq \alpha$. Q is the constraint matrix such that $Q_{ij} = 1$ for must-link, $Q_{ij} = -1$ for cannot-link and $Q_{ij} = 0$ for unknown pairs. u is the cluster assignment vector. $u^T Q u$ can thus be considered as a measure of how well the pairwise constraints conform to the cluster assignment u. Since $u \in R^N$ is a vector, the method is limited into two-cluster cases.

In our experiments, we apply OCSNU in conjunction with our local nonparametric and global parametric uncertainty measures to the above two semi-supervised clustering algorithms, and the results show the superiority and generality of our methods.

3 Experiments

3.1 Dataset

We evaluate our proposed framework and uncertainty measures on UCI machine learning, digits [1], gene (Cho's [6] and image datasets. The image data is a subset of the Caltech-101 [8], with images represented by image gist features (reduced to 100 dimensions via PCA). More details in Table 1. All results are averaged over 30 runs.

Table 1. UCI Datasets, DIGITS, GENE and IMAGE Datasets

Name	$\#Classes$	$\#Instances$	$\#Features$
Balance	3	625	4
Bupa	2	345	6
Diab	2	768	8
Sonar	2	208	60
Wine	3	178	13
Semeion digits	10	1593	256
Multiple features digits	10	2000	649
Cho	5	386	16
Caltech 5-Class	5	307	100

3.2 Evaluation protocols

We evaluate all cluster solutions via two commonly used cluster evaluation metrics: Rand Index [18] and V-Measure [19].

Rand Index. To measure the performance, we first adopt the well-known Rand Index, defined by:

$$\text{Accuracy} = \sum_{i>j} \frac{1\{1\{c_i = c_j\} = 1\{\hat{c}_i = \hat{c}_j\}\}}{n(n-1)/2}$$

where $1\{\cdot\}$ is the indicator function that outputs 1 when the input is true and 0 otherwise. c_i and $\hat{c}_i$ are the true cluster membership and the predicted cluster membership of the ith data point, respectively. n is the number of data points in the dataset.

V-Measure. We also employ the well-known V-Measure metric, which defines entropy-based measures for the completeness and homogeneity of the clustering results, and computes the harmonic mean of the two (in our case, we weight both measures equally). It is defined as follow:

$$V_\beta = \frac{(1+\beta) * h * c}{(\beta * h) + c} \quad (11)$$

where h is homogeneity and c is completeness.

3.3 Baseline and state of the art methods

To evaluate our active clustering framework and the two different selection strategies, we compare our method with the following set of methods, including a baseline and multiple the state of the art[3]:

[3] All selection algorithm code was downloaded from authors' websites except FFQS, which we reimplemented.

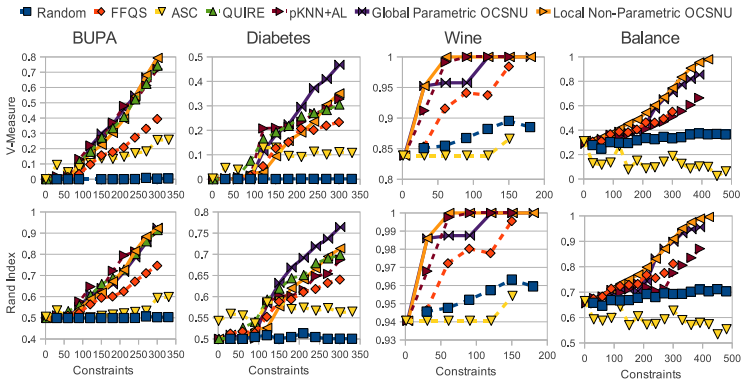


Figure 2. Rand Index and V-measure accuracy (vertical axes) with increasing pair relation queries on UCI datasets using spectral learning. *Best viewed in color.*

- **Random**: a baseline in which constraints are randomly sampled from the available pool.
- **FFQS** [2]: this method uses the farthest-first strategy to explore the data, then queries points randomly against the discovered node sets.
- **ASC** [21]: a pair-based method that queries pairs that will yield the maximum reduction in expected pair value error. In the original paper it is used in conjunction with flexible spectral clustering, and thus only applicable to two-class problems, but the active selection method itself can be applied to multiclass cases as well.
- **QUIRE** [12]: this is binary-only active learning method that computes node uncertainty based on the informativeness and representativeness of each node. We use our OCSNU framework to generate the requested node labels from pairwise queries.
- **pKNN+AL** [13]: this is is a minmax-based multi-class active learning method. Again, we use our OCSNU framework to translate node label requests into pairwise constraint queries.
- **Global Parametric OCSNU**: this is our proposed framework, using our global parametric node uncertainty measure to select nodes.
- **Local Nonparametric OCSNU**: this is our proposed framework, using our local nonparametric node uncertainty measure to select nodes.

All the above methods select constraints and feed the queried pair constraints to the two semi-supervised clustering methods we introduced in Section 2.4.

3.4 Results

Results on UCI dataset. We compared on 5 different UCI datasets, 3 binary and 2 multiclass.

Figure 2 shows the accuracy of different active selection methods with varying numbers of constraints when using spectral learning as the semi-supervised clustering algorithm. The first row of the figure shows V-measure values and the second Rand Index. Both metrics yield the same conclusion: our two methods are competitive in the two-cluster case and clearly superior for multicluster (wine and balance) problems. In particular, our local nonparametric uncertainty method is the best or tied for best on all but the Diabetes dataset, where it is beaten out by our global parametric method.

Figure 3 presents the results for various active selection methods in conjunction with the flexible semi-supervised spectral clustering method. As this is a binary clustering method, only 2-class datasets are used. Local nonparametric OCSNU still achieves competitive results on the BUPA and diabetes sets, and is clearly the best on sonar, though the global parametric uncertainty measure performs noticeably worse here. Interestingly, the different active selection methods are all much closer to each other in performance with this clustering method. For BUPA in particular, the choice of active method seems to have little effect on the results.

Results on Gene, Digits, images datasets. Figure 4 shows Rand Index and V-measure results for active clustering on the gene, image, digits datasets. Local nonparametric OCSNU performs particularly well here, most notably on the Caltech and Semeion datasets. We note that the robustness of the local method among all data tested may be partially attributable to the natural synergy between it and $\mathcal{K}$NN graph-based clustering methods, of which spectral clustering is a prime example. By comparison, the global parametric uncertainty measurement appears to be less robust, performing well on Caltech and Cho, but very poorly on the digits data. This is likely due to that fact that the parametric model underlying our global method may itself be a poor fit for some datasets.

Also notable is that the multiple feature digits dataset is the only case where the V-measure and Rand index metrics differ significantly in their relative assessment of the different selection algorithms, with V-measure scoring local OCSNU highest, and Rand index granting the best result to FFQS.

4 Conclusion

In this paper, we have considered the problem of active constraint selection for semi-supervised spectral clustering. Our paper makes two primary contributions: first, we describe a powerful general framework for online active semi-supervised clustering based on node uncertainty; second, we propose two methods for actively sampling constraints by transforming the pair-uncertainty problem into a node-uncertainty problem. We test our online active framework and selection criteria with two different semi-supervised clustering algorithms, against a number of existing active selection methods (including active clustering and active learning techniques), and find our method to be the most effective and robust of those surveyed.

In the future we hope to explore new node selection criteria. In particular, we wish to examine the possibility of a nonparametric global uncertainty measure, and of a compound uncertainty measure that considers both local and global structure.

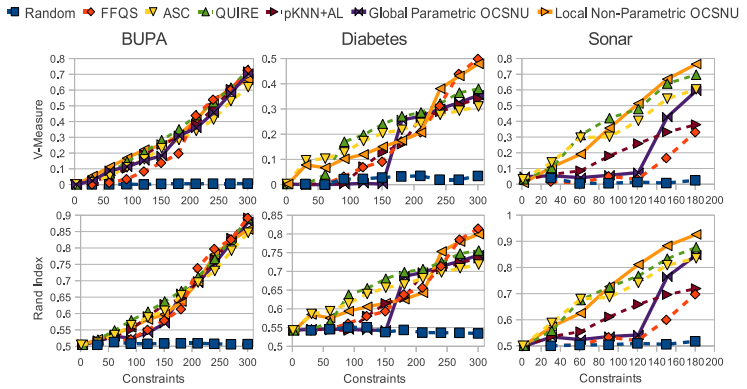


Figure 3. Rand Index and V-measure accuracy (vertical axes) with increasing pair relation queries on UCI datasets using flexible semi-supervised clustering. Since the method is limited to the two-class case, only two-class datsets were tested. *Best viewed in color.*

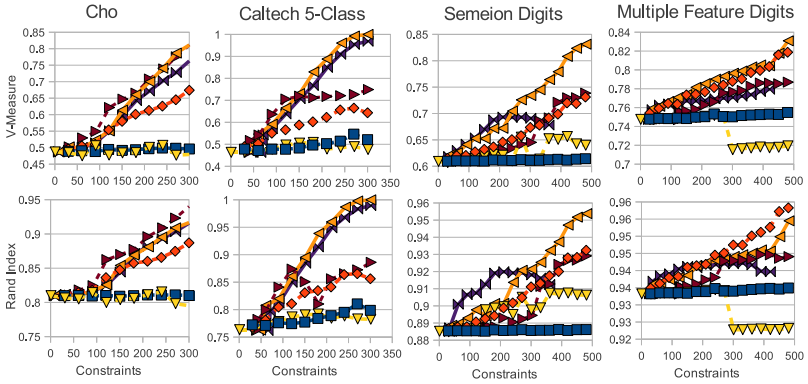


Figure 4. Rand Index and V-measure accuracy (vertical axes) with increasing pair relation queries on the gene, image and digits datasets using spectral learning. *Best viewed in color.*

REFERENCES

[1] A. Asuncion and D.J. Newman, 'UCI machine learning repository', (2007).

[2] S. Basu, A. Banerjee, and R.J. Mooney, 'Active semi-supervision for pairwise constrained clustering', in *ICDM*, pp. 333–344, (2004).

[3] S. Basu, M. Bilenko, and R.J. Mooney, 'A probabilistic framework for semi-supervised clustering', in *SIGKDD*, pp. 59–68. ACM, (2004).

[4] A. Beygelzimer, D. Hsu, J. Langford, and T. Zhang, 'Agnostic active learning without constraints', *Arxiv preprint arXiv:1006.2588*, (2010).

[5] Ling Chen and Chengqi Zhang, 'Semi-supervised variable weighting for clustering', in *SDM*, pp. 862–871, (2011).

[6] R.J. Cho, M.J. Campbell, E.A. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T.G. Wolfsberg, A.E. Gabrielian, D. Landsman, D.J. Lockhart, et al., 'A genome-wide transcriptional analysis of the mitotic cell cycle', *Molecular cell*, **2**(1), 65–73, (1998).

[7] I. Davidson, K. Wagstaff, and S. Basu, 'Measuring constraint-set utility for partitional clustering algorithms', *PKDD*, 115–126, (2006).

[8] L. Fei-Fei, R. Fergus, and P. Perona, 'One-shot learning of object categories', *PAMI*, 594–611, (2006).

[9] Y Fu, B LI, X Zhu, and C Zhang, 'Do they belong to the same class: active learning by querying pairwise label homogeneity', in *CIKM*, pp. 2161–2164. ACM, (2011).

[10] S.C.H. Hoi and R. Jin, 'Active kernel learning', in *ICML*, pp. 400–407. ACM, (2008).

[11] S.C.H. Hoi, R. Jin, and M.R. Lyu, 'Learning nonparametric kernel matrices from pairwise constraints', in *ICML*, pp. 361–368. ACM, (2007).

[12] S.J. Huang, R. Jin, and Z.H. Zhou, 'Active learning by querying informative and representative examples'. NIPS, (2010).

[13] P. Jain and A. Kapoor, 'Active learning for large multi-class problems', in *CVPR*, pp. 762–769. IEEE, (2009).

[14] K. Kamvar, S. Sepandar, K. Klein, D. Dan, M. Manning, and C. Christopher, 'Spectral learning', in *IJCAI*. Stanford InfoLab, (2003).

[15] Z. Li and J. Liu, 'Constrained clustering by spectral kernel learning', in *ICCV*, pp. 421–427. IEEE, (2009).

[16] Z. Lu and M.A. Carreira-Perpinán, 'Constrained spectral clustering through affinity propagation', in *CVPR*, pp. 1–8. IEEE, (2008).

[17] P.K. Mallapragada, R. Jin, and A.K. Jain, 'Active query selection for semi-supervised clustering', in *ICPR*, pp. 1–4. IEEE, (2008).

[18] W.M. Rand, 'Objective criteria for the evaluation of clustering methods', *JASA*, 846–850, (1971).

[19] A. Rosenberg and J. Hirschberg, 'V-measure: A conditional entropy-based external cluster evaluation measure.', in *EMNLP-CoNLL'07*, pp. 410–420, (2007).

[20] B. Settles, 'Active learning literature survey', Technical report, (2010). (unpublished report).

[21] X. Wang and I. Davidson, 'Active Spectral Clustering', in *ICDM*, (2010).

[22] X. Wang and I. Davidson, 'Flexible constrained spectral clustering', in *SIGKDD*, pp. 563–572. ACM, (2010).

[23] E.P. Xing, A.Y. Ng, M.I. Jordan, and S. Russell, 'Distance metric learning with application to clustering with side-information', *NIPS*, 521–528, (2003).

[24] Q. Xu, M. Desjardins, and K. Wagstaff, 'Active constrained clustering by examining spectral eigenvectors', in *Discovery Science*, pp. 294–307. Springer, (2005).

An Incremental On-line Classifier for Imbalanced, Incomplete, and Noisy Data

Marko Tscherepanow and **Sören Riechers**[1]

Abstract. Incremental on-line learning is a research topic gaining increasing interest in the machine learning community. Such learning methods are highly adaptive, not restricted to distinct training and application phases, and applicable to large volumes of data. In this paper, we present a novel classifier based on the unsupervised topology-learning TopoART neural network. We demonstrate that this classifier is capable of fast incremental on-line learning and achieves excellent results on standard datasets. We further show that it can successfully process imbalanced, incomplete, and noisy data. Due to these properties, we consider it a promising component for constructing artificial agents operating in real-world environments.

1 Introduction

The development of artificial agents with cognitive capabilities as they are found in humans and animals is still an unsolved problem. While biological agents can manage uncertain ever-changing environments with complex interdependencies, artificial agents are often limited to very specific, extremely simplified, and unchanging problems.

One possibility to improve the performance of current artificial systems is the usage of incremental learning mechanisms (e.g, [1], [18], and [19]). In contrast to traditional machine learning approaches based on distinct training and application phases, incremental approaches have to cope with additional difficulties – the most important being the *stability-plasticity dilemma* [15]: while plasticity is required in order to learn anything new, stability ensures that already acquired knowledge does not get lost in an uncontrolled way.

Sensor data obtained in natural environments often exhibit characteristics which further impede learning. In particular, their distributions can be non-stationary, noisy, and imbalanced. In addition, individual input vectors may be incomplete, for instance, due to different sensor latencies.

In this paper, we present an incremental classifier (see Section 4) based on the unsupervised TopoART neural network [26] (see Section 3). It is capable of stable and plastic incremental on-line learning and can cope with noisy, imbalanced, and incomplete data. These properties are shown using synthetic datasets (see Section 3) and real-world datasets from the UCI machine learning repository [12] (see Section 5).

2 Related Work

Adaptive Resonance Theory (ART) neural networks constitute an early approach to unsupervised incremental on-line learning. They incrementally learn a set of templates called categories. Some well-known ART variants are Fuzzy ART [5] and Gaussian ART [29]. While Fuzzy ART is capable of stable incremental learning using hyperrectangular categories but is prone to noise, the categories of Gaussian ART are Gaussians, which diminishes its sensitivity to noise but impairs the stability of learnt representations.

Regarding the formed representations, Gaussian ART is strongly related to on-line kernel density estimation (oKDE) [20]: oKDE incrementally estimates a Gaussian mixture model representing a given data distribution. Depending on an adjustable parameter, the estimated distribution is stable to a certain degree.

Incremental topology-learning neural networks, such as Growing Neural Gas [13], constitute an alternative approach to unsupervised on-line learning. Some of these networks, e.g., the Self-Organising Incremental Neural Network (SOINN)[14] and Incremental Growing Neural Gas (IGNG) [21], alleviate the problems resulting from the stability-plasticity dilemma. However, they rely on neurons representing prototype vectors. During learning, any shift of these prototype vectors in the input space inevitably causes some loss of information.

TopoART [26] has been proposed as a neural network combining properties from ART and topology-learning neural networks. As its architecture and representations are based on Fuzzy ART [5], each neuron (also called node) represents a hyperrectangular region of the input space, which can only grow during learning. As a result, once an input vector has been enclosed by a category, it will stay inside. In addition, TopoART inherited the insensitivity to noise from SOINN [14], which is a major improvement in comparison to Fuzzy ART.

The approaches mentioned above are not applicable to supervised learning tasks such as classification. However several extensions exist that enable their application to such problems, e.g., ARTMAP [4] for ART networks, Bayes' decision rule [28] for mixture models, and Life-long Learning Cell Structures [16] for prototype-based incremental topology-learning neural networks. The resulting supervised learning methods usually inherit the characteristics of their unsupervised components and ancestors, respectively. In particular, they learn locally; i.e., adaptations are restricted to a limited set of parameters.

The Perceptron [22], a very early approach to supervised on-line learning, possesses a distributed memory. As a consequence, all trainable parameters are altered during learning rendering Perceptrons prone to catastrophic forgetting. Furthermore, they have a fixed structure limiting the complexity of the knowledge that can be stored. These problems were inherited by multi-layer Perceptrons (MLPs) [23]. Cascade-Correlation neural networks [11] partially solve them by means of an incremental structure. But they are restricted to off-line (batch) learning.

[1] Applied Informatics, Bielefeld University, Germany, email: marko@techfak.uni-bielefeld.de, marko@tscherepanow.de

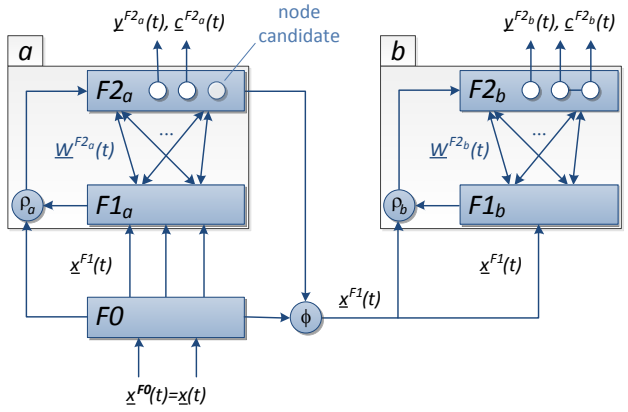


Figure 1. TopoART neural networks consist of two modules sharing the input layer $F0$. Both modules function in an identical way. However, input to module b is controlled by module a.

Support vector machines (SVMs) [8], an alternative extension of Perceptrons, learn by solving a quadratic programming problem based on a fixed training set; i.e., off-line. A subset of the training samples, called support vectors, is chosen to construct separating hyperplanes. Although there are approaches to on-line SVMs (e.g., [2] and [6]), the underlying model imposes several problems: an adequate kernel has to be selected in advance, the number of occurring classes needs to be known or is limited to two, and a possibly large set of input samples has to be collected in addition to the support vectors to stabilise the learning process.

Recently, several incremental classification frameworks based on ensemble learning, e.g, ADAIN [17] and Learn++.NSE [10], have been proposed. These approaches assume that data be provided in data chunks containing multiple samples. As for each of these chunks an individual base classifier is trained, a voting mechanism is required so as to obtain a common prediction. Furthermore, additional learning methods may be necessary; ADAIN, for instance, uses an MLP to construct a mapping function connecting past experience with present data.

TopoART-C, the classifier presented in this paper, is based on the unsupervised TopoART network. TopoART was chosen as a basis in order to take advantage of its beneficial properties, namely its capability of stable incremental on-line learning of noisy data. TopoART-C constitutes an extension of TopoART for classification tasks like the Simplified Fuzzy ARTMAP approach [27] used for Fuzzy ART. The usage of an additional mask layer further allows predictions to be made based on incomplete data.

3 TopoART

TopoART is strongly related to Fuzzy ART [5]: it shares its basic representations, its choice and match functions, and its principal search and learning mechanisms. However, TopoART extends Fuzzy ART in such a way that it becomes insensitive to noise and capable of topology learning. One important part of the noise filtering mechanism is the combination of multiple Fuzzy ART-like modules, where preceding modules filter the input for successive ones. Therefore, the standard TopoART architecture as proposed in [26] (see Fig. 1) consists of two modules (a & b). Besides noise filtering, these modules cluster input data at two different levels of detail.

The clusters are composed of hyperrectangular categories, which are encoded in the weights of neurons in the respective $F2$ layer. By learning edges between different categories, clusters of arbitrary shapes are formed.

If an input vector

$$\underline{x}(t) = \big[x_1(t), \ldots, x_d(t)\big]^T \tag{1}$$

is fed into such a network, complement coding is performed resulting in the vector

$$\underline{x}^{F1}(t) = \big[x_1(t), \ldots, x_d(t), 1 - x_1(t), \ldots, 1 - x_d(t)\big]^T. \tag{2}$$

As a consequence of this encoding that was inherited from Fuzzy ART, all elements $x_i(t)$ of the input vector $\underline{x}(t)$ must be normalised to the interval $[0, 1]$.[2]

$\underline{x}^{F1}(t)$ is first propagated to the $F1$ layer of module a. From here, it is used to activate the $F2$ nodes of module a based on their weights given by the matrix $\underline{W}^{F2a}(t)$.

As TopoART networks learn incrementally and on-line, training and prediction steps can be mixed arbitrarily. During training, the activation (choice function)

$$z_j^{F2}(t) = \frac{\big\|\underline{x}^{F1}(t) \wedge \underline{w}_j^{F2}(t)\big\|_1}{\alpha + \big\|\underline{w}_j^{F2}(t)\big\|_1} \tag{3}$$

of each $F2$ node j is computed first. $\|\cdot\|_1$ and $\wedge$ denote the city block norm and a component-wise minimum operation, respectively (cf. [5]). The node with the highest activation becomes the best-matching node bm. Its weights are adapted if the match function

$$\frac{\big\|\underline{x}^{F1}(t) \wedge \underline{w}_j^{F2}(t)\big\|_1}{\big\|\underline{x}^{F1}(t)\big\|_1} \geq \rho \tag{4}$$

is fulfilled for $j{=}bm$. Otherwise, the current node bm is reset and a new best-matching node is determined. If a suitable best-matching node bm has been found, a second-best-matching node sbm fulfilling Eq. 4 is sought.

The categories of the best-matching node and the second-best-matching node are allowed to grow in order to enclose $\underline{x}^{F1}(t)$ or partially learn $\underline{x}^{F1}(t)$, respectively:

$$\underline{w}_{bm}^{F2}(t+1) = \underline{x}^{F1}(t) \wedge \underline{w}_{bm}^{F2}(t) \tag{5}$$

$$\underline{w}_{sbm}^{F2}(t+1) = \beta_{sbm}\big(\underline{x}^{F1}(t) \wedge \underline{w}_{sbm}^{F2}(t)\big) + (1 - \beta_{sbm})\underline{w}_{sbm}^{F2}(t) \tag{6}$$

In order to learn the topology of the data, bm and sbm are connected by an edge. Already existing edges are not modified. If the $F2$ layer is empty or no node is allowed to learn, a new node with $\underline{w}_{new}^{F2a}(t+1){=}\underline{x}^{F1}(t)$ is incorporated.

According to Eq. 4, the maximum size of the categories is limited by the vigilance parameter ρ. Here, the vigilance parameter ρ_b of module b is determined depending on the vigilance parameter ρ_a of module a:

$$\rho_b = \frac{1}{2}(\rho_a + 1) \tag{7}$$

A value of $\rho_a{=}0$ means that a single category of module a can cover the entire input space, while a value of $\rho_a{=}1$ results in categories containing single samples.

[2] This normalisation usually requires an estimation of the minimum and maximum values for each x_i.

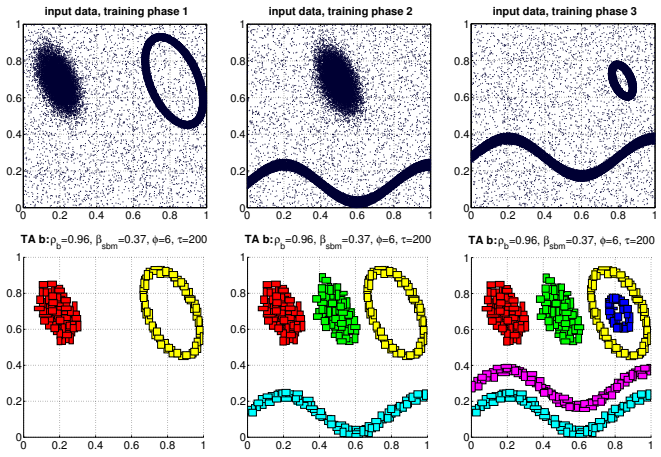


Figure 3. Results for non-stationary data. The training was performed in three successive phases (top row). In the bottom row, the corresponding clusters formed by a TopoART network after finishing the respective phase are shown. The network parameters were adopted from Fig. 2.

The $F2$ neurons of both modules possess a counter denoted by n_j. Each time a node is adapted, the corresponding counter is incremented. Furthermore, all $F2$ nodes j with $n_j < \phi$ are removed every τ learning cycles of the respective module. Therefore, such neurons are called node candidates. In contrast, nodes with $n_j \geq \phi$ are permanent; i.e., their categories are completely stable. The node candidates and the permanent nodes can be considered as the short-term memory and the long-term memory of the network, respectively.

$\underline{x}^{F1}(t)$ is only propagated to module b for training if the best-matching node of module a is permanent. In module b, the processes of category search and weight adaptation are repeated for the corresponding $F2$ nodes using ρ_b instead of ρ_a. In conjunction with the input filtering and the node-removal process, this constitutes a powerful noise reduction mechanism. Figure 2 illustrates this mechanism in comparison to two other popular unsupervised learning methods. The applied dataset consists of six clusters with 15,000 samples each as well as ten percent of uniformly distributed random noise (100,000 samples in total). In order to create a stationary data distribution, the samples were presented once in random order.

TopoART and SOINN were able to determine a detailed clustering reflecting the six clusters of the input distribution in a high level of detail. Here, the representation was refined from module a to module b and from SOINN layer 1 (SOINN 1) to SOINN layer 2 (SOINN 2). The representation of oKDE also reflects the underlying components, but does not include the topological structures. Furthermore, several Gaussians exclusively represent noise regions.

The capability of TopoART to incrementally learn stable representations from noisy non-stationary data is demonstrated in Fig. 3. Here, the input data used before were reordered and presented in three consecutive phases. After each training phase, TopoART has learnt the respective new clusters and the clusters formed during earlier training phases remained stable.

For prediction, $\underline{x}^{F1}(t)$ is directly propagated to both modules where the respective best-matching nodes are determined. Here, usually the alternative activation function

$$z_j^{F2}(t) = 1 - \frac{\left\| \left(\underline{x}^{F1}(t) \wedge \underline{w}_j^{F2}(t)\right) - \underline{w}_j^{F2}(t) \right\|_1}{d} \qquad (8)$$

that is independent from the category size is applied and the match function is not computed. The output of a module consists of a vector $\underline{y}^{F2}(t)$ with

$$y_j^{F2}(t) = \begin{cases} 0 & \text{if } j \neq bm \\ 1 & \text{if } j = bm \end{cases} \qquad (9)$$

and a vector $\underline{c}^{F2}(t)$ reflecting the clustering structure. For reasons of stability, node candidates are ignored during prediction.

Details on the adjustment and the effects of the parameters ρ_a, β_{sbm}, ϕ, and τ can be found in [26].

4 TopoART-C

In contrast to TopoART, which clusters presented data, a classifier requires additional information. In particular, a class label λ_i is associated with each input vector $\underline{x}(t)$ comprising one or more features $x_i(t)$. This label is either presented for training or predicted based on $\underline{x}(t)$. $\Lambda(t)$ denotes the set of all known class labels at time step t. In incremental learning scenarios $\Lambda(t)$ may grow if new data become available. The current number of known classes is given by $|\Lambda(t)|$. In order to construct a classifier inheriting the advantageous properties of TopoART, the principal structure of TopoART was preserved and extended by three additional layers (see Fig. 4).

Both modules obtained a classification layer $F3$, the nodes of which represent possible classes λ_i. These layers receive the output $\underline{y}^{F2}(t)$ of the respective F2 layer as input. Furthermore, a mask layer $F0_m$ was incorporated so as to enable predictions based on incomplete input data.

4.1 Training

TopoART-C is trained in a similar way to TopoART. But in order to account for the class labels, the match function (cf. Eq. 4) was modified:

$$\frac{\left\| \underline{x}^{F1}(t) \wedge \underline{w}_j^{F2}(t) \right\|_1}{\left\| \underline{x}^{F1}(t) \right\|_1} \geq \rho \quad \text{and} \quad \text{class}(j) = \lambda(t) \qquad (10)$$

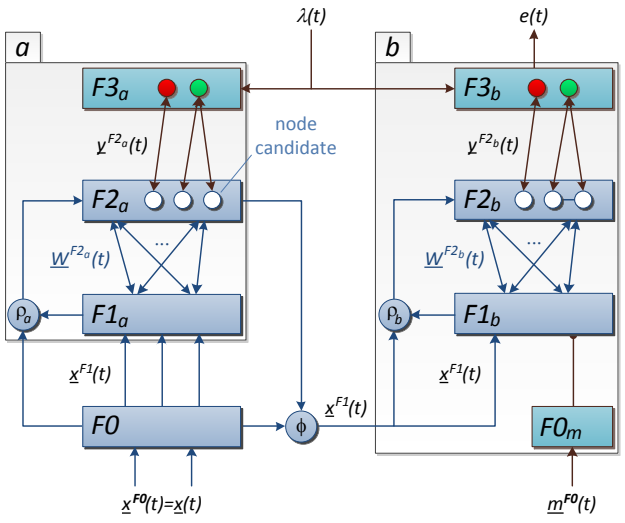


Figure 4. Structure of TopoART-C. TopoART-C extends the structure of TopoART by adding a classification layer $F3$ to each module and a mask layer $F0_m$ to module b.

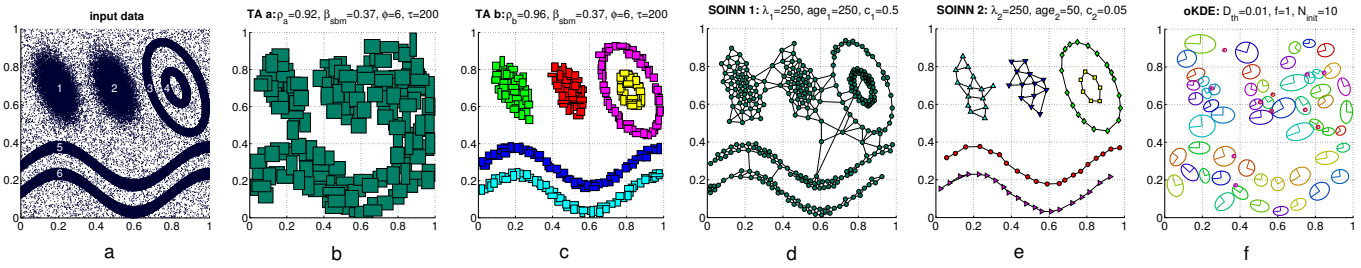


Figure 2. Clustering results for stationary data. A two-dimensional synthetic data distribution was learnt by TopoART (TA), SOINN, and oKDE. The relevant parameters were manually chosen in such a way as to fit the data. Different clusters of TopoART (b & c) and SOINN (d & e) are coloured differently. For SOINN, the edges connecting individual prototype vectors are shown, as well. In contrast to TopoART and SOINN, the distribution estimated by oKDE (f) consists of a mixture of Gaussians drawn as ellipses marking the standard deviations. It does not reflect the topological structure of the data.

Here, $\lambda(t)$ denotes the class label of $\underline{x}(t)$ and $\mathrm{class}(j)$ the class encoded by the $F3$ node that is connected with node j.[3]

Assuming the match function cannot be fulfilled for any existing $F2$ node, a new one with $\underline{w}^{F2}_{new}(t+1)=\underline{x}^{F1}(t)$ is incorporated like in the original TopoART network. Additionally, it is linked to the $F3$ node representing $\lambda(t)$. If the network does not know $\lambda(t)$, a new $F3$ node representing this label is inserted.

4.2 Prediction

During prediction, the class label λ_i which best fits the input vector $\underline{x}(t)$ is computed by module b using the decision rule:

$$e(t) = \arg \max_{\lambda_i \in \Lambda(t)} d_{\lambda_i}(t) \tag{11}$$

The discrimination function $d_{\lambda_i}(t)$ measures the similarity of $\underline{x}(t)$ with the internal representation of class λ_i:

$$d_{\lambda_i}(t) = \sum_{\substack{j \in \Upsilon_b(t) \\ \mathrm{class}(j)=\lambda_i}} y_j^{F2_b}(t) \tag{12}$$

$d_{\lambda_i}(t)$ depends on the output $\underline{y}^{F2_b}$ of the $F2$ layer of module b. The set of all nodes of this layer is denoted by $\Upsilon_b(t)$. Module a is completely neglected, as it is only required for training in order to filter irrelevant data.

If the original output function (Eq. 9) is applied, $e(t)$ yields the class label of the permanent node whose category has the closest distance to $\underline{x}(t)$. But depending on the class boundaries, the resulting prediction may be suboptimal. Therefore, we propose a more general output function, which can consider more than one node and allows for problem specific adaptations. Here, two principal cases have to be distinguished: either $\underline{x}(t)$ lies inside one or more categories or it is enclosed by no category. In the first case, the class label can be derived from the enclosing categories summarised in the set

$$\mathcal{E}(t) = \left\{ j \in \Upsilon_b(t) : z_j^{F2_b}(t) = 1 \right\}. \tag{13}$$

These nodes are characterised by a maximum activation $z_j^{F2_b}(t)$ according to Eq. 8. Using the modified output function

$$y_j^{F2_b}(t) = \begin{cases} 1 & \text{, if } j = \arg \min_{k \in \mathcal{E}(t)} S_k(t) \\ 0 & \text{, otherwise,} \end{cases} \tag{14}$$

[3] Each $F2$ node can only be connected with a single $F3$ node.

the prediction corresponds to the class label associated with the smallest category containing $\underline{x}(t)$. The category size $S_k(t)$ is defined as

$$S_k(t) = \sum_{i=1}^{d} \left| \left(1 - w^{F2}_{k,d+i}(t)\right) - w^{F2}_{k,i}(t) \right|. \tag{15}$$

In the second case, i.e., if no category encloses $\underline{x}(t)$, predictions are computed based on the set $\mathcal{N}$ of closest neighbours:

$$\mathcal{N}(t) = \left\{ j \in \Upsilon_b(t) : z_j^{F2_b}(t) \geq \mu + 1.28\sigma \right\} \tag{16}$$

μ and σ denote the arithmetic mean and the standard deviation of $z_j^{F2_b}(t)$ over all $F2_b$ neurons, respectively. If the activations were normally distributed, $\mathcal{N}(t)$ would only contain those 10% of the neurons that have the highest activations.

The contribution of each neighbour to the output function is inversely proportional to the distance between its category and $\underline{x}(t)$:

$$y_j^{F2_b}(t) = \begin{cases} \dfrac{\frac{1}{1-z_j^{F2_b}(t)}}{\sum_{n \in \mathcal{N}(t)} \frac{1}{1-z_n^{F2_b}(t)}} & \text{, if } j \in \mathcal{N}(t) \\ 0 & \text{, otherwise} \end{cases} \tag{17}$$

Depending on the data distribution and for computational reasons it may be advantageous to further limit the number of considered nodes. Therefore, we incorporated an additional parameter ν which denotes the maximum cardinality of $\mathcal{E}(t)$ and $\mathcal{N}(t)$. It does not affect the underlying representations and may be changed during the application of the network. To obtain repeatable results, elements need to be added to both sets in a predefined way (cf. Eqs. 13 and 16). Therefore, we decided to add nodes in increasing order of their indices to $\mathcal{E}(t)$ and in decreasing order of their activations to $\mathcal{N}(t)$. Provided that the cardinality has reached the value of ν, the insertion of new elements is stopped. As a result, established and certain knowledge is preferred over recently acquired and uncertain knowledge. Due to this difference to the original output function (cf. Eq. 9), which treats all nodes equally, we decided to consider not only permanent nodes but also node candidates for prediction. As a result, the predictions become slightly less stable. However, the network is better adapted to recent input, in particular if no established knowledge is available.

In order to make predictions based on incomplete input vectors, the mask layer $F0_m$ is used. Its neurons, the output of which is given by the mask vector

$$\underline{m}^{F0}(t) = \left[m_1^{F0}(t), \ldots, m_d^{F0}(t) \right]^T, \tag{18}$$

inhibit the network connections that encode elements of the input vector that are not available; i.e., presented elements are characterised by a mask value $m_i^{F0}(t)$ of 0 and unknown elements by a value of 1. Hence, the indices of the relevant elements of $\underline{x}^{F1}(t)$ (cf. Eq. 2) are given by the index set

$$\mathcal{M}^0 = \big\{i, i+d : m_i^{F0}(t) = 0\big\}. \tag{19}$$

Using $\mathcal{M}^0$, the activation of the $F2$ nodes of module b can be determined solely based on the non-inhibited F1 neurons:

$$z_j^{F2_b}(t) = 1 - \frac{\sum\limits_{i \in \mathcal{M}^0} \left| \min\left(x_i^{F1}(t), w_{ji}^{F2_b}(t)\right) - w_{ji}^{F2_b}(t) \right|}{\frac{1}{2}|\mathcal{M}^0|} \tag{20}$$

If required, the maximum activation over all $F2$ nodes of module b can be applied as a measure of the degree of knowledge the network has about a certain input vector. Then, input vectors can be rejected as unknown if the maximum activation is below a threshold.

5 Results

We evaluated TopoART-C using several real-world datasets from the UCI machine learning repository [12]. In order to show the beneficial properties of TopoART-C, we selected datasets with varying numbers of classes and features, with and without missing values, and with balanced and imbalanced classes (see Table 1). Here, the class ratio denotes the ratio between the number of samples contained in the smallest class and in the largest class, respectively. Thus, a class ratio of 1 shows that a dataset is completely balanced, while class ratios close to 0 indicate imbalanced datasets.

dataset	$\lvert\Lambda\rvert$	d	missing values	class ratio
iris	3	4	no	1.000
ISOLET*	26	617	no	0.992
optical digits*	10	64	no	0.967
ozone level (1 hour)	2	73	yes	0.030
ozone level (8 hours)	2	73	yes	0.067
page blocks	5	10	no	0.006
pen digits*	10	16	no	0.921
wine	3	13	no	0.676
wine quality (red) [9]	6	11	no	0.015
wine quality (white) [9]	7	11	no	0.002
yeast	10	8	no	0.011

Table 1. Number of classes $|\Lambda|$, number of features d, existence of missing values and class ratio for the considered datasets. Those datasets marked with * contain an independent test set.

For comparison, we used several well-known on-line and off-line classifiers: the k-nearest neighbour classifier[3] (kNN), the naïve Bayes classifier[3] (NB), random trees[3] (RTs), the Simplified Fuzzy ARTMAP (SFAM) [27], and support vector machines[4] (SVMs) using different kernels. These classifiers were compared based on the harmonic mean accuracy

$$\overline{\mathrm{ACC}}_{\mathrm{hm}} = \frac{|\Lambda|}{\sum\limits_{\lambda_i \in \Lambda} \frac{1}{\mathrm{ACC}(\lambda_i)}} \tag{21}$$

as proposed in [25] for imbalanced datasets. Here, $\mathrm{ACC}(\lambda_i)$ denotes the fraction of correctly classified samples of class λ_i. In contrast to the total accuracy[5] and the arithmetic mean of the class-specific accuracies $\mathrm{ACC}(\lambda_i)$, $\overline{\mathrm{ACC}}_{\mathrm{hm}}$ prevents large correctly classified classes from dominating the classification results; e.g., if one class cannot be recognised at all, $\overline{\mathrm{ACC}}_{\mathrm{hm}}$ drops to zero, independent of the number of test samples available for this class. Provided that the classification problem is entirely balanced and the class-specific accuracies are equal, all three accuracy measures are equal.

Table 2 shows the classification results. These results were either obtained using five-fold cross-validation or refer to the independent test set that has neither been used for training nor the optimisation of model parameters before, if available (cf. Table 1). The relevant parameters of the classifiers were determined by means of grid search.[6] For training, all features were normalised to the interval $[0.05, 0.95]$. Input vectors containing missing values were ignored (training and prediction) and counted as errors (prediction) if the respective classifier was not able to process incomplete data.

TopoART-C achieved excellent results for the majority (6 of 11) of the datasets. In particular, it outperformed the other classifiers on 4 of 6 imbalanced[7] datasets including those with missing values and reached comparatively high accuracies for the remaining two datasets 'wine quality (red)' and 'wine quality (white)'. Regarding balanced data, SVMs performed better, especially on the 'ISOLET' dataset, which is most likely caused by its large number of features. Nevertheless, TopoART-C achieved the maximum accuracy on 2 of 5 balanced datasets. In addition, TopoART-C often reached very high accuracies after a single presentation of all training samples, which is a good benchmark for incremental on-line learning; without the capability of stable incremental learning, an on-line learning approach such as TopoART (cf. Eqs. 5 and 6) would be prone to catastrophic forgetting resulting in worse results.

6 Conclusion and Outlook

We presented the novel incremental classifier TopoART-C that is capable of fast on-line learning (cf. Table 2). Accurate predictions can even be made if the input vectors contain missing values. As TopoART-C contains the unsupervised TopoART network as major learning component, it is insensitive to noise (cf. Fig. 2) and can be applied to non-stationary data (cf. Fig. 3). These properties of TopoART-C make it an excellent choice for the application to real-world on-line learning tasks, as they occur, for instance, in cognitive robotics. In addition, the clusters learnt by the TopoART subnet could provide additional information on the underlying data.

Furthermore, TopoART-C is not restricted to the usage of TopoART; alternative neural networks with a TopoART-like structure such as Hypersphere TopoART [24] can be applied as well. Since Hypersphere TopoART does not perform complement coding, the resulting classifier could process arbitrarily scaled values even if their range is not completely known in advance.

[3] implemented in OpenCV v2.2 [3]

[4] implemented in LIBSVM v3.11 [7]

[5] overall ratio of correctly classified samples

[6] **kNN:** $k \in \{1, 2, \ldots, 25\}$; **RTs:** `use_surrogates` $\in \{$`false`, `true`$\}$, `variableImportance` $\in \{$`false`, `true`$\}$, `nactive_vars` $= d^x$ with $x \in [0.1, 0.9]$ and step size 0.1; **SFAM:** $\rho \in [0.75, 1]$ with step size 0.01, $\beta \in [0.2, 1]$ with step size 0.2; **SVMs:** $C = 10^x$ with $x \in [-4, 4]$ and step size 0.5, $\gamma = 10^x$ with $x \in [-4, 4]$ and step size 0.5 (RBF kernel), `coef0` $= 10^x$ with $x \in [-4, 4]$ and step size 0.5 (polynomial kernel), degree $\in \{1, 2, 3, 4, 5\}$ (polynomial kernel); **TopoART-C**: $\rho_a \in [0.75, 1]$ with step size 0.01, $\beta_{sbm} \in [0, 1]$ with step size 0.2, $\phi \in \{1, 2, 3, 4, 5\}$, $\nu \in \{1, 2, \ldots, 25\}$

[7] class ratio < 0.1

dataset	kNN^{o}	NB	RTs^{tp}	SFAM^{o}		SVMs		TopoART-C^{op}	
	1 it.			1 it.	$\leq$25 it.	polynomial	RBF	1 it.	$\leq$25 it.
iris	97.4±2.7	97.0±2.8	96.7±2.4	97.0±2.0	96.9±3.0	98.4±2.0	96.8±3.1	**98.9±1.4**	98.2±2.7
ISOLET	90.9	0.0	94.5	92.6	93.9	96.3	**96.7**	91.2	91.6
optical digits	97.9	94.6	96.9	96.9	97.7	97.5	**98.4**	97.4	97.4
ozone level (1 hour)	29.5±10.7	0.0±0.0	2.2±4.3	42.8±11.6	42.7±11.6	34.1±6.5	26.3±14.4	50.6±27.5	**54.4±22.1**
ozone level (8 hours)	45.0±6.8	5.5±4.8	10.7±6.9	50.1±9.1	53.7±6.7	50.2±4.1	45.6±4.2	**63.9±3.8**	60.6±10.3
page blocks	69.6±6.3	79.6±6.0	**82.5±4.1**	75.4±3.4	75.5±3.3	77.9±3.0	76.4±6.5	**82.5±4.0**	**82.5±6.5**
pen digits	97.7	95.8	96.4	97.4	97.9	97.5	**98.2**	97.7	97.7
wine	96.6±3.1	98.8±1.5	98.8±1.5	98.7±1.7	98.7±1.7	98.3±1.6	98.8±1.0	98.7±1.7	**99.1±1.7**
wine quality (red)	0.0±0.0	0.0±0.0	0.0±0.0	15.8±20.1	**15.9±20.2**	0.0±0.0	0.0±0.0	9.1±18.2	9.3±18.6
wine quality (white)	0.0±0.0	5.6±11.2	0.0±0.0	6.2±12.3	0.0±0.0	2.2±4.4	**8.5±16.9**	7.3±14.5	7.7±15.5
yeast	11.4±14.0	0.0±0.0	0.0±0.0	24.9±20.9	24.4±20.8	7.3±14.5	11.7±23.4	**37.7±19.5**	36.7±19.7

Table 2. Harmonic mean accuracies and their standard deviations (over the cross-validation runs) in percent. The best results for each dataset are highlighted. In order to alleviate the comparison, some relevant capabilities of the classifiers are indicated by superscripts: o = on-line learning, t = accept missing values for training, and p = accept missing values for prediction. In order to compensate for the negative effects of a possibly too small number of training steps, the respective training sets were presented to the on-line learning approaches except for the kNN classifier up to 25 times. The results are given for the first iteration (1 it.) and when they converged or the maximum number of iterations was reached ($\leq$25 it.).

ACKNOWLEDGEMENTS

This work was partially funded by the German Research Foundation (DFG), Excellence Cluster 277 "Cognitive Interaction Technology".

REFERENCES

[1] Elmar Berghöfer, Denis Schulze, Marko Tscherepanow, and Sven Wachsmuth, 'ART-based fusion of multi-modal information for mobile robots', in *Proceedings of the International Conference on Engineering Applications of Neural Networks*, volume 363 of *IFIP AICT*, pp. 1–10, Corfu, Greece, (2011). Springer.

[2] Antoine Bordes, Seyda Ertekin, Jason Weston, and Léon Bottou, 'Fast kernel classifiers with online and active learning', *Journal of Machine Learning Research*, **6**, 1579–1619, (2005).

[3] Gary Bradski and Adrian Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*, O'Reilly, 2008.

[4] Gail A. Carpenter, Stephen Grossberg, and John H. Reynolds, 'ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network', *Neural Networks*, **4**, 565–588, (1991).

[5] Gail A. Carpenter, Stephen Grossberg, and David B. Rosen, 'Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system', *Neural Networks*, **4**, 759–771, (1991).

[6] Gert Cauwenberghs and Tomaso Poggio, 'Incremental and decremental support vector machine learning', in *Neural Information Processing Systems*, pp. 409–415, (2000).

[7] Chih-Chung Chang and Chih-Jen Lin, 'LIBSVM: A library for support vector machines', *ACM Transactions on Intelligent Systems and Technology*, **2**(3), 27:1–27:27, (2011). Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[8] Corinna Cortes and Vladimir Vapnik, 'Support-vector networks', *Machine Learning*, **20**, 273–297, (1995).

[9] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis, 'Modeling wine preferences by data mining from physicochemical properties', *Decision Support Systems*, **47**(4), 547–553, (2009).

[10] Ryan Elwell and Robi Polikar, 'Incremental learning of concept drift in nonstationary environments', *IEEE Transactions on Neural Networks*, **22**(10), 1517–1531, (2011).

[11] Scott E. Fahlman and Christian Lebiere, 'The cascade-correlation learning architecture', in *Neural Information Processing Systems*, pp. 524–532, (1989).

[12] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[13] Bernd Fritzke, 'A growing neural gas network learns topologies', in *Neural Information Processing Systems*, pp. 625–632, (1994).

[14] Shen Furao and Osamu Hasegawa, 'An incremental network for on-line unsupervised classification and topology learning', *Neural Networks*, **19**, 90–106, (2006).

[15] Stephen Grossberg, 'Competitive learning: From interactive activation to adaptive resonance', *Cognitive Science*, **11**, 23–63, (1987).

[16] Fred H. Hamker, 'Life-long learning cell structures—continuously learning without catastrophic interference', *Neural Networks*, **14**, 551–573, (2001).

[17] Haibo He, Sheng Chen, Kang Li, and Xin Xu, 'Incremental learning from stream data', *IEEE Transactions on Neural Networks*, **22**(12), 1901–1914, (2011).

[18] Marc Kammer, Marko Tscherepanow, Thomas Schack, and Yukie Nagai, 'A perceptual memory system for affordance learning in humanoid robots', in *Proceedings of the International Conference on Artificial Neural Networks*, volume 6792 of *LNCS*, pp. 349–356. Springer, (2011).

[19] Stephan Kirstein and Heiko Wersing, 'A biologically inspired approach for interactive learning of categories', in *Proceedings of the International Conference on Development and Learning*, pp. 1–6. IEEE, (2011).

[20] Matej Kristan, Aleš Leonardis, and Danijel Skočaj, 'Multivariate online kernel density estimation with Gaussian kernels', *Pattern Recognition*, **44**(10–11), 2630–2642, (2011).

[21] Yann Prudent and Abdellatif Ennaji, 'An incremental growing neural gas learns topologies', in *Proceedings of the International Joint Conference on Neural Networks*, volume 2, pp. 1211–1216. IEEE, (2005).

[22] Frank Rosenblatt, 'The perceptron: A probabilistic model for information storage and organization in the brain', *Psychological Review*, **65**(6), 386–408, (1958).

[23] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams, 'Learning internal representations by error propagation', in *Parallel Distributed Processing – Explorations in the Microstructure of Cognition*, volume 1, 318–362, MIT Press, seventh edn., (1988).

[24] Marko Tscherepanow, 'Incremental on-line clustering with a topology-learning hierarchical ART neural network using hyperspherical categories', in *Poster and Industry Proceedings of the Industrial Conference on Data Mining*, pp. 22–34. ibai-publishing, (2012).

[25] Marko Tscherepanow, Nickels Jensen, and Franz Kummert, 'An incremental approach to automated protein localisation', *BMC Bioinformatics*, **9**(445), (2008).

[26] Marko Tscherepanow, Marko Kortkamp, and Marc Kammer, 'A hierarchical ART network for the stable incremental learning of topological structures and associations from noisy data', *Neural Networks*, **24**(8), 906–916, (2011).

[27] Mohammad-Taghi Vakil-Baghmisheh and Nikola Pavešić, 'A fast simplified fuzzy ARTMAP network', *Neural Processing Letters*, **17**(3), 273–316, (2003).

[28] Andrew R. Webb and Keith D. Copsey, *Statistical Pattern Recognition*, Wiley, third edn., 2011.

[29] James R. Williamson, 'Gaussian ARTMAP: a neural network for fast incremental learning of noisy multidimensional maps', *Neural Networks*, **9**(5), 881–897, (1996).

Episodic Clustering of Data Streams Using a Topology-Learning Neural Network

Marko Tscherepanow[1,2] and **Sina Kühnel**[2,3] and **Sören Riechers**[1,2]

Abstract. In this paper, an extension of the unsupervised topology-learning TopoART neural network is presented. Like TopoART, it is capable of stable incremental on-line clustering of real-valued data. However, it incorporates temporal information in such a way that consecutive input vectors with a low distance in the input space are summarised to episode-like clusters. Inspired by natural memory systems, we propose two recall methods enabling the selection and retrieval of these episodes. They are demonstrated at the example of a video stream recorded in a natural environment.

1 Introduction

Incremental on-line learning is a branch of machine learning and artificial intelligence that has been gaining increasing interest over the recent years (e.g., [2], [7], [10], [14], and [20]). In contrast to traditional models requiring distinct training, validation, and test phases, such approaches allow for a continuous extension of existing knowledge while they are already in application. Hence, they are particularly useful for tasks involving incomplete knowledge or non-stationary data distributions such as the representation of visual [10] and multi-modal [2] categories in robotic scenarios or dynamic topic mining [12]. These methods process incoming data sample by sample; i.e., in a temporal order. However, this aspect is barely accounted for, although it might provide additional information.

Humans and animals exploit this information in a natural way. Consequently, sequence learning is considered as "the most prevalent form of human and animal learning" [16, p. 67]. This is reflected by the formed representations, which relates to the vast research area of memory.

Memory has been classified along time (short-term, long-term), processes (encoding, storage, retrieval) as well as regarding stored content (procedural, perceptual, semantic, episodic) [13]. Episodic memory is the highest developed system that allows us to remember our own past in detail. During encoding of episodic memory information from sensory systems, semantic knowledge as well as perceptual and procedural information is connected to one coherent event. This complex event knowledge allows us to perform mental time travel when remembering past experiences [21]. The importance of episodic memory becomes clearer when evolutionary traits and today's requirements are taken into account. When interacting in social situations we rely strongly on our ability to encode semantic as well as episodic memories of events. For example, established impressions of people and situations can be reevaluted and updated over time [11].

From the machine learning perspective, Sun and Gilles [16] distinguish between four major categories of sequence learning approaches: sequence prediction, sequence generation, sequence recognition, and sequential decision making. Some popular approaches are content-addressable memories for temporal patterns [3] (sequence generation), echo state networks [8] (sequence prediction and generation), hidden Markov models [15] (sequence prediction, generation, and recognition), as well as reinforcement learning [17] (sequential decision making). Unsupervised vector quantizers for time series such as Recursive Self-Organizing Maps (RSOMs) [22] constitute a further approach to sequence learning: they learn a mapping from subsequences to prototype sequences.

The approaches mentioned above deal with sequential data within a limited time frame. They do not have an absolute representation of time. Consequently, retrieval of past sequences as performed by natural memory systems is not possible. Furthermore, they are limited by a predefined model structure and capacity.

Machine learning approaches including those dedicated to sequence learning have been frequently employed so as to develop artificial memory systems. The CLARION architecture [6] possessing procedural and semantic memory components and the memory of the humanoid robot ISAC [9] comprising short-term and long-term memory components for processing procedural, perceptual, semantic, and episodic data are two examples for complex artificial memory systems. In addition, specific aspects of natural memory systems such as consolidation processes for procedural learning [1] and the categorisation of perceptual patterns [6] have been emulated.

In this paper, we present a novel approach to incremental on-line clustering (see Section 3) which incorporates temporal information for the life-long learning of episode-like clusters. In addition to the common prediction functionality, two recall methods for retrieving learnt information and reconstructing past episodes based on these clusters are proposed. As our approach originates from the TopoART neural network (see Section 2) [18][20], it inherits its capabilities of fast and stable on-line clustering of possibly noisy or non-stationary data. Therefore, we call our approach Episodic TopoART. In Section 4, we demonstrate the recall methods and show that the inclusion of temporal information may be advantageous given that input is provided in a meaningful temporal order.

2 TopoART

Adaptive Resonance Theory (ART) neural networks learn top-down expectations which are matched with bottom-up input. These expectations, which encode different regions of the input space, are called

[1] Applied Informatics, Faculty of Technology, Bielefeld University, Germany, email: marko@techfak.uni-bielefeld.de, marko@tscherepanow.de
[2] CITEC, Cognitive Interaction Technology, Center of Excellence, Bielefeld University, Germany
[3] Physiological Psychology, Faculty of Psychology and Sport Sciences, Bielefeld University, Germany

categories. Their maximum size is controlled by the vigilance parameter ρ.

TopoART (TA) [18][20] is an Adaptive Resonance Theory (ART) neural network consisting of two modules called TA a and TA b.[4] These modules are closely related to Fuzzy ART [4]. They have a three-layered structure and the input layer $F0$ is shared by them (see Fig, 1). Input to TA b is filtered by TA a, which renders the network insensitive to noise.

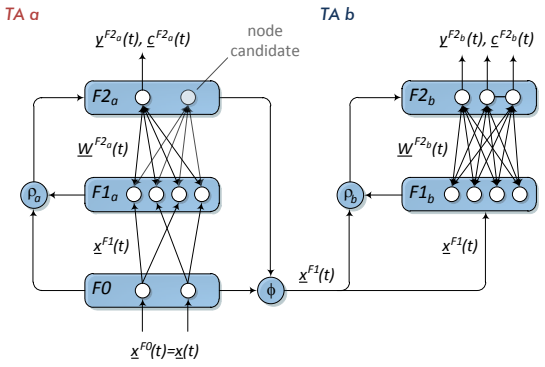


Figure 1. TopoART architecture. TopoART networks comprise two modules (TA a and TA b) sharing the input layer $F0$.

As TopoART is an incremental neural network that can be trained on-line, training and prediction steps can be mixed arbitrarily. In both cases, input is presented in discrete time steps t. Each input vector $\underline{x}(t)$ consists of d real-valued elements $x_i(t)$:

$$\underline{x}(t) = \big[x_1(t), \ldots, x_d(t)\big]^T. \tag{1}$$

In the $F0$ layer, $\underline{x}(t)$ is complement coded. The resulting vector

$$\underline{x}^{F1}(t) = \big[x_1(t), \ldots, x_d(t), 1-x_1(t), \ldots, 1-x_d(t)\big]^T \tag{2}$$

is propagated to the respective $F1$ layer. Due to the usage of complement coding, each of the elements $x_i(t)$ has to lie in the interval $[0, 1]$.

2.1 Training

During training, $\underline{x}^{F1}(t)$ is first propagated to the $F2$ layer of TA a where the neurons (also called nodes) are activated (choice function):

$$z_j^{F2}(t) = \frac{\big\|\underline{x}^{F1}(t) \wedge \underline{w}_j^{F2,s}(t)\big\|_1}{\alpha + \big\|\underline{w}_j^{F2,s}(t)\big\|_1} \quad \text{with} \quad \alpha = 0.001. \tag{3}$$

The activation $z_j^{F2}(t)$ measures the similarity between $\underline{x}^{F1}(t)$ and the category of node j, which is encoded in the weight vector $\underline{w}_j^{F2,s}(t)$. $\wedge$ denotes an element-wise minimum operation.

In addition, a match value

$$\zeta_j^{F2,s}(t) = \frac{\big\|\underline{x}^{F1}(t) \wedge \underline{w}_j^{F2,s}(t)\big\|_1}{\big\|\underline{x}^{F1}(t)\big\|_1} \tag{4}$$

is computed for all $F2$ nodes j. It constitutes a measure for the size of the extended category that includes $\underline{x}^{F1}(t)$.

[4] In general, the number of modules must be larger than or equal to 1.

The maximum category size $S^{\max}$ depends on the dimensionality of the input space d and the vigilance parameter ρ:

$$S^{\max} = d(1-\rho). \tag{5}$$

Therefore, the weights $\underline{w}_j^{F2,s}(t)$ of an $F2$ node j are only allowed to be adapted if

$$\zeta_j^{F2,s} \geq \rho. \tag{6}$$

In order to learn a new input vector, the nodes with the highest and the second highest activation while fulfilling Eq. 6 (match function) are sought. They are referred to as the best-matching node (bm) and the second-best-matching node (sbm), respectively. Only the weights of these two neurons are adapted:

$$\begin{aligned} \underline{w}_j^{F2,s}(t+1) \quad = \quad & \beta_j\big(\underline{x}^{F1}(t) \wedge \underline{w}_j^{F2,s}(t)\big) \\ & +(1-\beta_j)\underline{w}_j^{F2,s}(t) \\ & \text{with } j \in \{bm, sbm\} \text{ and } \beta_{bm} = 1. \end{aligned} \tag{7}$$

In addition to its weight vector, each $F2$ node j possesses a counter n_j which is incremented whenever its weights are adapted. Furthermore, if two neurons bm and sbm that fulfil Eq. 6 were found, they are connected by an edge so as to learn the topological structure of the input data.

In order to reduce the sensitivity to noise, all $F2$ nodes with $n_j < \phi$ including their edges are removed every τ learning cycles.[5] Therefore, such nodes are called node candidates. If $n_j \geq \phi$, node j is permanent.

TA b learns in an identical way like TA a using a higher value for its vigilance parameter ρ_b:

$$\rho_b = \frac{1}{2}(\rho_a + 1). \tag{8}$$

In addition, $\underline{x}^{F1}(t)$ is only propagated to TA b if the best-matching node of TA a is permanent. As a consequence, TA b learns a refined clustering which is less prone to noise.

If the respective $F2$ layer does not contain any node yet or no node fulfilling Eq. 6 could be found, a new $F2$ node with $\underline{w}_{new}^{F2,s}(t+1) = \underline{x}^{F1}(t)$ is incorporated.

2.2 Prediction

During prediction steps, learnt cluster labels are associated to unknown input. After complement coding, presented input vectors are directly propagated to both modules. Here, the nodes of the respective $F2$ layer are activated using a modified activation function:

$$z_j^{F2}(t) = 1 - \frac{\Big\|\big(\underline{x}^{F1}(t) \wedge \underline{w}_j^{F2,s}(t)\big) - \underline{w}_j^{F2,s}(t)\Big\|_1}{d}. \tag{9}$$

In contrast to Eq. 3, Eq. 9 is independent of the category size.

After activation, the node with the highest activation is chosen as the best-matching node bm of the respective module. The match function is not checked. Then, both modules provide an output vector $\underline{y}^{F2}(t)$ with

$$y_j^{F2}(t) = \begin{cases} 0 & \text{if } j \neq bm \\ 1 & \text{if } j = bm \end{cases} \tag{10}$$

[5] The learning cycles are individually counted for each module.

and a clustering vector $\underline{c}^{F2}(t)$ containing the cluster labels of the $F2$ neurons. These cluster labels are determined by a labelling algorithm assigning unique integer labels to connected components of $F2$ nodes.

3 Episodic TopoART

Episodic TopoART (ETA) contains a TopoART network as a major learning component (see Fig. 2). This component is extended in order to enable the encoding and the retrieval of information within its spatio-temporal context.

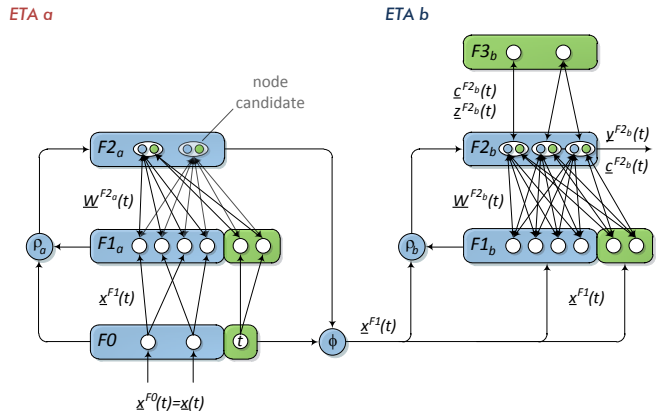


Figure 2. Structure of Episodic TopoART networks. Like TopoART, Episodic TopoART consists of two modules sharing a common input layer $F0$. The structures adopted from TopoART (blue) are extended by neurons representing temporal information and an additional layer required for recall (green).

Due to the structural similarities of TopoART and Episodic TopoART, both networks can easily be substituted by each other, e.g., in order to serve as components of more complex networks fulfilling alternative tasks such as the supervised TopoART-R network [19]. However, there are several functional differences between Episodic TopoART and TopoART, which are explained in the following. In addition to the training and prediction steps known from TopoART, Episodic TopoART provides a more complex recall functionality.

3.1 Training

The input of Episodic TopoART networks is equal to TopoART; i.e., individual input vectors $\underline{x}(t)$ comprise d elements $x_i(t)$. $\underline{x}(t)$ is complement coded and propagated to the respective $F1$ layer. Thus, Episodic TopoART is able to learn spatial relations of samples in the input space like TopoART and it requires a normalisation of all $x_i(t)$ into the interval $[0, 1]$.

In addition to the nodes representing the current input, the $F0$ layer of Episodic TopoART networks contains a single node representing the current time step $t{=}t^{F0}(t)$. It reflects the total number of performed training steps. Its actual value is not crucial as long as it is incremented by 1 after each training step. Therefore, it constitutes a subjective, internal representation of time.

The problem with clustering temporal data in conjunction with presented input consists in the different characteristics of this information. While the elements of the input vector $\underline{x}(t)$ are real-valued and normalised, $t^{F0}(t)$ is a positive integer value which is strictly increasing during learning and not bounded. Therefore, it is not possible to use complement coding for $t^{F0}(t)$. However, the effects of complement coding can be emulated. In particular, $\underline{x}^{F1}(t)$ corresponds to a category comprising only $\underline{x}(t)$ as a single point in the input space where $x_i(t)$ and $x_{i+d}(t)$ encode for the lower and upper bounds along dimension i, respectively. During learning, a category grows; i.e., it spans a certain range along different dimensions. Regarding $t^{F0}(t)$, a similar effect is achieved by the following encoding:

$$\underline{t}^{F1}(t) = \left[t_1^{F1}(t), t_2^{F1}(t)\right]^T. \tag{11}$$

Here, $t_1^{F1}(t)$ encodes the minimum time step and $t_2^{F1}(t)$ the maximum time step that is represented. For an individual sample, both values are equal to $t^{F0}(t)$.

Due to the different type of information processed, all $F2$ nodes j have two types of weights: the spatial weights $\underline{w}_j^{F2,s}(t)$ adopted from TopoART and the temporal weights

$$\underline{w}_j^{F2,t}(t) = \left[w_{j,1}^{F2,t}(t), w_{j,2}^{F2,t}(t)\right]^T. \tag{12}$$

Like in TopoART networks, the activation of the $F2$ nodes is computed according to Eq. 3; i.e., it reflects only spatial similarities. However, an additional temporal match value

$$\zeta_j^{F2,t}(t) = \frac{t^{max} - \min\left(t_2^{F1}(t) - w_{j,1}^{F2,t}(t), t^{max}\right)}{t^{max}} \tag{13}$$

is computed in order to incorporate temporal information in the learning process. The match values $\zeta_j^{F2,s}(t)$ and $\zeta_j^{F2,t}(t)$ are combined in a new match function:

$$\zeta_j^{F2,s} \geq \rho \quad \text{and} \quad \zeta_j^{F2,t} \geq \rho. \tag{14}$$

As a result, the $F2$ nodes represent spatial similarities which were encountered within a certain time frame bounded by t^{max}.

Using Eq. 14 for the processing of data streams causes a new problem. As explained in Section 2.1, edges are added between two nodes fulfilling the match function. However, if input is arriving as a data stream and temporal information is also considered, the overlap of categories is less probable, since new nodes are only added if no existing node can fulfil Eq. 14. Hence, the chance to find two nodes fulfilling the match function is considerably smaller. As a result, categories belonging to a cluster cannot be connected. Therefore, nodes need to be added earlier utilising a stricter match function for the determination of the best-matching nodes (cf. Eqs. 5 and 6):

$$\zeta_j^{F2,s} \geq \frac{1}{2}(\rho + 1) \quad \text{and} \quad \zeta_j^{F2,t} \geq \frac{1}{2}(\rho + 1). \tag{15}$$

If new input is to be learnt and the $F2$ nodes have been activated, the node with the highest activation while fulfilling Eq. 15 is determined. If such a node can be found, it becomes the best-matching node bm. Otherwise, a new node with $\underline{w}_{new}^{F2,s}(t){=}\underline{x}^{F1}(t)$ and $\underline{w}_{new}^{F2,t}(t){=}\underline{t}^{F1}(t)$ is added. This new node automatically fulfils Eq. 15 and, therefore, becomes the new best-matching node.

Afterwards, a second-best-matching node is sought. Here, Eq. 14 is applied as match function; i.e., the unmodified value of the respective vigilance parameter (ρ_a or ρ_b) is used. Hence, the categories can reach the same size in the input space as with the original TopoART. Furthermore, nodes rejected as best-matching nodes before can still become the second-best-matching node.

The spatial weights $\underline{w}_j^{F2,t}(t)$ and the temporal weight $w_{j,2}^{F2,t}(t)$ of the nodes bm and sbm are adapted according to Eq. 7. However, $w_{j,1}^{F2,t}(t)$ remains constant once a node has been created, as the time step t is strictly increasing and $w_{j,1}^{F2,t}(t)$ denotes the lower temporal bound of the respective category.

Like in the original algorithm, node candidates are removed every τ learning cycles, ETA b is trained in an identical way to ETA a using a vigilance value of ρ_b according Eq. 8, and input to ETA b is filtered by ETA a. As a result, ETA b learns a refined and noise-reduced clustering. Therefore, the output of ETA a is neglected for recall; the main function of this module consists in directing the attention of the network to relevant areas of the input space (cf. [19]).

3.2 Prediction of Cluster Labels

The prediction of cluster labels is performed in an identical way to TopoART (see Section 2.2). Temporal information is completely neglected and $t^{F0}(t)$ is not incremented. However, the formed clusters reflect the spatio-temporal relationships encountered during training; i.e., each cluster summarises similar samples which were learnt in close succession.

3.3 Recall of Spatio-Temporal Relationships

For recall, the formed clusters are interpreted as episodes, as they represent related input vectors (stimuli) in their temporal order. To recall information within the respective spatio-temporal context, Episodic TopoART distinguishes between two principal procedures: inter-episode recall and intra-episode recall. While inter-episode recall provides access to different episodes comprising stimuli similar to the presented input, intra-episode recall reconstructs episodes starting from a time step when a stimulus similar to the presented input vector was observed. Like the prediction mechanism, both procedures require that the $F2$ nodes of ETA b have activated according to Eq. 9 and labelled.

3.3.1 Inter-Episode Recall

The procedure for inter-episode recall is strongly related to the iterative recall procedure used by TopoART-AM [20] for recalling associations between real-world associative keys. However, TopoART-AM is not able to account for temporal relationships.

The actual recall mechanism is realised by the temporary $F3$ layer of ETA b. It is created after a stimulus has been presented. Each node of this layer represents an individual episode and is connected to all of its $F2$ nodes. The activation

$$z_l^{F3}(t) = \max_{j, c_j^{F2_b}(t)=l} z_j^{F2}(t) \tag{16}$$

of an $F3$ node l is equal to the maximum activation of the connected $F2$ nodes; i.e., it is a measure for the similarity of the presented stimulus with this episode. After the activation of the $F3$ nodes, the iterative recall process is initiated:

1. set iteration counter i to 1
2. find the $F3$ node r_i with the highest activation
3. inhibit all $F2$ nodes j with $z_j^{F2}(t) < z_{r_i}^{F3}(t)$ which are connected to r_i
4. find the $F2$ node bm_i with the highest activation within the current episode
5. return the output vector $\underline{y}(t,i)$ of the current iteration i
6. reset r_i $\left(z_{r_i}^{F3}(t) = -1\right)$
7. increment i
8. start next iteration (go to step 2)

The recall process either stops if all $F3$ nodes have been reset or a desired number of recall steps has been performed. Afterwards, the $F3$ layer is removed.

The output vector $\underline{y}(t,i)$ is computed as the centre of gravity of the respective best-matching category bm_i:

$$\underline{y}(t,i) = \frac{1}{2} \begin{bmatrix} w_{bm_i,1}^{F2,s}(t) & + & 1 - w_{bm_i,d+1}^{F2,s}(t) \\ \vdots & & \vdots \\ w_{bm_i,d}^{F2,s}(t) & + & 1 - w_{bm_i,2d}^{F2,s}(t) \end{bmatrix} \tag{17}$$

3.3.2 Intra-Episode Recall

Intra-episode recall requires an $F2$ node j as a starting point. For example, the best-matching nodes bm_i determined by means of inter-episode recall can be applied here.

After a suitable $F2$ node j has been chosen, the temporal order of all its topological neighbours n is analysed. Those nodes which were created after j, i.e. $w_{n,1}^{F2,t}(t) > w_{j,1}^{F2,t}(t)$, are put into the set $\mathcal{N}^+(j)$. Then, a best-matching node bm_i is computed as

$$bm_i = \arg \max_{n \in \mathcal{N}^+(j)} \left\| \underline{w}_j^{F2,t}(t) - \underline{w}_n^{F2,t}(t) \right\|_1. \tag{18}$$

Like with inter-episode recall, Eq. 17 is used, to generate an output for bm_i. Afterwards, bm_i is used as the starting node for the next intra-episode recall cycle. The recall process is stopped, if $\mathcal{N}^+(j) = \emptyset$. In this case, one possible end of the episode has been reached.

4 Results

We conducted two different experiments in order to analyse Episodic TopoART. First, we compared the prediction results of TopoART[6] and Episodic TopoART using a synthetic dataset (see Section 4.1). Then, we investigated its prediction and recall capabilities by means of real-world video data (see Section 4.2).

4.1 Synthetic Data

For the first experiment, we employed the well-known two spiral dataset (see Fig. 3a) [5]. It consists of two intertwined spirals comprising 97 points each. For validation, we randomly determined 250 additional samples for each spiral (see Fig. 3b). During training, both spirals were presented one after another. Furthermore, the samples of each spiral were presented with increasing radius. Thereby, both spirals can be considered as two consecutive episodes.

The clustering results for TopoART and Episodic TopoART are shown in Figs. 3c–f. The parameters ρ, β_{sbm}, and ϕ of both approaches were obtained by grid search using the validation dataset. Here, the Rand index R [23] for separating both spirals into two distinct clusters/episodes was maximised. Based on previous experiments (e.g., in [19]), τ was set to 200. As the new parameter t^{max} of Episodic TopoART denotes a time frame like τ, t^{max} was also set to 200. Each training sample was only presented once to each network.

[6] LibTopoART (version 0.37), available at `www.LibTopoART.eu`

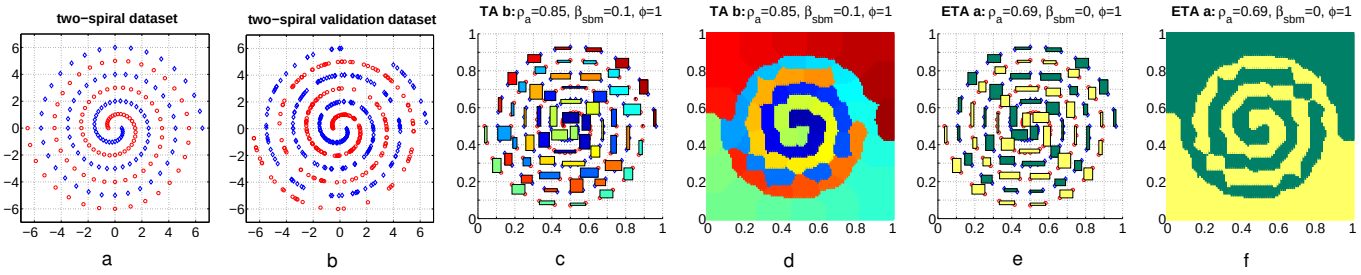


Figure 3. Clustering results for the two-spiral problem. The categories formed after training with the two-spiral dataset (a) are depicted as coloured rectangles (c and e). Here, categories connected to the same cluster share a common colour. In addition, the cluster labels were predicted for 101×101 equidistant test points distributed in the entire input space (d and f).

Figure 3 shows that both neural networks were able to learn the training samples after a single presentation. While Episodic TopoART correctly created two clusters corresponding to the two spirals (see Figs. 3e and 3f), TopoART created numerous clusters (see Figs. 3c and 3d), since its categories could not be linked appropriately. Furthermore, some categories enclose samples from both spirals. We therefore conclude that the inclusion of temporal information and the modified learning mechanism of Episodic TopoART supported the clustering process.

4.2 Real-World Data

In order to examine Episodic TopoART under more realistic conditions, we recorded a video stream by cycling with a mountain bike in the Teutoburg Forest. In comparison to indoor scenarios, outdoor environments are less structured and more diverse. In addition, they probably had a higher impact on human and animal evolution. The experimental setup and the generation of training data is explained in Fig. 4.

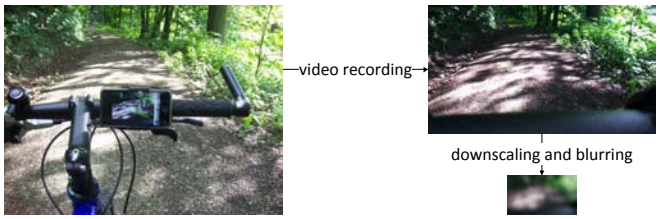


Figure 4. Experimental setup. An iPod touch 4G mounted on the handlebar of a mountain bike was used for recording image sequences in HD 720p with 30 frames per second. These images were downscaled to 64×36 pixels and subjected to a Gaussian blur (11×11 pixels). Images with even indices were used for training, while the remaining images were reserved for test purposes.

After preprocessing, a total of $29,747$ training and $29,747$ test images was available. Several Episodic TopoART networks were trained in a systematic way. However, to our knowledge there is no commonly accepted quality criterion for evaluating episodes. Rather, episodes formed based on the same event may even differ between different persons. Therefore, we resorted to manual evaluation. Figure 5 depicts the assignment of episodes[7] to video scenes computed by two different networks.

[7] prediction for the test images

Figure 5. Assignment of episodes. The video is represented by 100 test images taken at every 600th time step (left to right and top to bottom). The colour bars over each image visualise the assignment of episodes for two different networks (top bar: ρ_a=0.5; bottom bar: ρ_a=0.7; remaining parameters for both networks: β_{sbm}=0.25, ϕ=5, τ=200, t^{max}=400). Each episode is denoted by an individual colour.

Here, it needs to be emphasised that the episode length is not predefined. Rather, episodes are split if the input vectors differ considerably[8] for a longer[9] time interval. This is reflected by Fig. 5. While two episodes suffice to group the presented test images for ρ_a=0.5, the episodes are refined for ρ_a=0.7. In particular, Episodic TopoART formed a reasonable set of episodes with episode changes mainly caused by visible scene changes for ρ_a=0.7. A further increase of ρ_a would result in a higher number of created episodes. Hence, higher values of ρ_a result in a decline of the average episode length.

The more complex recall functionality of Episodic TopoART is demonstrated in Fig. 6. Here, the network trained with ρ_a=0.7 for the previous experiment (cf. Fig. 5) was used again.

Figure 6 demonstrates that the similarity of the test stimulus to the episodes provided by inter-episode recall decreases with each itera-

[8] defined by ρ_a, see Eqs. 4 and 14
[9] defined by ρ_a and t^{max}, see Eqs. 13 and 14

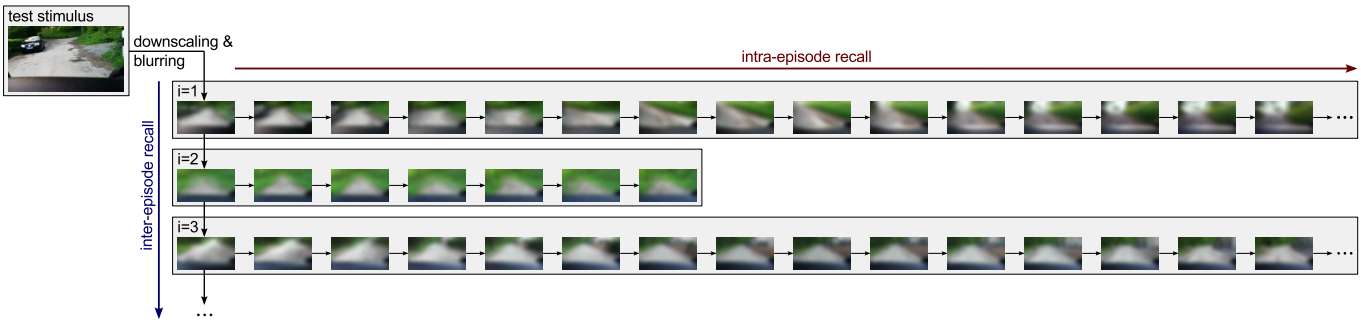


Figure 6. Recall functionality of Episodic TopoART. An exemplary test image was applied as a stimulus for initiating the inter-episode recall process. It originates from the last 10% of the video, between the second and the third image in the bottom line of Fig. 5. In each iteration i, the node bm_i was further used as the starting point for intra-episode recall. The recall results are limited to the first three iterations of the inter-episode recall algorithm and a maximum of 14 cycles for each intra-episode recall call.

tion. In this example, the best matching node bm_1 of the first iteration encodes the correct episode. The reconstruction of this episode by means of intra-episode recall shows how the input changed from the second image to the third image in the bottom line of Fig. 5.

5 Conclusion

We extended the TopoART neural network in such a way that it can create spatio-temporal representations of presented input vectors. In particular, input is grouped in episode-like clusters which can be accessed by two novel recall methods. Furthermore, the modified training procedure may be superior to TopoART provided that the input is presented in a meaningful temporal order. In the future, additional recall methods could be developed, in particular for intra-episode recall, as each episode is an undirected graph, which can be traversed in numerous ways. In addition, multi-modal data and semantic information could be applied in order to create episodes being more similar to their natural counterparts.

ACKNOWLEDGEMENTS

This work was partially funded by the German Research Foundation (DFG), Excellence Cluster 277 “Cognitive Interaction Technology”.

REFERENCES

[1] Heni Ben Amor, Shuhei Ikemoto, Takashi Minato, Bernhard Jung, and Hiroshi Ishiguro, ‘A neural framework for robot motor learning based on memory consolidation’, in *Proceedings of the International Conference on Adaptive and Natural Computing Algorithms*, volume 4432 of *LNCS*, pp. 641–648. Springer, (2007).

[2] Elmar Berghöfer, Denis Schulze, Marko Tscherepanow, and Sven Wachsmuth, ‘ART-based fusion of multi-modal information for mobile robots’, in *Proceedings of the International Conference on Engineering Applications of Neural Networks*, volume 363 of *IFIP AICT*, pp. 1–10, Corfu, Greece, (2011). Springer.

[3] J. Buhmann and K. Schulten, ‘Noise-driven temporal association in neural networks.’, *Europhysics Letters*, **4**(10), 1205–1209, (1987).

[4] Gail A. Carpenter, Stephen Grossberg, and David B. Rosen, ‘Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system’, *Neural Networks*, **4**, 759–771, (1991).

[5] Stephan K. Chalup and Lukasz Wiklendt, ‘Variations of the two-spiral task’, *Connection Science*, **19**(2), 183–199, (2007).

[6] Sylvain Chartier, Gyslain Giguère, and Dominic Langlois, ‘A new bidirectional heteroassociative memory encompassing correlational, competitive and topological properties’, *Neural Networks*, **22**(5–6), 568–578, (2009).

[7] Shen Furao and Osamu Hasegawa, ‘An incremental network for on-line unsupervised classification and topology learning’, *Neural Networks*, **19**, 90–106, (2006).

[8] Herbert Jaeger, ‘Adaptive nonlinear system identification with echo state networks’, in *Neural Information Processing Systems*, pp. 593–600. MIT Press, (2002).

[9] Kazuhiko Kawamura, Stephen M. Gordon, Palis Ratanaswasd, Erdem Erdemir, and Joseph F. Hall, ‘Implementation of cognitive control for a humanoid robot’, *International Journal of Humanoid Robotics*, **5**(4), 547–586, (2008).

[10] Stephan Kirstein and Heiko Wersing, ‘A biologically inspired approach for interactive learning of categories’, in *Proceedings of the International Conference on Development and Learning*, pp. 1–6. IEEE, (2011).

[11] Stanley B. Klein, Leda Cosmides, Cynthia E. Gangi, Betsy Jackson, John Tooby, and Kristi a. Costabile, ‘Evolution and Episodic Memory: An Analysis and Demonstration of a Social Function of Episodic Recollection’, *Social Cognition*, **27**(2), 283–319, (April 2009).

[12] Jean-Charles Lamirel, Ghada Safi, Navesh Priyankar, and Pascal Cuxac, ‘Mining research topics evolving over time using a diachronic multi-source approach’, in *International Conference on Data Mining Workshops (ICDMW)*, pp. 17–24. IEEE, (2010).

[13] Hans J Markowitsch and Angelica Staniloiu, ‘Amnesic disorders’, *The Lancet*, **6736**(11), 1–12, (April 2012).

[14] Yann Prudent and Abdellatif Ennaji, ‘An incremental growing neural gas learns topologies’, in *Proceedings of the International Joint Conference on Neural Networks*, volume 2, pp. 1211–1216. IEEE, (2005).

[15] Lawrence R. Rabiner, ‘A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition’, *Proceedings of the IEEE*, **77**(2), 257–267, (1989).

[16] Ron Sun and C. Lee Giles, ‘Sequence learning: From recognition and prediction to sequential decision making’, *IEEE Intelligent Systems*, **16**(4), 67–70, (2001).

[17] Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning – An Introduction*, MIT Press, 4th edn., 2002.

[18] Marko Tscherepanow, ‘TopoART: A topology learning hierarchical ART network’, in *Proceedings of the International Conference on Artificial Neural Networks*, volume 6354 of *LNCS*, pp. 157–167. Springer, (2010).

[19] Marko Tscherepanow, ‘An extended TopoART network for the stable on-line learning of regression functions’, in *Proceedings of the International Conference on Neural Information Processing*, volume 7063 of *LNCS*, pp. 562–571. Springer, (2011).

[20] Marko Tscherepanow, Marko Kortkamp, and Marc Kammer, ‘A hierarchical ART network for the stable incremental learning of topological structures and associations from noisy data’, *Neural Networks*, **24**(8), 906–916, (2011).

[21] Endel Tulving, ‘Episodic memory: from mind to brain’, *Annual Review of Psychology*, **53**, 1–25, (2002).

[22] Thomas Voegtlin, ‘Recursive self-organizing maps’, *Neural Networks*, **15**(8–9), 979–991, (2002).

[23] Rui Xu and Donald C. Wunsch II, *Clustering*, Wiley–IEEE Press, 2009.