An Approach to Numeric Refinement in Description Logic Learning for Learning Activities Duration in Smart Homes

An C. Tran, Hans W. Guesgen, Jens Dietrich, Stephen Marsland
School of Engineering and Advanced Technology
Massey University
New Zealand

Abstract

In spatio-temporal reasoning, granularity is one of the factors to be considered when aiming at an effective and efficient representation of space and time. There is a large body of work which addresses the issue of granularity by representing space and time on a qualitative level. Other approaches use a predefined scale which implicitly determines granularity (e.g., seconds, minutes, hours, days, month, etc.). However, there are situations where the right level of granularity is unknown in the beginning, and is only determined in the problem solving process itself. This is the case in machine learning, where the learner has to find a representation for a problem and with that the right granularity for representing space and time. This paper introduces an algorithm which determines the most appropriate level of granularity during training. It uses several description logic learners as the learners, and the positive and negative examples presented to them as the determinators for refining coarse temporal representations to the most appropriate level of granularity.

Introduction

One of the main tasks in smart homes is recognising human behaviour. The approaches to behaviour recognition that have been developed over the last ten years range from logic-based approaches to probabilistic machine learning approaches (Augusto and Nugent 2004; Chua, Marsland, and Guesgen 2011; Gopalratnam and Cook 2004; Rivera-Illingworth, Callaghan, and Hagras 2007; Tapia, Intille, and Larson 2004). In the majority of these approaches, recognition is based on information obtained from a sensor stream. Although the reported successes with these approaches are promising, determining the correct behaviour from sensor data alone is often impossible, since sensors can only provide very limited information and human behaviours are inherently complex.

Several researchers have argued that contextual information, in particular temporal information, can be useful to improve the behaviour recognition process (Aztiria et al. 2008; Guesgen and Marsland 2011; Jakkula and Cook 2008; Tavenard, Salah, and Pauwels 2007). For example, breakfast usually occurs in the morning, which means that if something is happening at 19:00, it is more likely to be some other behaviour than breakfast.

In this paper, we discuss how temporal information can efficiently be dealt with in a particular approach to behaviour recognition that bases on description logic learning. An investigation of the existing description logic learning algorithms reveals that these algorithms do not have efficient means to learn numeric data properties, such as temporal data, since they mainly focus upon the concept hierarchy and object properties. Appropriate refinement strategies for temporal data is missing from the existing studies.

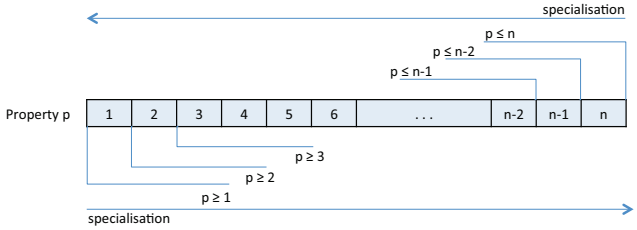
The area of qualitative spatial temporal reasoning (QSR) offers a number of formalisms to deal with temporal information on a qualitative scale. Since such approaches allow for coarse and vague representations of temporal information, they potentially are useful in a process where specialisation and generalisation is of importance. Most qualitative approaches to temporal reasoning are based on relations between objects (such as time intervals), such as Allen's temporal logic (Allen 1983) and the point algebra (Vilain and Kautz 1986). However, in some real-world application, numeric data cannot always be replaced by qualitative representations, since refinement is not always possible along fixed qualitative notions but requires a refinement of the numerical information itself.

The most common approach for refinement of numeric data is to use the *maximal* ($\leq$) and *minimal* ($\geq$) restriction operators on a set of values, as implemented in CELOE and OCEL. Figure 1 visualises the specialisation process for a data property *p*. Specialisation in Figure 1(a) uses either maximal or minimal value restriction while both of these restrictions are used in Figure 1(b).

Given *n* possible values for the refinement, there are total $2n$ possible combinations for specialisation of the data property *p* in Figure 1(a) and $\frac{n(n+1)}{2}$ combinations for the specialisation in Figure 1(b). The set of values used for the refinement are usually based on the range values of the data property assertions. To the best of our knowledge, the only description logic learning algorithms that explicitly describes and implemented the specification of the numeric data property refinement are CELOE, OCEL and ELTL based algorithms in the DL-Learner framework. As the number of the assertions of a data property may be very big, a

Figure 1: Specialisation of numeric data properties.

(a) Either maximal or minimal value restriction is used by the specialisation.



(b) Both maximal and minimal value restrictions are used by the specialisation.

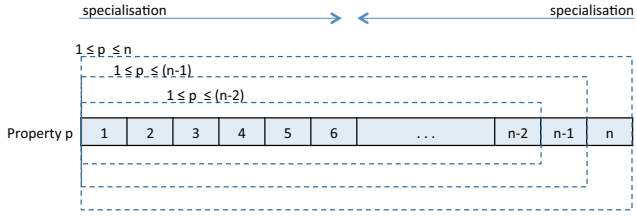
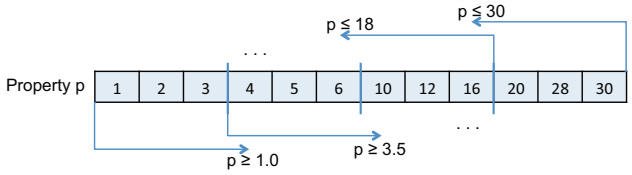


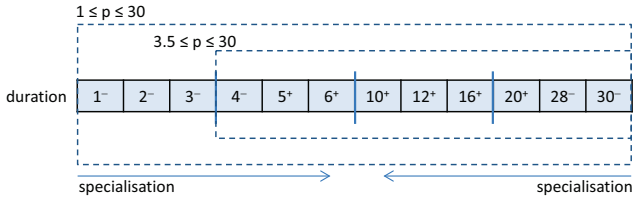
Figure 2: Segmentation of a data property assertion values to reduce the refinement space. The refinement space of this property decreases from 102 possible combinations (w.r.t 12 values) to 25 (w.r.t 5 values).



fixed-size segment of the refinement values is used to reduce the refinement space. Given k is the maximal number of values used for the refinement of a data property, the assertion values of the data property are split into k equal parts and the middle values of split parts are used for refinement (they are rounded for the integer datatype). Figure 2 describes a segmentation for a data property p that has 12 values into 4 parts.

However, the fixed-size segmentation for the data properties may lead to a circumstance that some *necessary values* are ignored while *redundant values* are included the set of values used for the refinement. For example, given the data property p in Figure 2 describes the durations of showering activities and the values are retrieved from the normal and abnormal showers, in which the set of values $\{5, 6, 10, 12, 16, 20\}$ are durations of normal shower and $\{1, 2, 3, 4, 28, 30\}$ are durations of abnormal showers. It is obvious that the segmentation in Figure 1 is not appropriate and it prevents the specialisation working correctly. Figure 3 demonstrates this problem. The expression $3.5 \leq \texttt{duration} \leq 30$ is overly general (complete and not correct) but it cannot be specialised as the next values for the specialisation (8 and 18) produces overly specific concepts.

Figure 3: Overly general expression, $3.5 \leq \texttt{duration} \leq 30$ cannot be further specialised due to an inappropriate segmentation of the data property values. Superscript $^+$ indicates the duration of normal showers and superscript $^-$ indicates duration of abnormal showers.



Therefore, fixed segmentation for data properties values may lead to the inappropriate segmentations. Using all values of the data properties values can help to avoid the problem but it may exploit the refinement spaces with unnecessary values. For example, some values such as $\{2, 3, 6, 10, 12, 16\}$ in Figure 2 can be eliminated from the segmentation without any problem. However, the relations between the values of data properties and the examples are not explicitly available. Therefore, we approach this problem by propose a method that creates a relation graph between literals of data properties and examples, and segments the data properties values based on the relation graph.

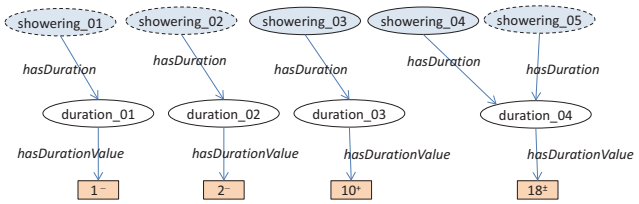
Our Method for Numeric Refinement in Description Logic Learning

As was described in the problem description, the aim of the data property values segmentation is to identify the set of values that can be used for refinement of numeric data properties such that the *redundant values* are eliminated to reduce the refinement space of the data properties. In addition, the segmentation method should also ensure that the set of values contains *necessary values* for constructing the definitions for positive examples.

Figure 3 gives an example of an inappropriate segmentation that misses some *necessary values* for constructing the definitions. The set values for refinement of the *duration* property in this segmentation method is $\{1, 3.5, 8, 18, 30\}$. However, we need some values between 4 and 5, and 20 and 28 to construct the definitions for positive examples. If the above values are segmented into 6 parts, the set of values for the segmentation is $\{1, 2.5, 4.5, 8, 14, 24, 30\}$ and thus we can get the necessary values. However, this segmentation produce *redundant values*, e.g. $\{2.5, 8, 14\}$.

Therefore, to eliminate the redundancy and to avoid missing the necessary values, our approach to segmentation of data properties values requires the information about the relations between the values of the data properties and the examples. We identify these relations by using a *relation graph*. A relation graph is a directed graph that represents the relations between the individuals and the literals based on the assertions in an ABox. The nodes in a relation graph are the class assertions or literals of data properties and the

Figure 4: A relation graph. Shaded eclipse nodes are examples in which solid lines represent positive examples and dashed lines represent negative examples. Unshaded eclipse nodes are other class assertions. Rectangle nodes represent values (literals) of data properties. Superscript $^+$, $^-$ or $^\pm$ of a value implies the value has relation with only positive example(s), only negative example(s), or both positive and negative examples respectively. The edges represent properties assertions.



edges are the property assertions that connect their domain value (individuals) to their range value (individuals or literals). Figure 4 shows a simple relation graph that describes the relations between some showering examples and the values of `hasDurationValue` data property.

Given a relation graph, if there exists a path from an example to a value, we said the value is *related* to that example. Each value of a data property may be related only to the positive or negative example(s) or both types of example. For example, in Figure 4, the literals "1" and "2" of the data property `hasDurationValue` relate only to the negatives examples (denoted by +), literal "18" related only to the negative example (denoted by –) and the literal "18" relates to both positive and negative examples (denoted by $\pm$).

To segment the values of each data property, they are firstly sorted by an arbitrary order. Then, it is obvious that for the values types "+" or "–", jumping through the values with the same type in the specialisation does not effect on the overly general or overly specific property of the refined concept. For example, given the values for the data property `hasDurationValue` as in Figure 3 and an expression `hasDurationValue some double`$[\geq 1]$, then the specialisation of the expression by increasing 1 to 2, 3 and 4 will not result an overly specific expression. This property may only be changed when we move to a values with another type, i.e. from 4 to 5. Therefore, these values can be considered as redundant values for specialisation. We only need the values at the boundaries of each group for the segmentation. For the values with $\pm$ type, they cannot alone be able to distinguish the positive examples. Therefore, no redundancy strategy is proposed for this type of values. They are segmented value by value.

Finally, the set of values used for the specialisation of each data property is computed from the values at the boundary of the segments. They may be the average of the two values at the boundary. Rounding may be needed for integer data properties (i.e. data properties with datatype of their range is integer). Figure 5 demonstrates a segmentation and the computation of the values for specialisation with 5 values are computed for the specialisation. Figure 6 gives a particular example for segmenting the values in Figure 6. There are three segments and only two values are computed for the specialisation.

Figure 5: Segmentation of data property values. Values are sorted and then grouped by type. There are six segments from s1 to s6 and five values are computed for specialisation.

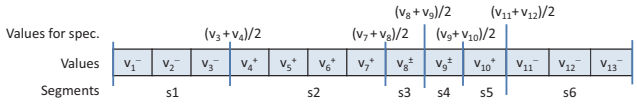
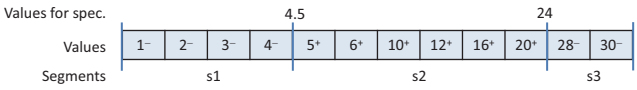


Figure 6: Applying our segmentation method for segmenting the values in Figure 3 and computing the values for the specification. There are three segments and three values computed.



The Algorithm

Our data property values segmentation aims to reduce redundant and avoid missing of necessary values in refinement of numeric data properties. Our approach bases on the directed graph that represents relations examples and values of the numeric data properties. Therefore, the first step in our algorithm is to build a relation graph. Then, we check if a value is connected with positive or negative examples or both of these types of example. Finally, we segment the values of numeric data properties using the above information.

The algorithm for building relation graph is given in Algorithm 1. Each vertex v in the relation graph has the form of $(startnode, endnode, label)$, in which *label* is the label of v. Algorithm 1 represents simple algorithm for creating a relation graph. However, this graph is redundant as we only need to find paths from examples to literal. Therefore, some properties assertions can be eliminated from the graph.

Algorithm 2 shows our approach in segmentation of the values of a numerical data property. In this algorithm, function EXITPATH checks for the existence of a path between two nodes in the relation graph such that one of the vertices must have a given label. In addition, a function AVERAGE calculates the average of to values depending on datatype of those values.

Evaluation

This evaluation is to investigate the effect of segmentation strategy on the learning result. This strategy aims to reduce the unnecessary values of a numeric data property and avoid missing of necessary values for refinement. Reduction of unnecessary value can help to reduce the search tree size and avoidance of missing necessary values can help to prevent the missing of accurate definitions in refinement.

In this evaluation, we will compare our strategy with a popular segmentation strategy used in other learning al-

Algorithm 1: Relation graph builder - RGRAPHBUILDER($\mathcal{T}, \mathcal{A}$)

Input: an ontology $\mathcal{O}$ including a TBox $\mathcal{T}$, consisting of a set of concept names N_C and role names N_R, and an ABox $\mathcal{A}$.

Output: a labelled directed graph $G = (V, E)$ such that for each property assertion *property*(*domain*, *range*) $\in \mathcal{A} : domain \in E$, $range \in E$ and there exists a vertex *v* = (*domain, range, property*) $\in V$, in which *domain* and *range* are two vertices of *v* and *property* is the label of *v*.

begin
  $E = \emptyset$ /* set of edges */
  $V = \emptyset$ /* set of vertices */
  foreach $property \in N_R$ **and** $property(domain, range) \in \mathcal{A}$ **do**
    if *domain* is not $\in E$ **then**
      $E = E \cup domain$
    if *range* is not $\in E$ **then**
      $E = E \cup range$
    if $(domain, range, property)$ is not $\in V$ **then**
      $V = V \cup (domain, range, property)$
      /* see text */
return $G = (V, E)$

gorithms, i.e. the fixed segmentation. We firstly chose the learning algorithms for implementation our strategy. These algorithms must also supported fixed segmentation strategy for comparison with our strategy. Then, we selected datasets used for the evaluation. The selection includes noisy and noiseless datasets. Finally, we select the experiment methodology for running the experiment. They are described below.

Evaluation algorithms

We implement our segmentation strategy in three description learning algorithms. The first algorithm used is CELOE, one of the best algorithms in DL-Learner framework (Lehmann and Hitzler 2010). This is a top-down description logic learning learning algorithm. The second algorithm is ParCEL, a parallel description logic learning algorithm (Tran et al. 2012a). This is basically a top-down approach but it use disjunction as a generalisation to combine partial solutions into an overall solution. The third algorithm is ParCELEx, a two-way parallel class expression learning algorithm proposed in (Tran et al. 2012b). These algorithms give the best experimental result in comparison with existing description logic learning algorithms (Fahnrich, Lehmann, and Hellmann 2008; Lehmann 2010; Tran et al. 2012b).

For each algorithm, we implemented two segmentation strategies that can be selected before running the experiment.

Algorithm 2: Numeric data property segmentation – SEGMENT($\mathcal{E}^+, \mathcal{E}^-, pro$)

Input: a relation graph $G = (V, E)$, a set of positive examples $\mathcal{E}^+$, a set of negative examples $\mathcal{E}^-$, and a numeric data property *pro*.

Output: a set of values $\{v_1, v_2, \ldots, v_n\}$ for refinement of the data property *pro*.

begin
  values = $\{val : val.value = x, val.type = \text{nil} \mid v = (\_, x, property) \in V\}$
  seg_values = $\emptyset$ /* segmented values */
  /* identify *type* of the *values* */
  foreach *val* $\in$ *values* **do**
    if $\exists e \in \mathcal{E}^+ |$ EXITPATH*(e, val.x, pro, G)* = true **then** /* see text */
      val.type = pos /* val is connected to positive example(s) */
    if $\exists e \in \mathcal{E}^- |$ EXITPATH*(e, val.x, pro, G)* = true **then**
      if *val.type* = nil **then**
        val.type = neg
      else
        val.type = both
  sort *values* by its elements value
  /* segment the *values* */
  for *i=1* **to** *values.length-1* **do**
    if $values[i].type \neq values[i+1].type$ **or** $values[i].type$ = *both* **then**
      seg_values = *seg_values* $\cup$ AVERAGE(*values*[i].*val*, *values*[i+1].val)
return *seg_values*

Evaluation datasets

We select UCA1 and ILPD datasets for our evaluation. UCA1 is a smart home dataset that is simulated based on a smart home use case (Tran et al. 2010). This dataset aims describes the normal and abnormal showers duration. It is available at the DL-Learner repository[1]. Showers duration in this dataset is assumed to depend upon the seasons of the year, e.g. summer, winter, etc. This is a noiseless datasets and the definition of a normal shower is reported around 30 to 50 axioms.

The second dataset used in our evaluation is a noisy dataset ILPD (Indian Liver Cancer Dataset). This dataset contains liver function test records of patients collected from the North East of Andhra Pradesh, India. More detailed description and the data of this dataset is available at the UCI repository [2].

Evaluation methodology

We use 10-fold cross validation method (Kohavi and others 1995) to run the experiments and measure the predictive accuracy and search tree size. For each algorithm, we run two experiment on each evaluation dataset: the first experiment uses the *fixed size segmentation strategy* and the second experiment uses our proposed segmentation strategy (we called *adaptive segmentation strategy*). The reported evaluation result for each metric is the average and standard deviation of the 10 folds.

Evaluation result

Tables 1 and 2 show the experimental results on UCA1 and ILPD datasets respectively. The results exhibited in Table 1 show that our proposed segmentation strategy helps to improve the predictive accuracy for both CELOE and ParCEL on the UCA1 dataset. A t-test on this results suggests that the improvement of the predictive accuracy is statistically significant at 99% significance level.

The improvement is also achieved by ParCEL on the ILPD dataset as shown in Table 2. Since this dataset is unbalanced, we present both predictive accuracy and balanced predictive accuracy results for a more thorough assessment. A t-test also suggests that the improvement is statistically significant at the 99% confidence level, applied for both types of accuracy. However, the segmentation does not help to improve the (balanced) predictive accuracy of CELOE this dataset. One possibly reason is that this dataset is noisy. Therefore, the learning not only effected by appropriation of the refinement but also the distraction of noise.

Table 3 summarises the reduction of numeric data properties values by our segmentation strategy for both UCA1 and ILPD datasets. Overall, most values of numeric data properties in these datasets used for refinement are reduced significantly.

The experimental result suggests that the reduction of the data property values cut down the search tree size significantly. Table 4 summaries the search tree size generated by

Table 1: Predictive accuracy on the UCA1 dataset.

Segmentation strategy	Algorithm		
	CELOE	ParCEL	ParCELEx
Fixed size	91.24 ± 6.4	94.70 ± 7.43	92.08 ± 6.82
Adaptive	94.04 ± 0	100.00 ± 0	100.00 ± 0

Table 2: Predictive accuracy on the ILPD dataset.

Segmentation strategy	Algorithm		
	CELOE	ParCEL	ParCELEx
	Accuracy		
Fixed size	76.02 ± 3.43	54.96 ± 4.43	53.457 ± 9.12
Adaptive	76.01 ± 2.61	71.12 ± 5.36	72.671 ± 8.12
	Balanced accuracy		
Fixed size	64.62 ± 4.83	64.63 ± 6.07	63.171 ± 10.09
Adaptive	64.76 ± 5.83	70.94 ± 10.87	71.734 ± 12.29

Table 3: Reduction of numeric data properties values resulted by our segmentation strategy *(averages ± standard deviations of 10 folds)*.

Property	No of values	No of reduced values
UCA1 dataset		
hasDurationValue	37	6
hasMonthValue	12	12
ILPD dataset		
hasAlbumin	142	67
hasTotalBilirubin	102	33
hasDirectBilirubin	74	24
hasAGRatio	163	81

[1] http://dl-learner.svn.sourceforge.net/viewvc/dl-learner/trunk/examples/showering-duration/

[2] http://archive.ics.uci.edu/ml/datasets/ILPD+(Indian+Liver+Patient+Dataset)

Table 4: Search tree size generated by the algorithms on UCA1 dataset.

Segmentation strategy	Algorithm		
	CELOE	ParCEL	ParCELEx
Fixed size	22,688.7 ± 7,939.6	83,885.0 ± 72,737.9	243,512.0 ± 128,303.8
Adaptive	11,979.7 ± 3,520.3	26,245.4 ± 29,889.6	442.0 ± 148.3

three algorithms in learning UCA1 dataset using two segmentation strategies. CELOE could not find accurate definition on training sets on both cases, therefore the search tree sizes reported are the size of the search trees at which the best solutions were generated. It is treated similarly for ParCEL and ParCELEx when they use the default segmentation strategy. Out t-test on this result shows that the reduction of search tree sizes is statistically significant at 99% significance level.

Conclusion

We implemented our strategy in three description learning algorithms and evaluated using a smart home and a healthcare datasets. Our numeric data property segmentation strategy exhibits a promising result. It helps to reduce the number of values used in refinement of numeric data properties significantly. As a result, the search tree generated is significant smaller. More importantly, our experimental results show that this reduction does not decrease the predictive accuracy. Moreover, it increases the predictive accuracy in several learning problems by avoid missing necessary values in refinement. The reduction of unnecessary values in refinement may also indirectly increase the predictive accuracy as it help to avoid the distraction from those values. When the search tree is smaller with featured values, the change for reaching an accurate definition is higher.

Our approach represents a basic idea for segmentation of numeric data properties for description logic learning. Although its efficacy is demonstrated though the above evaluations on basic metrics such as the reduction of the number of values used in refinement, the reduction of and the search tree sizes or the improvement on learning time and predictive accuracy, there still have some problems need to be addressed. The first problem is on the segmentation of values that are connected to both positive and negative examples. It might be useful to look into other properties to reduce the number of segments for this type of value. The second problem is to utilise meta data for more complex numeric data patterns such as leap-years. Therefore, for a more adaptive numeric learning strategy, not only appropriate segmentation strategies but also more complex refinement operators are needed (e.g. modulo operator for the above example).

Related work

Amongst the existing description and class expression learning algorithms such as $\mathcal{AL}$-QUIN (Lisi and Malerba 2003), DL-FOIL (Fanizzi, dAmato, and Esposito 2008), YinYang (Iannone, Palmisano, and Fanizzi 2007) and a set of learning algorithms in the DL-Learner framework such as CELOE, OCEL, ELTL (Lehmann 2010), numeric data property learning strategy was only explicitly described and supported by the algorithms in DL-Learner framework. As mentioned above, a fixed size segmentation strategy is used for this family of algorithms.

In inductive logic programming, the root of description logic learning, there were more strategies proposed. For example, in Aleph (Srinivasan 2004), several strategies were supported such as guessing the values used for refinement or using lazy evaluation (this strategy is not described in detail), etc. However, it was reported that these approaches might not always work. Fixed increment and decrement are also used in Aleph. This is used more commonly for integer datatype. For example, the value of an attribute is increased or decreased by a fixed value, e.g. 1, 2, etc. in each refinement step. Closer to our approach, a modification of discretisation method proposed in (Fayyad and Irani 1993) was used in the Inductive Constraint Logic (ILC) system (Van Laer, De Raedt, and Dzeroski 1997). The essential idea of discretisation of continuous data proposed by Fayyad and Irani is based on the *information entropy*. *Potential cut points* (segmentations) are estimated using a class entropy function. However, when this approach was used in ILC, it was reported that very few subintervals were generated and therefore they combined discretisation approach with a fixed numbers of values desired to be generated, e.g. 10 values or 20 val

References

Allen, J. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM* 26:832–843.

Augusto, J., and Nugent, C. 2004. The use of temporal reasoning and management of complex events in smart homes. In *Proc. ECAI-04*, 778–782.

Aztiria, A.; Augusto, J.; Izaguirre, A.; and Cook, D. 2008. Learning accurate temporal relations from user actions in intelligent environments. In *Proceedings of the 3th Symposium of Ubiquitous Computing and Ambient Intelligence*.

Chua, S.-L.; Marsland, S.; and Guesgen, H. 2011. Unsupervised learning of patterns in data streams using compression and edit distance. In *Proc. IJCAI*, 1231–1236.

Fahnrich, K.; Lehmann, J.; and Hellmann, S. 2008. Comparison of concept learning algorithms.

Fanizzi, N.; dAmato, C.; and Esposito, F. 2008. Dl-foil concept learning in description logics. *Inductive Logic Programming* 107–121.

Fayyad, U., and Irani, K. 1993. Multi-interval discretization of continuous-valued attributes for classification learning. In *The 13th International Joint Conference on Artificial Intelligence*, 1022–1027.

Gopalratnam, K., and Cook, D. 2004. Active LeZi: An incremental parsing algorithm for sequential prediction. *International Journal of Artificial Intelligence Tools* 14(1–2):917–930.

Guesgen, H., and Marsland, S. 2011. Recognising human behaviour in a spatio-temporal context. In Chong, N.-Y., and Mastrogiovanni, F., eds., *Handbook of Research on Ambient Intelligence and Smart Environments: Trends and Perspective*. Hershey, Pennsylvania: IGI Global. 443–459.

Iannone, L.; Palmisano, I.; and Fanizzi, N. 2007. An algorithm based on counterfactuals for concept learning in the semantic web. *Applied Intelligence* 26(2):139–159.

Jakkula, V., and Cook, D. 2008. Anomaly detection using temporal data mining in a smart home environment. *Methods of Information in Medicine*.

Kohavi, R., et al. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International joint Conference on artificial intelligence*, volume 14, 1137–1145. Lawrence Erlbaum Associates Ltd.

Lehmann, J., and Hitzler, P. 2010. Concept learning in description logics using refinement operators. *Machine Learning* 78(1):203–250.

Lehmann, J. 2010. *Learning OWL Class Expressions*. AKA Akademische Verlagesellschaft.

Lisi, F., and Malerba, D. 2003. Ideal refinement of descriptions in al-log. *Inductive Logic Programming* 215–232.

Rivera-Illingworth, F.; Callaghan, V.; and Hagras, H. 2007. Detection of normal and novel behaviours in ubiquitous domestic environments. *The Computer Journal*.

Srinivasan, A. 2004. *The Aleph Manual*.

Tapia, E.; Intille, S.; and Larson, K. 2004. Activity recognition in the home using simple and ubiquitous sensors. *Pervasive Computing* 158–175.

Tavenard, R.; Salah, A.; and Pauwels, E. 2007. Searching for temporal patterns in ami sensor data. In *Proceedings of AmI2007*, 53–62.

Tran, A.; Marsland, S.; Dietrich, J.; Guesgen, H.; and Lyons, P. 2010. Use cases for abnormal behaviour detection in smart homes. *Aging Friendly Technology for Health and Independence* 144–151.

Tran, A.; Dietrich, J.; Guesgen, H.; and Marsland, S. 2012a. An approach to parallel class expression learning. *Rules on the Web: Research and Applications* 302–316.

Tran, A. C.; Dietrich, J.; Guesgen, H. W.; and Marsland, S. 2012b. Two-way parallel class expression learning. *Journal of Machine Learning Research - Proceedings Track* 25:443–458.

Van Laer, W.; De Raedt, L.; and Dzeroski, S. 1997. *On multi-class problems and discretization in inductive logic programming*. Springer.

Vilain, M., and Kautz, H. 1986. Constraint propagation algorithms for temporal reasoning. In *Proc. AAAI-86*, 377–382.